Daily Coding Problem #296

Problem

This problem was asked by Etsy.

Given a sorted array, convert it into a height-balanced binary search tree.

Solution

If the tree did not have to be balanced, we could initialize the first element as the root of the tree, and add each subsequent element as a right child.

However, as the problem is written, adding each element in list order would require us to rotate the tree several times, in order to change the root value, making our algorithm inefficient.

Instead, since the list is sorted, we know that the root should be the element in the middle of the list, which we can call `M`. Furthermore, the left subtree will be equivalent to a balanced binary search tree created from the first `M - 1` elements in the list. Analogously, the right subtree can be constructed from the elements after `M` in our input.

Therefore, we can create this tree recursively by calling our function on successively smaller input ranges to create each subtree.

```
class Node:
```

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

def make_bst(array):
    if not array:
        return None

    mid = len(array) // 2

    root = Node(array[mid])

    root.left = make_bst(array[:mid])
    root.right = make_bst(array[mid + 1:])

    return root
```

This algorithm will take `O(N)` time and space, since for each element of the list, we must construct a node and add it as a child to the tree, each of which can be considered `O(1)` operations.