Daily Coding Problem #80

Problem

This problem was asked by Google.

Given the root of a binary tree, return a deepest node. For example, in the following tree, return d.

```
    a
   / \
  b   c
 /
d
```

Solution

Base case for this question actually can't be null, because it's not a real result that can be combined (null is not a node).

Here we should use the leaf node as the base case and return itself.

The recursive step for this problem is a little bit tricky because we can't actually use the results of the left and right subtrees directly. So we need to ask, what other information do we need to solve this question? It turns out if we tagged with each subresult node their depths, we could get the final solution by picking the higher depth leaf and then incrementing it:

```
def deepest(node):
    if node and not node.left and not node.right:
        return (node, 1) # Leaf and its depth

    if not node.left: # Then the deepest node is on the right subtree
        return increment_depth(deepest(node.right))
    elif not node.right: # Then the deepest node is on the left subtree
        return increment_depth(deepest(node.left))

    return increment_depth(
            max(deepest(node.left), deepest(node.right),
                    key=lambda x: x[1])) # Pick higher depth tuple and then increment its depth

def increment_depth(node_depth_tuple):
    node, depth = node_depth_tuple
    return (node, depth + 1)
```

Terms of Service

Press