129. Sum Root to Leaf Numbers

There are two intuitive solutions, one is child sending its height info back to parent, and another is parent passing its depth info down to its children. The former is hard because we must maintain all children's height info. So I pick the second option.

Sol:
Assume R is root. One of its root to leaf number is $\overline{Rc_1c_2, \ldots, c_n}$. Now after it's split into two part, $\overline{R}$ and $\overline{c_1c_2, \ldots, c_n}$, a recursion will be shown up straightforwardly. That means for each root to leaf number($\overline{c_1c_2, \ldots, c_n}$) from a child, after it appended to its prefix, its parent's value($\overline{R}$), it will become one of its parent's root to leaf number($\overline{Rc_1c_2, \ldots, c_n}$).

So that's a simple recursion problem:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Solution {

    public int sumNumbers(TreeNode root) {
        if (root == null) {
            return 0;
        } else {
            return calculate(root, 0);
        }
    }

    private int calculate(TreeNode root, int parentValue) {
        if (root.left == null && root.right == null) {
            return parentValue * 10 + root.val;
        } else if(root.left == null) {
```

```
            return calculate(root.right, parentValue * 10 + root.val);
        } else if(root.right == null) {
            return calculate(root.left, parentValue * 10 + root.val);
        } else {
            return calculate(root.right, parentValue * 10 + root.val)
                + calculate(root.left, parentValue * 10 + root.val);
        }
    }
}
```