Sum Root to Leaf Numbers

Given a binary tree containing digits from `0-9` only, each root-to-leaf path could represent a number.

An example is the root-to-leaf path `1->2->3` which represents the number `123`.

Find the total sum of all root-to-leaf numbers.

For example,

```
    1
   / \
  2   3
```

The root-to-leaf path `1->2` represents the number `12`.
The root-to-leaf path `1->3` represents the number `13`.

Return the sum = 12 + 13 = `25`.

Solution 1

```
/**
 * Definition for binary tree
 * 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;
        return sumR(root, 0);
    }
    public int sumR(TreeNode root, int x) {
        if (root.right == null && root.left == null)
            return 10 * x + root.val;
        int val = 0;
        if (root.left != null)
            val += sumR(root.left, 10 * x + root.val);
        if (root.right != null)
            val += sumR(root.right, 10 * x + root.val);
        return val;
    }
}
```

written by potpie original link here

Solution 2

I use recursive solution to solve the problem.

```
public int sumNumbers(TreeNode root) {
    return sum(root, 0);
}

public int sum(TreeNode n, int s){
    if (n == null) return 0;
    if (n.right == null && n.left == null) return s*10 + n.val;
    return sum(n.left, s*10 + n.val) + sum(n.right, s*10 + n.val);
}
```

written by pavel-shlyk original link here

Solution 3

The idea is to do a preorder traversal of the tree. In the preorder traversal, keep track of the value calculated till the current node, let this value be val. For every node, we update the val as val*10 plus node's data.

```
/**
 * Definition for binary tree
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    int sumNumbers(TreeNode *root) {
       return  sumNumberUtil(root,0);

    }
    // preorder
    int sumNumberUtil(struct TreeNode* node, int val)
    {
        if(node==NULL)
        return 0;

        val= val*10+node->val;
        if(node->left==NULL && node->right==NULL)
        {
            return val;
        }

        return sumNumberUtil(node->left,val)+sumNumberUtil(node->right, val);
    }
};
```

written by Deepalaxmi original link here