Previous (/articles/strange-printer/) Next (/articles/judge-route-circle/)

662. Maximum Width of Binary Tree (/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:

```
Input:

           1
         /   \
        3     2
       / \     \
      5   3     9

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

Example 2:

```
Input:

          1
         /
        3
       / \
      5   3

Output: 2
Explanation: The maximum width existing in the third level with the length 2 (5,3).
```

Example 3:

```
Input:

          1
         / \
        3   2
       /
      5

Output: 2
Explanation: The maximum width existing in the second level with the length 2 (3,2).
```

Example 4:

```
Input:

          1
         / \
        3   2
       /     \
      5       9
     /         \
    6           7
Output: 8
Explanation:The maximum width existing in the fourth level with the length 8 (6,null,null,null,nul
```

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.

Java | Python

Copy

```
class Solution {
    public int widthOfBinaryTree(TreeNode root) {
        Queue<AnnotatedNode> queue = new LinkedList();
        queue.add(new AnnotatedNode(root, 0, 0));
        int curDepth = 0, left = 0, ans = 0;
        while (!queue.isEmpty()) {
            AnnotatedNode a = queue.poll();
            if (a.node != null) {
                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));
                if (curDepth != a.depth) {
                    curDepth = a.depth;
                    left = a.pos;
                }
                ans = Math.max(ans, a.pos - left + 1);
            }
        }
        return ans;
    }
}

class AnnotatedNode {
    TreeNode node;
    int depth, pos;
    AnnotatedNode(TreeNode n, int d, int p) {
        node = n;
        depth = d;
```

Complexity Analysis

- 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.

Java | Python

Copy

```
class Solution {
    int ans;
    Map<Integer, Integer> left;
    public int widthOfBinaryTree(TreeNode root) {
        ans = 0;
        left = new HashMap();
        dfs(root, 0, 0);
        return ans;
    }
    public void dfs(TreeNode root, int depth, int pos) {
        if (root == null) return;
        left.computeIfAbsent(depth, x-> pos);
        ans = Math.max(ans, pos - left.get(depth) + 1);
        dfs(root.left, depth + 1, 2 * pos);
        dfs(root.right, depth + 1, 2 * pos + 1);
    }
}
```

Complexity Analysis

- 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 the implicit call stack in our DFS.

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

Rate this article:

Previous (/articles/strange-printer/) Next (/articles/judge-route-circle/)

Comments: 11

Sort By

Type comment here... (Markdown is supported)

Preview Post

dantegt (dantegt) ★ 7 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 February 23, 2018 1:28 AM

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

3 Share Reply

SHOW 2 REPLIES

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

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 July 26, 2019 10:10 AM

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

0 Share Reply

ritakiv (ritakiv) ★ 6 July 17, 2019 3:22 AM

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 May 24, 2019 10:51 PM

Similar BFS python solution using just position:

```
def widthOfBinaryTree(self, root):
        if not root:
            return 0
```

Read More

0 Share Reply

vlearner (vlearner) ★ 45 May 19, 2019 9:14 AM

nice explanation!

0 Share Reply

bxs3514 (bxs3514) ★ 5 April 22, 2019 2:31 AM

Reset the left index(position) to 0 for every level can avoid the overflow.

0 Share Reply

SHOW 1 REPLY

fudonglai (fudonglai) ★ 380 February 28, 2019 3:33 PM

Now the solutions may cause overflow, using Python can avoid it.

0 Share Reply

gogol1anonly (gogol1anonly) ★ 2 January 23, 2019 5:45 PM

Really good problem.
Love the twists!!!

0 Share Reply

< 1 2 >