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.

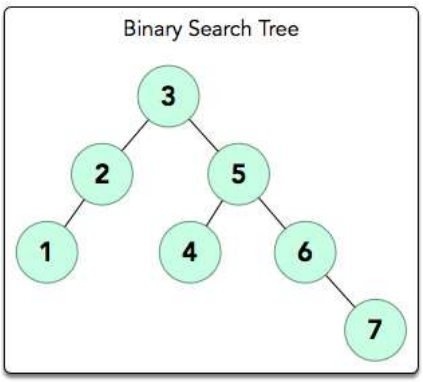
Input Format

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



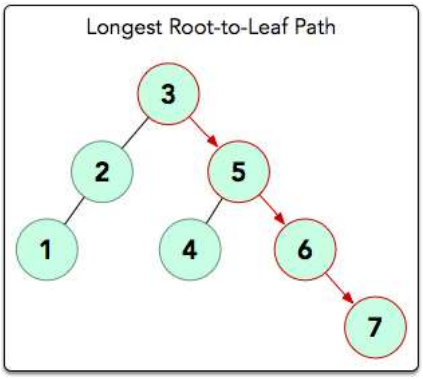
Note: A *binary search tree* is a binary tree in which the value of each parent node's left child is less than the value the parent node, and the value of the parent node is less than the value of its right child.

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.

Submissions: 43919
Max Score: 10
Difficulty: Easy

Rate This Challenge:

More

Current Buffer (saved locally, editable) Java 7

```
import java.util.*;
import java.io.*;
class Node {
    Node left;
    Node right;
    int data;

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

class Solution {
    /*
    class Node
        int data;
        Node left;
        Node right;
    */
    static int height(Node root) {
        // Write your code here.
        int htLeft = 0, htRight = 0, htMax = -1;

        if (root == null)
            return -1;
        if (root.left != null)
            htLeft = 1 + height(root.left);
        if (root.right != null)
            htRight =  1 + height(root.right);

        htMax = Math.max(htLeft, htRight);

        return htMax;
    }

    public static Node insert(Node root, int data) {
        if(root == null){
            return new Node(data);
        }
```

```
        else {
            Node cur;
            if(data <= root.data){
                cur = insert(root.left, data);
                root.left = cur;
            }
            else{
                cur = insert(root.right, data);
                root.right = cur;
            }
            return root;
        }
    }
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int t = scan.nextInt();
        Node root = null;
        while(t-- > 0){
            int data = scan.nextInt();
            root = insert(root, data);
        }
        scan.close();
        int height = height(root);
        System.out.println(height);
    }
}
```

Line: 67 Col: 2

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

Next Challenge