







662. Maximum Width of Binary Tree 2 (/problems/maximum-width-of-binary-tree/)

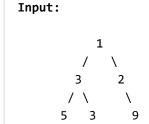
Dec. 10, 2017 | 23.3K views

Average Rating: 2.17 (58 votes)

Given a binary tree, write a function to get the maximum width of the given tree. The width of a tree is the maximum width among all levels. The binary tree has the same structure as a full binary tree, but some nodes are null.

The width of one level is defined as the length between the end-nodes (the leftmost and right most non-null nodes in the level, where the null nodes between the end-nodes are also counted into the length calculation.

Example 1:

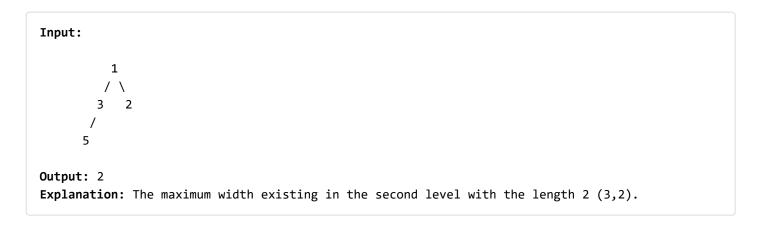


Output: 4

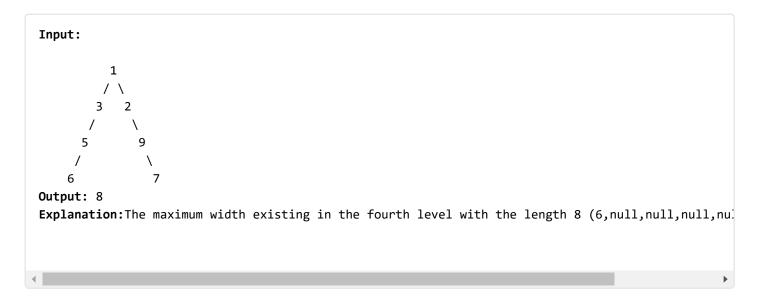
Explanation: The maximum width existing in the third level with the length 4 (5,3,null,9).

Example 2:

Example 3:



Example 4:



Note: Answer will in the range of 32-bit signed integer.

Approach Framework

Explanation

As we need to reach every node in the given tree, we will have to traverse the tree, either with a depth-first search, or with a breadth-first search.

The main idea in this question is to give each node a position value. If we go down the left neighbor, then position -> position * 2; and if we go down the right neighbor, then position -> position * 2 + 1. This makes it so that when we look at the position values L and R of two nodes with the same depth, the width will be R - L + 1.

Approach #1: Breadth-First Search [Accepted]

Intuition and Algorithm

Traverse each node in breadth-first order, keeping track of that node's position. For each depth, the first node reached is the left-most, while the last node reached is the right-most.

```
🖺 Сору
       Python
Java
    class Solution {
 1
 2
        public int widthOfBinaryTree(TreeNode root) {
 3
            Queue<AnnotatedNode> queue = new LinkedList();
 4
             queue.add(new AnnotatedNode(root, 0, 0));
 5
             int curDepth = 0, left = 0, ans = 0;
 6
            while (!queue.isEmpty()) {
 7
                 AnnotatedNode a = queue.poll();
 8
                 if (a.node != null) {
 9
                     queue.add(new AnnotatedNode(a.node.left, a.depth + 1, a.pos * 2));
                     queue.add(new AnnotatedNode(a.node.right, a.depth + 1, a.pos * 2 + 1));
10
                     if (curDepth != a.depth) {
11
12
                         curDepth = a.depth;
13
                         left = a.pos;
                     }
14
                     ans = Math.max(ans, a.pos - left + 1);
15
16
                 }
17
             }
18
            return ans;
19
        }
    }
20
21
22
    class AnnotatedNode {
23
        TreeNode node;
24
        int depth, pos;
25
        AnnotatedNode(TreeNode n, int d, int p) {
26
             node = n;
27
             depth = d:
```

Complexity Analysis

- ullet Time Complexity: O(N) where N is the number of nodes in the input tree. We traverse every node.
- Space Complexity: O(N), the size of our queue .

Approach #2: Depth-First Search [Accepted]

Intuition and Algorithm

Traverse each node in depth-first order, keeping track of that node's position. For each depth, the position of the first node reached of that depth will be kept in left[depth].

Then, for each node, a candidate width is pos - left[depth] + 1. We take the maximum of the candidate answers.

```
Copy
       Python
Java
 1
    class Solution {
 2
        int ans;
 3
        Map<Integer, Integer> left;
 4
        public int widthOfBinaryTree(TreeNode root) {
 5
             ans = 0;
 6
            left = new HashMap();
 7
             dfs(root, 0, 0);
 8
            return ans;
 9
10
        public void dfs(TreeNode root, int depth, int pos) {
             if (root == null) return;
11
            left.computeIfAbsent(depth, x-> pos);
12
             ans = Math.max(ans, pos - left.get(depth) + 1);
13
            dfs(root.left, depth + 1, 2 * pos);
14
15
             dfs(root.right, depth + 1, 2 * pos + 1);
16
        }
    }
17
```

Complexity Analysis

- ullet Time Complexity: O(N) where N is the number of nodes in the input tree. We traverse every node.
- ullet Space Complexity: O(N), the size of the implicit call stack in our DFS.

Analysis written by: @awice (https://leetcode.com/awice).

Rate this article:

Previous (/articles/strange-printer/)

Next **②** (/articles/judge-route-circle/)

Comments:

Sort By ▼

Type comment here... (Markdown is supported)

Preview

Post

dantegt (dantegt) ★ 7 ② December 11, 2017 10:05 AM

Was that really accepted? I wonder about the case when the input is a tree always going to the right. As far as I understand, your 'pos' integer variable would overflow with a depth > 32. I am asking it because it happened to me.

7 ∧ ∨ © Share ¬ Reply

SHOW 7 REPLIES

Dird (dird) ★ 29 ② February 23, 2018 1:28 AM

i

What is (depth, x-> pos)?? Why not just (depth, pos)

3 ∧ ∨ ☑ Share ¬ Reply

SHOW 2 REPLIES

habibullah (habibullah) ★ 16 ② March 27, 2019 12:03 AM

i

The solution is not correct. It will overflow for some input. Converting the solution in c++ shows the overflow, overflow test cases doesn't have any max result in overflow area so java solution doesn't shows problem.

A correct approach could be to recenter the current level of tree to avoid overflow.

Read More

2 ∧ ∨ ☑ Share ¬ Reply

SHOW 1 REPLY

shlykovich (shlykovich) ★ 14 ② July 26, 2019 10:10 AM

i

I didn't find the explanation very intuitive, so this is my attempt to make things clear.

First of all, if you have two positive integers a and b where a < b, than there are exactly (b - a + 1) integers between them including a and b. For example if a = 5, b = 10 than there are 10 - 5 + 1 = 6 integers.

Read More

```
ritakiv (ritakiv) ★ 6 ② July 17, 2019 3:22 AM
                                                                                            i
For those who struggle with an overflow exception in c++, just use ulong type for pos value.
0 ∧ ∨ © Share ¬ Reply
SHOW 1 REPLY
wanders (wanders) ★ 10 ② May 24, 2019 10:51 PM
                                                                                            i
Similar BFS python solution using just position:
def widthOfBinaryTree(self, root):
        if not root:
            return 0
                                                                                    Read More
0 ∧ ∨ © Share ¬ Reply
                                                                                            i
vlearner (vlearner) ★ 45 ② May 19, 2019 9:14 AM
nice explanation!
bxs3514 (bxs3514) ★ 5 ② April 22, 2019 2:31 AM
                                                                                            i
Reset the left index(position) to 0 for every level can avoid the overflow.
SHOW 1 REPLY
                                                                                           i
fudonglai (fudonglai) ★ 380 ② February 28, 2019 3:33 PM
Now the solutions may cause overflow, using Python can avoid it.
gogol1anonly (gogol1anonly) ★2 ② January 23, 2019 5:45 PM
                                                                                            i
Really good problem.
Love the twists!!!
< (1) (2) →
```

Copyright © 2019 LeetCode

Help Center (/support/) | Terms (/terms/) | Privacy Policy (/privacy/)

Lunited States (/region/)