Daily Coding Problem #160

Problem

This problem was asked by Uber.

Given a tree where each edge has a weight, compute the length of the longest path in the tree.

For example, given the following tree:

```
   a
  /|\
 b c d
    / \
   e   f
  / \
 g   h
```

and the weights: `a-b: 3, a-c: 5, a-d: 8, d-e: 2, d-f: 4, e-g: 1, e-h: 1`, the longest path would be `c -> a -> d -> f`, with a length of 17.

The path does not have to pass through the root, and each node can have any amount of children.

Solution

There are two cases: either the longest path goes through the root or it doesn't.

If it doesn't, then we only need to look at the longest path of any of the current node's children. If it does, then the longest path must come from the paths made by combining the two highest childrens' heights.

What we'll do is recursively look at each child's height and longest path, and keep track of the longest path we've seen so far. At the end we select the largest between the longest subpath, or the two largest heights that can be combined to make a new larger path.

The base case here is when we look at a node with no children, in which case the longest path should be 0.

```
from math import inf


class Node:
    def __init__(self, val, children=[]):
        self.val = val
        self.children = children


def longest_path(root):
    height, path = longest_height_and_path(root)
    return path


def longest_height_and_path(root):
    longest_path_so_far = -inf
    highest, second_highest = 0, 0
    for length, child in root.children:
```

```
        height, longest_path_length = longest_height_and_path(child)

        longest_path_so_far = max(longest_path_so_far, longest_path_length)

        if height + length > highest:
            highest, second_highest = height + length, highest
        elif height + length > second_highest:
            second_highest = height + length


    return highest, max(longest_path_so_far, highest + second_highest)
```

Since we're looking at every node recursively, this algorithm runs in O(n) time and O(h) space.
