You have successfully solved Tree: Height of a Binary Tree Share Tweet

Try the Next Challenge | Try a Random Challenge

Tree: Height of a Binary Tree

by vatsalchanana

Problem Submissions Leaderboard Discussions Editorial

The height of a binary tree is the number of edges between the tree's root and its furthest leaf. This means that a tree containing a single node has a height of **0**.

Complete the *getHeight* function provided in your editor so that it returns the height of a binary tree. This function has a parameter, ***root***, which is a pointer to the root node of a binary tree.
Note -The Height of binary tree with single node is taken as zero.

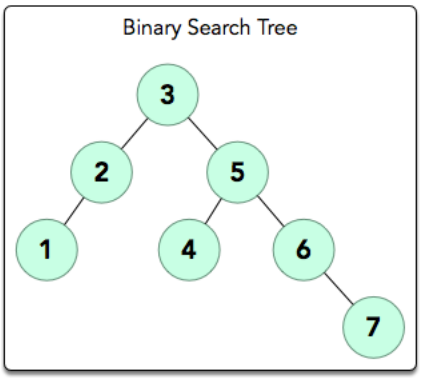
Input Format

You do not need to read any input from stdin. Our grader will pass the root node of a binary tree to your *getHeight* function.

Output Format

Your function should return a single integer denoting the height of the binary tree.

Sample Input



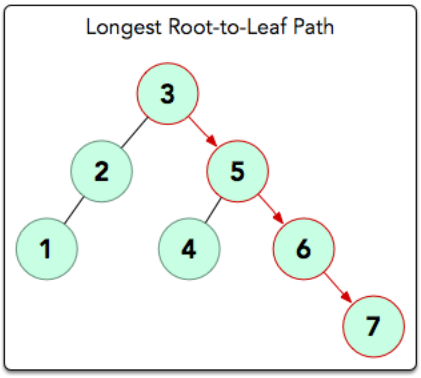
Note: A *binary search tree* is a binary tree in which the value of each parent node's left child is less than the value the parent node, and the value of the parent node is less than the value of its right child.

Sample Output

```
3
```

Explanation

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



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

Easy Submitted 72408 times
Max Score 10

Need Help?

View Discussions

View Editorial Solution

View Top Submissions

Rate This Challenge:

Download problem statement

Download sample test cases

Suggest Edits

Current Buffer (saved locally, editable) Python 3

```
class Node:↔

class BinarySearchTree:
    def __init__(self):
        self.root = None

    def create(self, val):
        if self.root == None:
            self.root = Node(val)
        else:
            current = self.root

            while True:
                if val < current.info:
                    if current.left:
                        current = current.left
                    else:
                        current.left = Node(val)
                        break
                elif val > current.info:
                    if current.right:
```

```
                        current = current.right
                    else:
                        current.right = Node(val)
                        break
                else:
                    break


# Enter your code here. Read input from STDIN. Print output to STDOUT
'''
class Node:
      def __init__(self,info):
          self.info = info
          self.left = None
          self.right = None


       // this is a node of the tree , which contains info as data, left , right
'''
def height(root):
    if root==None:
        return -1
    else:
        left=height(root.left)
        right=height(root.right)
        return max(left,right)+1


tree = BinarySearchTree()
t = int(input())

for _ in range(t):
    x = int(input())
    tree.create(x)

print(height(tree.root))

```

Line: 55 Col: 16

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

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

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

Your Output (stdout)

```
3
```

Expected Output

```
3
```

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