Daily Coding Problem #284

Problem

This problem was asked by Yext.

Two nodes in a binary tree can be called cousins if they are on the same level of the tree but have different parents. For example, in the following diagram 4 and 6 are cousins.

```
    1
   / \
  2   3
 / \   \
4   5   6
```

Given a binary tree and a particular node, find all cousins of that node.

Solution

We can solve this in two steps.

First, we must find out what level of the tree our node resides on, which can be done recursively. If we define the root node as the first row, then we can search the left and right children of each node, starting with the root, incrementing the level as we step down. As soon as we find our input node, we return the current level value.

```
class Node:
    def __init__(self, data, left=None, right=None):
        self.data = data
        self.left = left
```

```
        self.right = right

def get_level(root, node, level=1):
    if not root:
        return 0
    if root == node:
        return level

    return get_level(root.left, node, level + 1) or \
           get_level(root.right, node, level + 1)
```

Next, we must find all nodes in our tree at this level, that are not also siblings of the input node. Let's think through some cases to figure out how we can accomplish this.

First, if we are looking at the root or second level of the tree, it is impossible for there to be any cousins. The first interesting situation, then, occurs when the input node is on the third row. To find cousins in this case, we must consider the nodes on the row above. For each of these nodes, as long as it does not have our input node as one of its children, we should add its children to the solution list.

Finally, if we are searching for cousins on a row greater than three, we can recursively move downward in the tree while decrementing the level, until our root is at level two and we can apply the logic above.

```
def get_cousins(root, node, level):
    if level <= 1:
        return []

    cousins = []
    if level == 2:
        if root.left != node and root.right != node:
            if root.left:
                cousins.append(root.left.data)
            if root.right:
                cousins.append(root.right.data)

    else:
        cousins += get_cousins(root.left, node, level - 1)
        cousins += get_cousins(root.right, node, level - 1)

    return cousins
```

Our main function, then, would look like this:

```
def find_cousins(root, node):
    level = get_level(root, node)

    return get_cousins(root, node, level)
```

Both of these functions require us to traverse each node of the tree in the worst case, so the time complexity of this solution will be O(N).

© Daily Coding Problem 2019

Privacy Policy

Terms of Service

Press