Trees: Is This a Binary Search Tree?

by saikiran9194

Problem Submissions Leaderboard Discussions Editorial

Check out the resources on the page's right side to learn more about binary search trees. The video tutorial is by Gayle Laakmann McDowell, author of the best-selling interview book Cracking the Coding Interview.

For the purposes of this challenge, we define a *binary search tree* to be a *binary tree* with the following ordering properties:

- The $data$ value of every node in a node's left subtree is *less than* the data value of that node.
- The $data$ value of every node in a node's right subtree is *greater than* the data value of that node.

Given the root node of a binary tree, can you determine if it's also a binary search tree?

Complete the function in your editor below, which has 1 parameter: a pointer to the root of a binary tree. It must return a *boolean* denoting whether or not the binary tree is a binary search tree. You may have to write one or more helper functions to complete this challenge.

Note: A binary tree is not a binary search if there are duplicate values.

Input Format

You are not responsible for reading any input from stdin. Hidden code stubs will assemble a binary tree and pass its root node to your function as an argument.

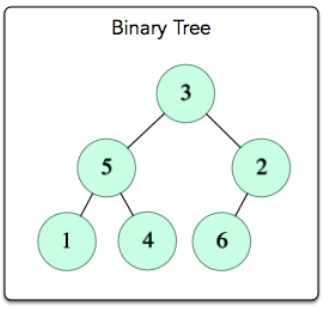
Constraints

- $0 \leq data \leq 10^4$

Output Format

You are not responsible for printing any output to stdout. Your function must return *true* if the tree is a binary search tree; otherwise, it must return *false*. Hidden code stubs will print this result as a *Yes* or *No* answer on a new line.

Sample Input



Sample Output

```
No
```

Explanation

The tree in the diagram does not satisfy the ordering property for a Binary Search Tree, so we print `No`.

Submissions: 14774
Max Score: 30
Difficulty: Medium

Rate This Challenge:

Need Help?

9:58

Binary Trees and Binary Search Trees

More

Current Buffer (saved locally, editable) C++

```cpp
/* Hidden stub code will pass a root argument to the function below. Complete the function to solve
the challenge. Hint: you may want to write one or more helper functions.

The Node struct is defined as follows:
   struct Node {
      int data;
      Node* left;
      Node* right;
   }
*/
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

bool 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 constraint */
  if (node->data < min || node->data > max)
     return false;

  /* 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
}

bool checkBST(Node* root) {
      return(isBSTUtil(root, INT_MIN, INT_MAX));
}
```

Line: 12 Col: 20

Upload Code as File Test against custom input Run Code Submit Code

Congrats, you solved this challenge!

- Test Case #0
- Test Case #1
- Test Case #2
- Test Case #3
- Test Case #4
- Test Case #5
- Test Case #6
- Test Case #7
- Test Case #8
- Test Case #9
- Test Case #10
- Test Case #11
- Test Case #12
- Test Case #13

Next Challenge