All Contests > SDA_HW_6 > Tree: Height of a Binary Tree

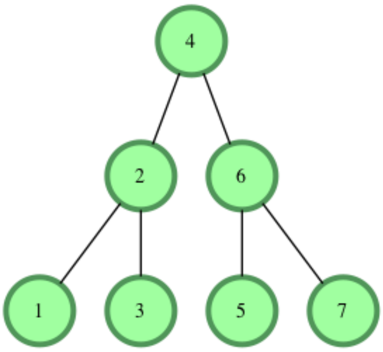
Tree: Height of a Binary Tree

locked

by vatsalchanana

Problem Submissions Leaderboard Discussions

The height of a binary tree is the number of edges between the tree's root and its furthest leaf. For example, the following binary tree is of height 2:



Function Description

Complete the *getHeight* or *height* function in the editor. It must return the height of a binary tree as an integer.

getHeight or height has the following parameter(s):

- *root*: a reference to the root of a binary tree.

Note -The Height of binary tree with single node is taken as zero.

Input Format

The first line contains an integer n, the number of nodes in the tree.
Next line contains n space separated integer where ith integer denotes node[i].data.

Note: Node values are inserted into a binary search tree before a reference to the tree's root node is passed to your function. In a binary search tree, all nodes on the left branch of a node are less than the node value. All values on the right branch are greater than the node value.

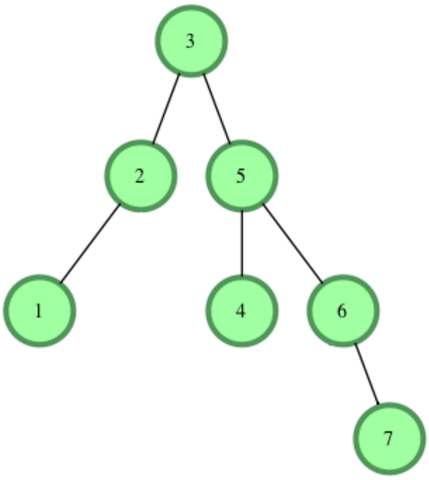
Constraints

$1 \leq node.data[i] \leq 20$
$1 \leq n \leq 20$

Output Format

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

Sample Input

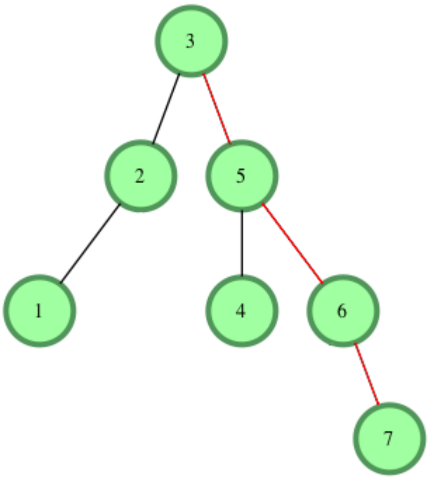


Sample Output

```
3
```

Explanation

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

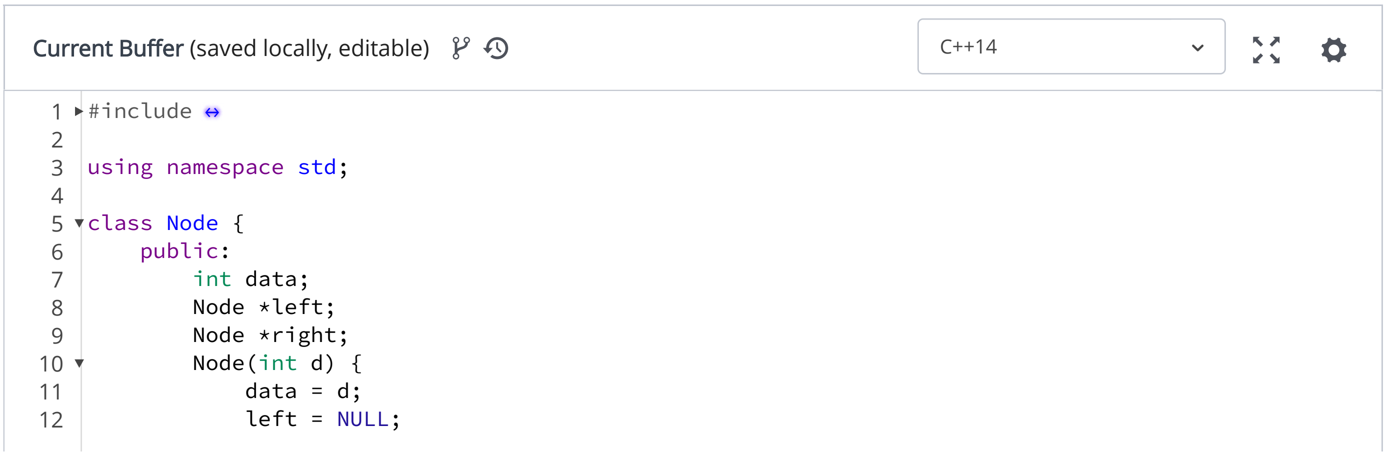


There are 4 nodes in this path that are connected by 3 edges, meaning our binary tree's $height = 3$.

Submissions: 175
Max Score: 10
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More



Current Buffer (saved locally, editable)

C++14

```
#include

using namespace std;

class Node {
    public:
        int data;
        Node *left;
        Node *right;
        Node(int d) {
            data = d;
            left = NULL;
```

```
      right = NULL;
    }
};

class Solution {
  public:
    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;
     }
    }

  int Max(int a, int b)
  {
    return a > b ? a : b;
  }
  int height(Node* root)
  {
    //if we have no root, then it is -1, but if we have at least one Node, then we should return 0, so we add 1
    if (root == nullptr)
      return -1;

    return 1 + Max(height(root->left), height(root->right));
  }

}; //End of Solution

int main() {

  Solution myTree;
  Node* root = NULL;

  int t;
  int data;

  std::cin >> t;

  while(t-- > 0) {
    std::cin >> data;
    root = myTree.insert(root, data);
  }

  int height = myTree.height(root);
  std::cout << height;

  return 0;
}

```

Line: 15 Col: 1

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

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