



Array Matrix Strings Hashing Linked List Stack Queue Binary Tree Binary Search Tree Heap Graph Searching Sorting D

A program to check if a binary tree is BST or not

Difficulty Level : Medium • Last Updated : 18 Feb, 2022

A binary search tree (BST) is a node based binary tree data structure which has the following properties.

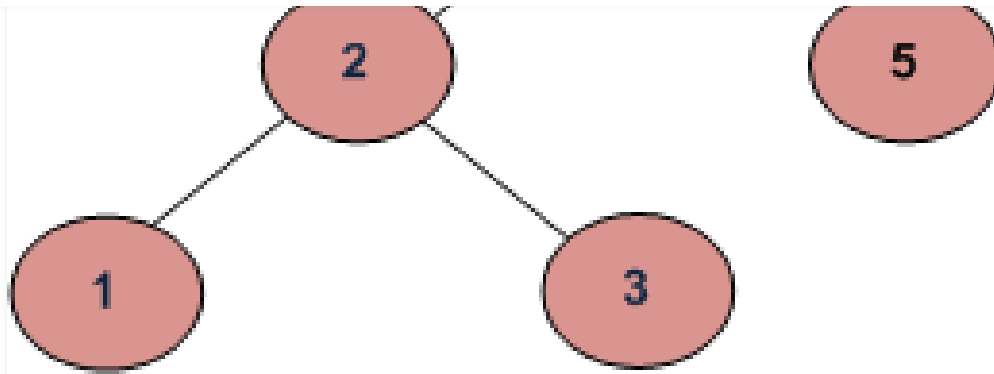
- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- Both the left and right subtrees must also be binary search trees.

From the above properties it naturally follows that:

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



Start Your Coding Journey Now!

[Login](#)[Register](#)

Recommended: Please solve it on “**PRACTICE**” first, before moving on to the solution.

METHOD 1 (Simple but Wrong)

Following is a simple program. For each node, check if the left node of it is smaller than the node and right node of it is greater than the node.

C++

```
int isBST(struct node* node)
{
    if (node == NULL)
        return 1;

    /* false if left is > than node */
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* false if, recursively, the left or right is not a BST */
if (!isBST(node->left) || !isBST(node->right))
    return 0;

/* passing all that, it's a BST */
return 1;
}
```

// This code is contributed by shubhamsingh10

C

```
int isBST(struct node* node)
{
    if (node == NULL)
        return 1;

    /* false if left is > than node */
    if (node->left != NULL && node->left->data > node->data)
        return 0;

    /* false if right is < than node */
    if (node->right != NULL && node->right->data < node->data)
        return 0;

    /* false if, recursively, the left or right is not a BST */
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

Java

```
boolean isBST(Node node)
{
    if (node == null)
        return true;

    /* False if left is > than node */
    if (node.left != null && node.left.data > node.data)
        return false;

    /* False if right is < than node */
    if (node.right != null && node.right.data < node.data)
        return false;

    /* False if, recursively, the left or right is not a BST */
    if (!isBST(node.left) || !isBST(node.right))
        return false;

    /* Passing all that, it's a BST */
    return true;
}

// This code is contributed by shubhamsingh10
```

Python3

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
1 if (node.left == None and node.right.data > node.data):  
    return 0  
  
''' false if right is < than node '''  
if (node.right != None and node.right.data < node.data):  
    return 0  
  
''' false if, recursively, the left or right is not a BST '''  
if (!isBST(node.left) or !isBST(node.right)):  
    return 0  
  
''' passing all that, it's a BST '''  
return 1
```

This code is contributed by Shubham Singh

C#

```
bool isBST(Node node)  
{  
    if (node == null)  
        return true;  
  
    /* False if left is > than node */  
    if (node.left != null && node.left.data > node.data)  
        return false;  
  
    /* False if right is < than node */  
    if (node.right != null && node.right.data < node.data)
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* Passing all that, it's a BST */  
return true;  
}  
  
// This code is contributed by Rajput-Ji
```

Javascript

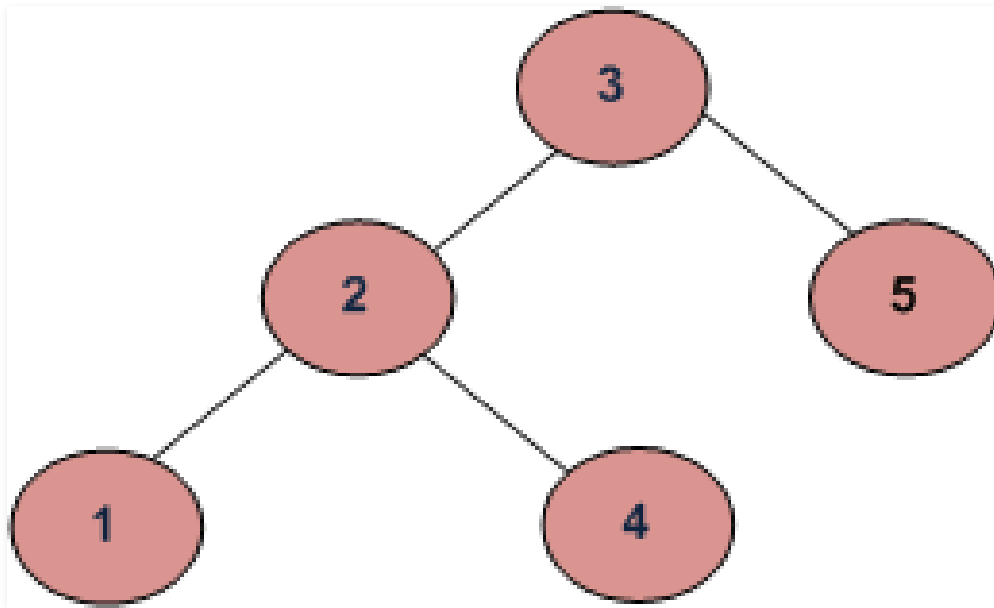
```
<script>  
  
function isBST(node)  
{  
    if (node == null)  
        return true;  
  
    /* False if left is > than node */  
    if (node.left != null && node.left.data > node.data)  
        return false;  
  
    /* False if right is < than node */  
    if (node.right != null && node.right.data < node.data)  
        return false;  
  
    /* False if, recursively, the left or right is not a BST */  
    if (!isBST(node.left) || !isBST(node.right))  
        return false;  
  
    /* Passing all that, it's a BST */
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

</script>

This approach is wrong as this will return true for below binary tree (and below tree is not a BST because 4 is in left subtree of 3)



METHOD 2 (Correct but not efficient)

For each node, check if max value in left subtree is smaller than the node and min value in right subtree greater than the node.

**C++**

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* false if the max of the left is > than us */
if (node->left != NULL && maxValue(node->left) >= node->data)
    return 0;

/* false if the min of the right is <= than us */
if (node->right != NULL && minValue(node->right) <= node->data)
    return 0;

/* false if, recursively, the left or right is not a BST */
if (!isBST(node->left) || !isBST(node->right))
    return 0;

/* passing all that, it's a BST */
return 1;
}

// This code is contributed by shubhamsingh10
```

C

```
/* Returns true if a binary tree is a binary search tree */
int isBST(struct node* node)
{
    if (node == NULL)
        return 1;

    /* false if the max of the left is > than us */
    if (node->left != NULL && maxValue(node->left) > node->data)
```


Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* false if, recursively, the left or right is not a BST */
if (!isBST(node->left) || !isBST(node->right))
    return 0;

/* passing all that, it's a BST */
return 1;
}
```

Java

```
/* Returns true if a binary tree is a binary search tree */
int isBST(Node node)
{
    if (node == null)
        return 1;

    /* false if the max of the left is > than us */
    if (node.left != null && maxValue(node.left) >= node.data)
        return 0;

    /* false if the min of the right is <= than us */
    if (node.right != null && minValue(node.right) <= node.data)
        return 0;

    /* false if, recursively, the left or right is not a BST */
    if (!isBST(node.left) || !isBST(node.right))
        return 0;
}
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

Python3

```
''' Returns true if a binary tree is a binary search tree '''
def isBST(node):
    if (node == None):
        return 1
    ''' false if the max of the left is > than us '''
    if (node.left != None and maxVal(node.left) >= node.data):
        return 0

    ''' false if the min of the right is <= than us '''
    if (node.right != None and minVal(node.right) <= node.data):
        return 0

    ''' false if, recursively, the left or right is not a BST '''
    if (!isBST(node.left) or !isBST(node.right)):
        return 0

    ''' passing all that, it's a BST '''
    return 1

# This code is contributed by Shubham Singh
```

C#

```
/* Returns true if a binary tree is a binary search tree */
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
if (node.left != null && maxValue(node.left) >= node.data)
    return false;


/* false if the min of the right is <= than us */
if (node.right != null && minValue(node.right) <= node.data)
    return false;

/* false if, recursively, the left or right is not a BST */
if (!isBST(node.left) || !isBST(node.right))
    return false;

/* passing all that, it's a BST */
return true;
}

// This code is contributed by Shubham Singh
```

Javascript



```
<script>

function isBST(node)
{
    if (node == null)
        return true;

    /* False if the max of the left is > than us */
    if (node.left != null && maxValue(node.left) >= node.data)
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* False if, recursively, the left or right is not a BST */
if (!isBST(node.left) || !isBST(node.right))
    return false;

/* Passing all that, it's a BST */
return true;
}

// This code is contributed by Shubham Singh

</script>
```

It is assumed that you have helper functions `minValue()` and `maxValue()` that return the min or max int value from a non-empty tree

METHOD 3 (Correct and Efficient):

Method 2 above runs slowly since it traverses over some parts of the tree many times. A better solution looks at each node only once. The trick is to write a utility helper function `isBSTUtil(struct node* node, int min, int max)` that traverses down the tree keeping track of the narrowing min and max allowed values as it goes, looking at each node only once. The initial values for min and max should be `INT_MIN` and `INT_MAX` — they narrow from there.

Note: This method is not applicable if there are duplicate elements with value `INT_MIN` or `INT_MAX`.

Below is the implementation of the above approach:



C++

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* Returns true if the given tree is a binary search tree (efficient version). */
a pointer to right child */
class node
{
    public:
    int data;
    node* left;
    node* right;

    /* Constructor that allocates
    a new node with the given data
    and NULL left and right pointers. */
    node(int data)
    {
        this->data = data;
        this->left = NULL;
        this->right = NULL;
    }
};

int isBSTUtil(node* node, int min, int max);

/* Returns true if the given
tree is a binary search tree
(efficient version). */
int isBST(node* node)
{
    return(isBSTUtil(node, INT_MIN, INT_MAX));
}

/* Returns true if the given
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        return 1;

    /* false if this node violates
    the min/max constraint */
    if (node->data < min || node->data > max)
        return 0;

    /* otherwise check the subtrees recursively,
    tightening the min or max constraint */
    return
        isBSTUtil(node->left, min, node->data-1) && // Allow only distinct values
        isBSTUtil(node->right, node->data+1, max); // Allow only distinct values
}

/* Driver code*/
int main()
{
    node *root = new node(4);
    root->left = new node(2);
    root->right = new node(5);
    root->left->left = new node(1);
    root->left->right = new node(3);

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

    return 0;
}
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

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

int isBSTUtil(struct node* node, int min, int max);

/* Returns true if the given tree is a binary search tree
   (efficient version). */
int isBST(struct node* node)
{
    return(isBSTUtil(node, INT_MIN, INT_MAX));
}

/* Returns true if the given tree is a BST and its
   values are >= min and <= max. */
int isBSTUtil(struct node* node, int min, int max)
{
    /* an empty tree is BST */
    if (node==NULL)
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* otherwise check the subtrees recursively,
tightening the min or max constraint */
return
    isBSTUtil(node->left, min, node->data-1) && // Allow only distinct values
    isBSTUtil(node->right, node->data+1, max); // Allow only distinct values
}

/* Helper function that allocates a new node with the
given data and NULL left and right pointers. */
struct node* newNode(int data)
{
    struct node* node = (struct node*)
                        malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}

/* Driver program to test above functions*/
int main()
{
    struct node *root = newNode(4);
    root->left = newNode(2);
    root->right = newNode(5);
    root->left->left = newNode(1);
    root->left->right = newNode(3);

    if(isBST(root))
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
}
```

Java

```
//Java implementation to check if given Binary tree
//is a BST or not

/* Class containing left and right child of current
node and key value*/
class Node
{
    int data;
    Node left, right;

    public Node(int item)
    {
        data = item;
        left = right = null;
    }
}

public class BinaryTree
{
    //Root of the Binary Tree
    Node root;

    /* can give min and max value according to your code or
    can write a function to find min and max value of tree. */
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
}

/* Returns true if the given tree is a BST and its
   values are >= min and <= max. */
boolean isBSTUtil(Node node, int min, int max)
{
    /* an empty tree is BST */
    if (node == null)
        return true;

    /* false if this node violates the min/max constraints */
    if (node.data < min || node.data > max)
        return false;

    /* otherwise check the subtrees recursively
       tightening the min/max constraints */
    // Allow only distinct values
    return (isBSTUtil(node.left, min, node.data-1) &&
            isBSTUtil(node.right, node.data+1, max));
}

/* Driver program to test above functions */
public static void main(String args[])
{
    BinaryTree tree = new BinaryTree();
    tree.root = new Node(4);
    tree.root.left = new Node(2);
    tree.root.right = new Node(5);
    tree.root.left.left = new Node(1);
    tree.root.left.right = new Node(3);
}
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
}
```

Python3

```
# Python program to check if a binary tree is bst or not

INT_MAX = 4294967296
INT_MIN = -4294967296

# A binary tree node
class Node:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Returns true if the given tree is a binary search tree
# (efficient version)
def isBST(node):
    return (isBSTUtil(node, INT_MIN, INT_MAX))

# Return true if the given tree is a BST and its values
# >= min and <= max
def isBSTUtil(node, mini, maxi):
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        return False

    # Otherwise check the subtrees recursively
    # tightening the min or max constraint
    return (isBSTUtil(node.left, mini, node.data - 1) and
            isBSTUtil(node.right, node.data + 1, maxi))

# Driver program to test above function
root = Node(4)
root.left = Node(2)
root.right = Node(5)
root.left.left = Node(1)
root.left.right = Node(3)

if (isBST(root)):
    print ("Is BST")
else:
    print ("Not a BST")

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

C#

```
using System;
```

```
// C# implementation to check if given Binary tree
//is a BST or not
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
public Node(int item)
{
    data = item;
    left = right = null;
}

public class BinaryTree
{
    //Root of the Binary Tree
    public Node root;

    /* can give min and max value according to your code or
    can write a function to find min and max value of tree. */

    /* returns true if given search tree is binary
    search tree (efficient version) */
    public virtual bool BST
    {
        get
        {
            return isBSTUtil(root, int.MinValue, int.MaxValue);
        }
    }

    /* Returns true if the given tree is a BST and its
    values are >= min and <= max. */
    public virtual bool isBSTUtil(Node node, int min, int max)
    {

```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* false if this node violates the min/max constraints */
if (node.data < min || node.data > max)
{
    return false;
}

/* otherwise check the subtrees recursively
tightening the min/max constraints */
// Allow only distinct values
return (isBSTUtil(node.left, min, node.data - 1) && isBSTUtil(node.right, node.data + 1, max))
}

/* Driver program to test above functions */
public static void Main(string[] args)
{
    BinaryTree tree = new BinaryTree();
    tree.root = new Node(4);
    tree.root.left = new Node(2);
    tree.root.right = new Node(5);
    tree.root.left.left = new Node(1);
    tree.root.left.right = new Node(3);

    if (tree.BST)
    {
        Console.WriteLine("IS BST");
    }
    else
    {
        Console.WriteLine("Not a BST");
    }
}
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

Javascript

```
<script>

// Javascript implementation to
// check if given Binary tree
// is a BST or not

/* Class containing left and right child of current
node and key value*/

class Node
{
    constructor(item)
    {
        this.data=item;
        this.left=this.right=null;
    }
}

//Root of the Binary Tree
let root;

/* can give min and max value according to your code or
can write a function to find min and max value of tree. */

/* returns true if given search tree is binary
search tree (efficient version) */
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* Returns true if the given tree is a BST and its
   values are >= min and <= max. */
function isBSTUtil(node,min,max)
{
    /* an empty tree is BST */
    if (node == null)
        return true;

    /* false if this node violates
    the min/max constraints */
    if (node.data < min || node.data > max)
        return false;

    /* otherwise check the subtrees recursively
    tightening the min/max constraints */
    // Allow only distinct values
    return (isBSTUtil(node.left, min, node.data-1) &&
            isBSTUtil(node.right, node.data+1, max));
}

/* Driver program to test above functions */
root = new Node(4);
root.left = new Node(2);
root.right = new Node(5);
root.left.left = new Node(1);
root.left.right = new Node(3);

if (isBST())
    document.write("IS BST<br>");
else
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

Output:

IS BST

Time Complexity: $O(n)$

Auxiliary Space: $O(1)$ if Function Call Stack size is not considered, otherwise $O(n)$

Simplified Method 3

We can simplify method 2 using NULL pointers instead of INT_MIN and INT_MAX values.

C++

```
// 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;
};

// Returns true if given tree is BST.
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
// if left node exist then check it has
// correct data or not i.e. left node's data
// should be less than root's data
if (l != NULL and root->data <= l->data)
    return false;

// if right node exist then check it has
// correct data or not i.e. right node's data
// should be greater than root's data
if (r != NULL and root->data >= r->data)
    return false;

// check recursively for every node.
return isBST(root->left, l, root) and
        isBST(root->right, root, r);
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct Node* newNode(int data)
{
    struct Node* node = new Node;
    node->data = data;
    node->left = node->right = NULL;
    return (node);
}

/* Driver program to test above functions*/
int main()
{
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

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

return 0;
}
```

Java

```
// Java program to check if a given tree is BST.
class Sol
{
    // A binary tree node has data, pointer to
    //left child && a pointer to right child /
    static class Node
    {
        int data;
        Node left, right;
    };

    // Returns true if given tree is BST.
    static boolean isBST(Node root, Node l, Node r)
    {
        // Base condition
        if (root == null)
            return true;
    }
}
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
// if right node exist then check it has
// correct data or not i.e. right node's data
// should be greater than root's data
if (r != null && root.data >= r.data)
    return false;

// check recursively for every node.
return isBST(root.left, l, root) &&
        isBST(root.right, root, r);
}

// Helper function that allocates a new node with the
//given data && null left && right pointers. /
static Node newNode(int data)
{
    Node node = new Node();
    node.data = data;
    node.left = node.right = null;
    return (node);
}

// Driver code
public static void main(String args[])
{
    Node root = newNode(3);
    root.left = newNode(2);
    root.right = newNode(5);
    root.left.left = newNode(1);
    root.left.right = newNode(4);
}
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
}
```

```
// This code is contributed by Arnab Kundu
```

Python3

```
""" Program to check if a given Binary
Tree is balanced like a Red-Black Tree """

# Helper function that allocates a new
# node with the given data and None
# left and right poers.
class newNode:

    # Construct to create a new node
    def __init__(self, key):
        self.data = key
        self.left = None
        self.right = None

# Returns true if given tree is BST.
def isBST(root, l = None, r = None):

    # Base condition
    if (root == None) :
        return True

    # if left node exist then check it has
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
# correct data or not i.e. right node's data
# should be greater than root's data
if (r != None and root.data >= r.data) :
    return False

# check recursively for every node.
return isBST(root.left, l, root) and \
       isBST(root.right, root, r)

# Driver Code
if __name__ == '__main__':
    root = newNode(3)
    root.left = newNode(2)
    root.right = newNode(5)
    root.right.left = newNode(1)
    root.right.right = newNode(4)
    #root.right.left.left = newNode(40)
    if (isBST(root, None, None)):
        print("Is BST")
    else:
        print("Not a BST")

# This code is contributed by
# Shubham Singh(SHUBHAMSINGH10)
```

**C#**

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
// A binary tree node has data, pointer to
//left child && a pointer to right child /
public class Node
{
    public int data;
    public Node left, right;
};

// Returns true if given tree is BST.
static Boolean isBST(Node root, Node l, Node r)
{
    // Base condition
    if (root == null)
        return true;

    // if left node exist then check it has
    // correct data or not i.e. left node's data
    // should be less than root's data
    if (l != null && root.data <= l.data)
        return false;

    // if right node exist then check it has
    // correct data or not i.e. right node's data
    // should be greater than root's data
    if (r != null && root.data >= r.data)
        return false;

    // check recursively for every node.
    return isBST(root.left, l, root) &&
        isBST(root.right, root, r);
}
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
Node node = new Node();
node.data = data;
node.left = node.right = null;
return (node);
}

// Driver code
public static void Main(String []args)
{
    Node root = newNode(3);
    root.left = newNode(2);
    root.right = newNode(5);
    root.left.left = newNode(1);
    root.left.right = newNode(4);

    if (isBST(root,null,null))
        Console.Write("Is BST");
    else
        Console.Write("Not a BST");
}
}
```

// This code is contributed by 29AjayKumar

Javascript



<script>

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        this.right = null;
        this.data = data;
    }
}

// Returns true if given tree is BST.
function isBST(root, l, r)
{
    // Base condition
    if (root == null)
        return true;

    // if left node exist then check it has
    // correct data or not i.e. left node's data
    // should be less than root's data
    if (l != null && root.data <= l.data)
        return false;

    // if right node exist then check it has
    // correct data or not i.e. right node's data
    // should be greater than root's data
    if (r != null && root.data >= r.data)
        return false;

    // check recursively for every node.
    return isBST(root.left, l, root) &&
        isBST(root.right, root, r);
}

// Helper function that allocates a new node with the
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
let root = newNode(3);
root.left = newNode(2);
root.right = newNode(5);
root.left.left = newNode(1);
root.left.right = newNode(4);

if (isBST(root,null,null))
    document.write("Is BST");
else
    document.write("Not a BST");

</script>
```

Output:

Not a BST

Thanks to [Abhinesh Garhwal](#) for suggesting above solution.

METHOD 4(Using In-Order Traversal)

Thanks to [LJW489](#) for suggesting this method.

- 1) Do In-Order Traversal of the given tree and store the result in a temp array.
- 2) This method assumes that there are no duplicate values in the tree
- 3) Check if the temp array is sorted in ascending order, if it is, then the tree is BST.



Start Your Coding Journey Now!

[Login](#)[Register](#)

C++

```
bool isBST(node* root)
{
    static node *prev = NULL;

    // traverse the tree in inorder fashion
    // and keep track of prev node
    if (root)
    {
        if (!isBST(root->left))
            return false;

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

        prev = root;

        return isBST(root->right);
    }

    return true;
}

// This code is contributed by rathbhupendra
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
// traverse the tree in inorder fashion and keep track of prev node
if (root)
{
    if (!isBST(root->left))
        return false;

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

    prev = root;

    return isBST(root->right);
}

return true;
}
```

Java

```
// Java implementation to check if given Binary tree
// is a BST or not

/* Class containing left and right child of current
node and key value*/
class Node
{
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        left = right = null;
    }
}

public class BinaryTree
{
    // Root of the Binary Tree
    Node root;

    // To keep track of previous node in Inorder Traversal
    Node prev;

    boolean isBST() {
        prev = null;
        return isBST(root);
    }

    /* Returns true if given search tree is binary
       search tree (efficient version) */
    boolean isBST(Node node)
    {
        // traverse the tree in inorder fashion and
        // keep a track of previous node
        if (node != null)
        {
            if (!isBST(node.left))
                return false;

            // allows only distinct values node
            if (prev != null && node.data <= prev.data )
                return false;

            prev = node;
        }
        return true;
    }
}
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* Driver program to test above functions */
public static void main(String args[])
{
    BinaryTree tree = new BinaryTree();
    tree.root = new Node(4);
    tree.root.left = new Node(2);
    tree.root.right = new Node(5);
    tree.root.left.left = new Node(1);
    tree.root.left.right = new Node(3);

    if (tree.isBST())
        System.out.println("IS BST");
    else
        System.out.println("Not a BST");
}
```

Python3

```
# Python implementation to check if
# given Binary tree is a BST or not

# A binary tree node containing data
# field, left and right pointers
class Node:
    # constructor to create new node
    def __init__(self, val):
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
# traversal
prev = None

# function to check if given binary
# tree is BST
def isbst(root):

    # prev is a global variable
    global prev
    prev = None
    return isbst_rec(root)

# Helper function to test if binary
# tree is BST
# Traverse the tree in inorder fashion
# and keep track of previous node
# return true if tree is Binary
# search tree otherwise false
def isbst_rec(root):

    # prev is a global variable
    global prev

    # if tree is empty return true
    if root is None:
        return True

    if isbst_rec(root.left) is False:
        return False
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
# store the current node in prev
prev = root
return isbst_rec(root.right)
```

```
# driver code to test above function
root = Node(4)
root.left = Node(2)
root.right = Node(5)
root.left.left = Node(1)
root.left.right = Node(3)
```

```
if isbst(root):
    print("is BST")
else:
    print("not a BST")
```

```
# This code is contributed by
# Shweta Singh(shweta44)
```

C#

```
// C# implementation to check if
// given Binary tree is a BST or not
using System;

/* Class containing left and
```


Start Your Coding Journey Now!

[Login](#)[Register](#)

```
public Node(int item)
{
    data = item;
    left = right = null;
}

public class BinaryTree
{
    // Root of the Binary Tree
    Node root;

    // To keep track of previous node
    // in Inorder Traversal
    Node prev;

    Boolean isBST()
    {
        prev = null;
        return isBST(root);
    }

    /* Returns true if given search tree is binary
    search tree (efficient version) */
    Boolean isBST(Node node)
    {
        // traverse the tree in inorder fashion and
        // keep a track of previous node
        if (node != null)
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        node.data <= prev.data )
            return false;
        prev = node;
        return isBST(node.right);
    }
    return true;
}

// Driver Code
public static void Main(String []args)
{
    BinaryTree tree = new BinaryTree();
    tree.root = new Node(4);
    tree.root.left = new Node(2);
    tree.root.right = new Node(5);
    tree.root.left.left = new Node(1);
    tree.root.left.right = new Node(3);

    if (tree.isBST())
        Console.WriteLine("IS BST");
    else
        Console.WriteLine("Not a BST");
}

// This code is contributed by Rajput-Ji
```



Javascript

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
class Node
{
    constructor(item)
    {
        this.data = item;
        this.left = this.right=null;
    }
}

// Root of the Binary Tree
let root;

// To keep track of previous node in Inorder Traversal
let prev;

function isBST()
{
    prev = null;
    return _isBST(root);
}

/* Returns true if given search tree is binary
   search tree (efficient version) */
function _isBST(node)
{
    // traverse the tree in inorder fashion and
    // keep a track of previous node
    if (node != null)
    {
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        prev = node;
        return _isBST(node.right);
    }
    return true;
}

/* Driver program to test above functions */
root = new Node(4);
root.left = new Node(2);
root.right = new Node(5);
root.left.left = new Node(1);
root.left.right = new Node(3);

if (isBST())
    document.write("IS BST");
else
    document.write("Not a BST");

// This code is contributed by unknown2108
</script>
```

The use of a static variable can also be avoided by using a reference to the prev node as a parameter.

C++



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

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
struct Node* left, *right;

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

};

bool isBSTUtil(struct Node* root, Node *&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 (prev != NULL && root->data <= prev->data)
            return false;

        prev = root;

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

    return true;
}
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
/* Driver program to test above functions*/
int main()
{
    struct Node *root = new Node(3);
    root->left      = new Node(2);
    root->right     = new Node(5);
    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;
}
```

Java

```
// Java program to check if a given tree is BST.
import java.io.*;

class GFG {
    /* A binary tree node has data, pointer to
    left child and a pointer to right child */
    public static class Node
    {
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        left = right = null;
    }
};

static Node prev;

static Boolean isBSTUtil(Node root)
{
    // traverse the tree in inorder fashion and
    // keep track of prev node
    if (root != null)
    {
        if (!isBSTUtil(root.left))
            return false;

        // Allows only distinct valued nodes
        if (prev != null &&
            root.data <= prev.data)
            return false;

        prev = root;

        return isBSTUtil(root.right);
    }
    return true;
}

static Boolean isBST(Node root)
{
    return isBSTUtil(root);
}
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
root.left = new Node(2);
root.right = new Node(5);
root.left.left = new Node(1);
root.left.right = new Node(4);

if (isBST(root))
    System.out.println("Is BST");
else
    System.out.println("Not a BST");
}
```

// This code is contributed by Shubham Singh

Python3

```
# Python3 program to check
# if a given tree is BST.
import math

# A binary tree node has data,
# pointer to left child and
# a pointer to right child
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None
```


Start Your Coding Journey Now!

[Login](#)[Register](#)

```
    if (isBSTUtil(root.left, prev) == True):
        return False

    # Allows only distinct valued nodes
    if (prev != None and
        root.data <= prev.data):
        return False

    prev = root
    return isBSTUtil(root.right, prev)

return True

def isBST(root):
    prev = None
    return isBSTUtil(root, prev)

# Driver Code
if __name__ == '__main__':
    root = Node(3)
    root.left = Node(2)
    root.right = Node(5)
    root.right.left = Node(1)
    root.right.right = Node(4)
    #root.right.left.left = Node(40)

    if (isBST(root) == None):
        print("Is BST")
    else:
        print("Not a BST")
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
// C# program to check if a given tree is BST.
using System;
public class GFG
{
    /* A binary tree node has data, pointer to
    left child and a pointer to right child */
    public class Node
    {
        public int data;
        public Node left, right;

        public Node(int data)
        {
            this.data = data;
            left = right = null;
        }
    };

    static Node prev;

    static Boolean isBSTUtil(Node root)
    {
        // traverse the tree in inorder fashion and
        // keep track of prev node
        if (root != null)
        {
            if (!isBSTUtil(root.left))
                return false;
        }
    }
}
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        return isBSTUtil(root.right);
    }
    return true;
}

static Boolean isBST(Node root)
{
    return isBSTUtil(root);
}

// Driver Code
public static void Main(String[] args)
{
    Node root = new Node(3);
    root.left = new Node(2);
    root.right = new Node(5);
    root.left.left = new Node(1);
    root.left.right = new Node(4);

    if (isBST(root))
        Console.WriteLine("Is BST");
    else
        Console.WriteLine("Not a BST");
}

// This code is contributed by Rajput-Ji
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
{
  constructor(data)
  {
    this.left = null;
    this.right = null;
    this.data = data;
  }
}

let prev;

function isBSTUtil(root)
{
  // traverse the tree in inorder fashion and
  // keep track of prev node
  if (root != null)
  {
    if (!isBSTUtil(root.left))
      return false;

    // Allows only distinct valued nodes
    if (prev != null && root.data <= prev.data)
      return false;

    prev = root;

    return isBSTUtil(root.right);
  }
  return true;
}
```



Start Your Coding Journey Now!

[Login](#)[Register](#)

```
let root = new Node(3);
root.left = new Node(2);
root.right = new Node(5);
root.left.left = new Node(1);
root.left.right = new Node(4);

if (isBST(root))
  document.write("Is BST");
else
  document.write("Not a BST");
```

```
// This code is contributed by divyeshrabadiya07.
</script>
```

Output:

Not a BST



Start Your Coding Journey Now!

[Login](#)[Register](#)

Sources:

http://en.wikipedia.org/wiki/Binary_search_tree

<http://cslibrary.stanford.edu/110/BinaryTrees.html>

Please write comments if you find any bug in the above programs/algorithms or other ways to solve the same problem.



Start Your Coding Journey Now!

Login

Register

Like 243

Previous

Construct BST from given preorder traversal | Set 1

Next


Print the longest leaf to leaf path in a Binary tree

RECOMMENDED ARTICLES

Page : 1 2 3

01 K'th Largest Element in BST when modification to BST is not allowed
19, Mar 15

05 Check whether a binary tree is a full binary tree or not | Iterative Approach
19, Dec 17

 **02** Iterative approach to check if a Binary Tree is BST or not
10, Dec 20

06 Check whether a given binary tree is skewed binary tree or not?
18, Nov 18

Start Your Coding Journey Now!

[Login](#)[Register](#)

04 Maximum sub-tree sum in a Binary Tree such that the sub-tree is also a BST

20, Mar 19

08 Find k-th smallest element in BST (Order Statistics in BST)

15, Feb 11

Article Contributed By :

**GeeksforGeeks**

Vote for difficulty

Current difficulty : [Medium](#)[Easy](#)[Normal](#)[Medium](#)[Hard](#)[Expert](#)

Improved By : shweta44, ChandrahasAbburi, shrikanth13, SHUBHAMSINGH10, rathbhupendra, andrew1234, 29AjayKumar, Rajput-Ji, sapnasingh4991, rwells1703, Abhijeet Kumar Srivastava, abdul kadir olia, rag2127, avanitrachhadiya2155, rameshtravel07, unknown2108, divyeshrabadiya07, surinderdawra388, angajala, akshitsaxenaa09, amartyaghoshgfg, adnanirshad158, simmytarika5

Article Tags : [Accolite](#), [Adobe](#), [Amazon](#), [Boomerang Commerce](#), [FactSet](#), [GreyOrange](#), [MakeMyTrip](#), [Microsoft](#), [OYO Rooms](#), [Qualcomm](#), [Snapdeal](#), [VMWare](#), [Walmart](#), [Wooker](#), [Binary Search Tree](#), [Tree](#)

Practice Tags : [VMWare](#), [Accolite](#), [Amazon](#), [Microsoft](#), [OYO Rooms](#), [Snapdeal](#), [FactSet](#), [MakeMyTrip](#), [Walmart](#),

Start Your Coding Journey Now!

[Login](#)[Register](#)

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

[Load Comments](#)

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

[About Us](#)[Careers](#)[In Media](#)

Learn

[Algorithms](#)[Data Structures](#)[SDE Cheat Sheet](#)

News

[Top News](#)[Technology](#)[Work & Career](#)

Languages

[Python](#)[Java](#)[CPP](#)

Web Development

[Web Tutorials](#)[Django Tutorial](#)[HTML](#)

Contribute

[Write an Article](#)[Improve an Article](#)[Pick Topics to Write](#)

Start Your Coding Journey Now!

[Login](#)[Register](#)

@geeksforgeeks , Some rights reserved

