Daily Coding Problem #179

Problem

This problem was asked by Google.

Given the sequence of keys visited by a postorder traversal of a binary search tree, reconstruct the tree.

For example, given the sequence `2, 4, 3, 8, 7, 5`, you should construct the following tree:

```
    5
   / \
  3   7
 / \   \
2   4   8
```

Solution

Recall the algorithm for a postorder traversal:

- Traverse the left subtree recursively
- Traverse the right subtree recursively
- Get the current node's value

The basic idea here is to recursively build the left subtree, right subtree, and then attach them to the root, returning the root.

We know the root is always going to be the last element in the sequence since it's the last step in the recursive algorithm, and we also know that the left subtree's traversal will be before the right subtree's traversal. But where exactly?

Since this is a binary search tree, all the nodes in the right subtree are greater than the the root,

and all the nodes in the left subtree are smaller than the root. And since the left subtree is traversed first, we can iterate through the sequence up to the last value and stop at the first index that has a value greater than `root_val`: all values up to here belong to the left subtree, and all values after here (up until the last element) belong to the right subtree.

```
class BSTNode:
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def reconstruct(postorder):
    if not postorder:
        return None
    elif len(postorder) == 1:
        return BSTNode(postorder[0])

    root_val = postorder[-1]
    root = BSTNode(root_val)

    for i, val in enumerate(postorder[:-1]):
        if val > root_val:

            left_subtree = reconstruct(postorder[:i])
            right_subtree = reconstruct(postorder[i:-1])
            root.left = left_subtree
            root.right = right_subtree
            return root

    left_subtree = reconstruct(postorder[:-1])
    root.left = left_subtree
    return root
```