250. Count Univalue Subtrees

Feb. 28, 2019 | 36K views

Average Rating: 4.19 (27 votes)

Given a binary tree, count the number of uni-value subtrees.

A Uni-value subtree means all nodes of the subtree have the same value.

Example :

```
Input:  root = [5,1,5,5,5,null,5]

              5
             / \
            1   5
           / \   \
          5   5   5

Output: 4
```

Solution

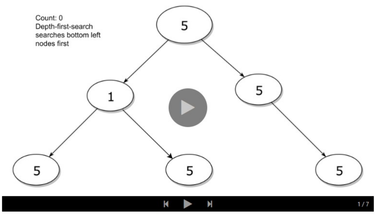
Approach 1: Depth First Search

Intuition

Given a node in our tree, we know that it is a univalue subtree if it meets one of the following criteria:

1. The node has no children (base case)
2. All of the node's children are univalue subtrees, and the node and its children all have the same value

With this in mind we can perform a depth-first-search on our tree, and test if each subtree is uni-value in a bottom-up manner.



Algorithm

Java | Python — Copy

```java
class Solution:
    def countUnivalSubtrees(self, root):
        if root is None: return 0
        self.count = 0
        self.is_uni(root)
        return self.count

    def is_uni(self, node):

        # base case - if the node has no children this is a univalue subtree
        if node.left is None and node.right is None:

            # found a univalue subtree - increment
            self.count += 1
            return True

        is_uni = True

        # check if all of the node's children are univalue subtrees and if they have the same value
        # also recursively call is_uni for children
        if node.left is not None:
            is_uni = self.is_uni(node.left) and is_uni and node.left.val == node.val

        if node.right is not None:
            is_uni = self.is_uni(node.right) and is_uni and node.right.val == node.val

        # increment self.res and return whether a univalue tree exists here
```

Complexity Analysis

- Time complexity : $O(n)$.

  Due to the algorithm's depth-first nature, the `is_uni` status of each node is computed from bottom up. When given the `is_uni` status of its children, computing the `is_uni` status of a node occurs in $O(1)$

  This gives us $O(1)$ time for each node in the tree with $O(N)$ total nodes for a time complexity of $O(N)$

- Space complexity : $O(H)$, with `H` being the height of the tree. Each recursive call of `is_uni` requires stack space. Since we fully process `is_uni(node.left)` before calling `is_uni(node.right)`, the recursive stack is bound by the longest path from the root to a leaf - in other words the height of the tree.

Approach 2: Depth First Search - Pass Parent Values

Algorithm

We can use the intuition from approach one to further simplify our algorithm. Instead of checking if a node has no children, we treat `null` values as univalue subtrees that we don't add to the count.

In this manner, if a node has a `null` child, that child is automatically considered to a valid subtree, which results in the algorithm only checking if other children are invalid.

Finally, the helper function checks if the current node is a valid subtree but returns a boolean indicating if it is a valid component for its parent. This is done by passing in the value of the parent node.

Java | Python — Copy

```java
class Solution:
    def countUnivalSubtrees(self, root):
        self.count = 0
        self.is_valid_part(root, 0)
        return self.count


    def is_valid_part(self, node, val):

        # considered a valid subtree
        if node is None: return True

        # check if node.left and node.right are univalue subtrees of value node.val
        if not all([self.is_valid_part(node.left, node.val),
                    self.is_valid_part(node.right, node.val)]):
            return False

        # if it passed the last step then this a valid subtree - increment
        self.count += 1

        # at this point we know that this node is a univalue subtree of value node.val
        # pass a boolean indicating if this is a valid subtree for the parent node
        return node.val == val
```

The above code is a commented version of the code here, originally written by Stefan Pochmann.

Complexity Analysis

- Time complexity : $O(N)$. Same as the previous approach.
- Space complexity : $O(H)$, with `H` being the height of the tree. Same as the previous approach.

Written by @alwinpeng.

Rate this article:

Previous Next

Comments: 18

Sort By

Type comment here... (Markdown is supported)

Preview Post

JAMESJJ78 ★ 207 April 10, 2019 7:47 PM

This is really a hard one for me but I'm going to persist :) You only lose if you give up

124 Share Reply

Skanda8 ★ 152 March 26, 2020 8:11 PM

Why do most of the solutions in LeetCode articles ***encourage*** use of global variables.
If you are really practicing for interview, you should not be in a habit of using global variables.

You'd rarely be encouraged in real-life software development to use global variables.

8 Share Reply

SHOW 4 REPLIES

Gypsophila ★ 71 October 7, 2019 7:05 AM

In solution 1. checking leaf node is redundant.

2 Share Reply

SHOW 1 REPLY

socialguy ★ 161 January 1, 2020 12:39 PM

```
def num_univals(node: BinaryTree[T]) -> int:
    if node is None:
        return 0
    if node.left is None and node.right is None:
```

Read More

1 Share Reply

yh32 ★ 36 November 4, 2019 11:56 PM

We can get the uni value from the child node. I think this is easier to understand for me.
if the value is null, the child is not a univalue binary tree.
if the value is not null and equals to the current node value, we increment the value and return the current node value.

Read More

1 Share Reply

powerrc ★ 13 May 31, 2019 3:21 AM

Is the time complexity really o(n)?

0 Share Reply

SHOW 1 REPLY

songdoudou ★ 3 July 2, 2020 7:50 AM

Is it actually a good practice to return "count", a variable defined outside the function, within the function?

0 Share Reply

YashKumar6640 ★ 2 June 30, 2020 11:27 PM

Why most of the leetcode solution contain global variables ? Not a good practice

0 Share Reply

MyNameIsCarrie ★ 8 June 26, 2020 9:23 PM

Why passing in 0 as a val in is_valid_part(root, 0)? ?

0 Share Reply

Mrule ★ 0 May 27, 2020 6:34 AM

c++ (clair solution)
/**

- Definition of TreeNode:
- class TreeNode {

Read More

0 Share Reply