Domains Contests Jobs Rank Leaderboard

All Domains > Tutorials > 30 Days of Code > Day 22: Binary Search Trees

Share 30 Days of Code

Your Day 22: Binary Search Trees submission got 30.00 points. Share Tweet

Try the Next Challenge | Try a Random Challenge

Day 22: Binary Search Trees

by vatsalchanana

Problem Submissions Leaderboard Discussions Editorial Tutorial

Objective

Today, we're working with Binary Search Trees (BSTs). Check out the Tutorial tab for learning materials and an instructional video!

Task

The height of a binary search tree is the number of edges between the tree's root and its furthest leaf. You are given a pointer, $root$, pointing to the root of a binary search tree. Complete the *getHeight* function provided in your editor so that it returns the height of the binary search tree.

Input Format

The locked stub code in your editor reads the following inputs and assembles them into a binary search tree:
The first line contains an integer, n, denoting the number of nodes in the tree.
Each of the n subsequent lines contains an integer, $data$, denoting the value of an element that must be added to the BST.

Output Format

The locked stub code in your editor will print the integer returned by your *getHeight* function denoting the height of the BST.

Sample Input

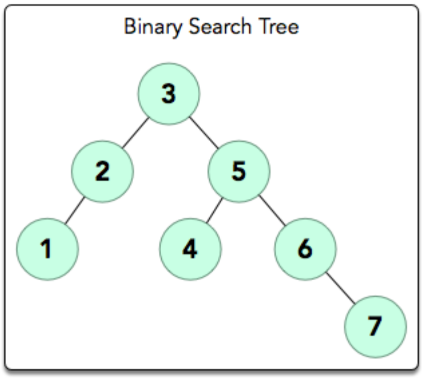
```
7
3
5
2
1
4
6
7
```

Sample Output

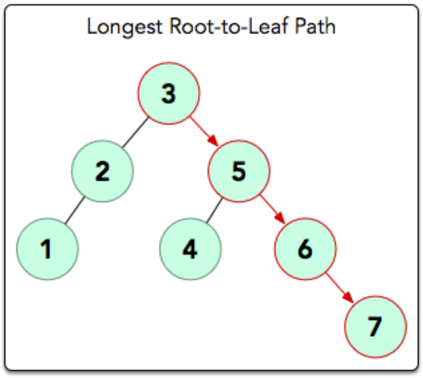
```
3
```

Explanation

The input forms the following BST:



The longest root-to-leaf path is shown below:



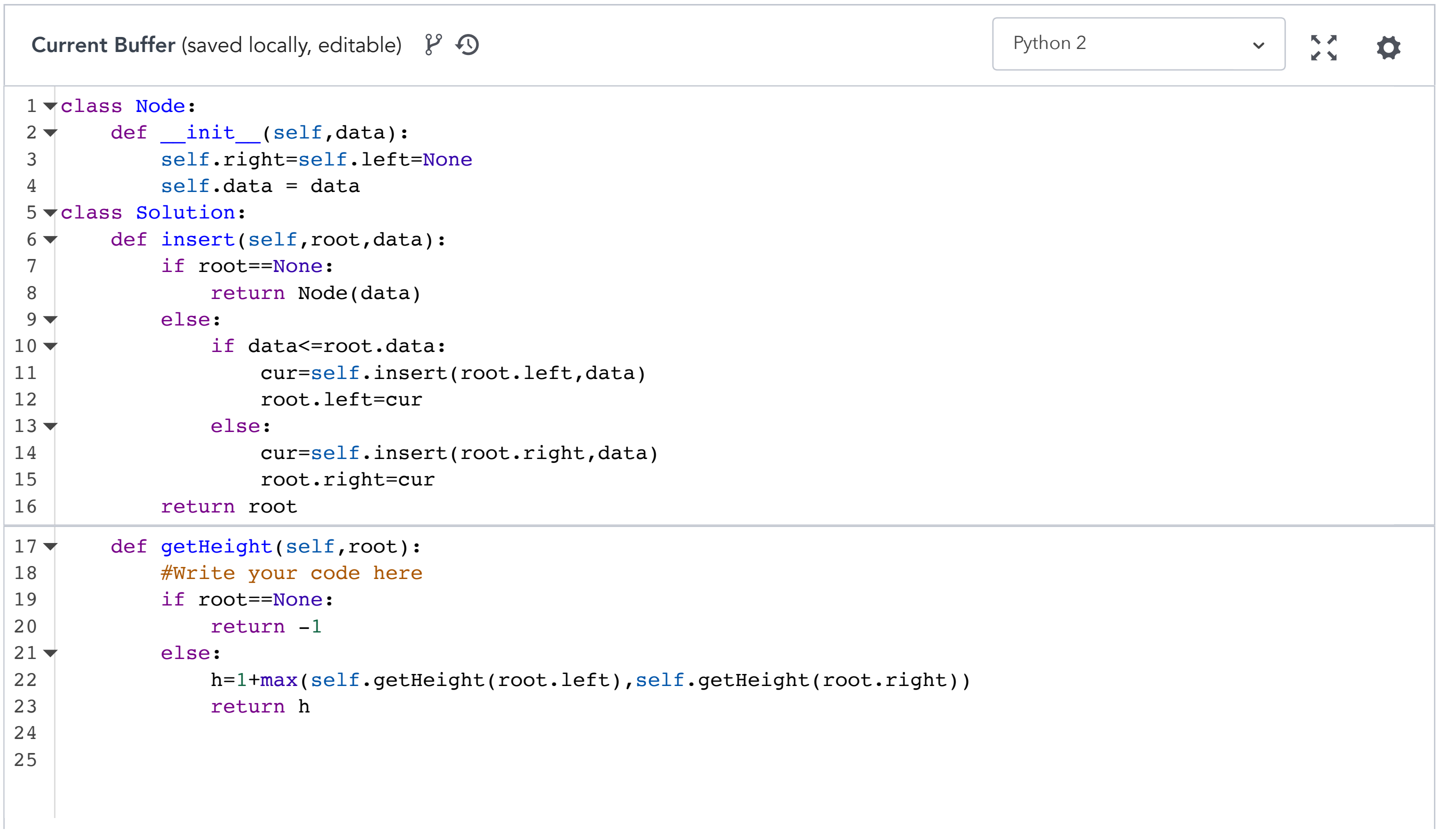
There are 4 nodes in this path that are connected by 3 edges, meaning our BST's $height = 3$. Thus, we print 3 as our answer.

Submissions: 4448

Max Score: 30

Difficulty: Easy

More



```python
class Node:
    def __init__(self,data):
        self.right=self.left=None
        self.data = data
class Solution:
    def insert(self,root,data):
        if root==None:
            return Node(data)
        else:
            if data<=root.data:
                cur=self.insert(root.left,data)
                root.left=cur
            else:
                cur=self.insert(root.right,data)
                root.right=cur
        return root

    def getHeight(self,root):
        #Write your code here
        if root==None:
            return -1
        else:
            h=1+max(self.getHeight(root.left),self.getHeight(root.right))
            return h


```

```
T=int(raw_input())
myTree=Solution()
root=None
for i in range(T):
    data=int(raw_input())
    root=myTree.insert(root,data)
height=myTree.getHeight(root)
print height
```

Line: 20 Col: 22

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

Next Challenge

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature