129. Sum Root to Leaf Numbers

题目描述：https://leetcode.com/problems/sum-root-to-leaf-numbers/

> 从root到叶子拼成数字，求所有路径拼成数字之和。
> 例如：
>
> ```
>     1
>    / \
>   2   3
>
> 12 + 13 = 25
> ```

解题思路：

dfs

代码 递归：

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    void dfs(TreeNode* root, vector<int> &res, int num) {
        if(root == NULL) return;
        num = num * 10 + root->val;
        if(root->left == NULL && root->right == NULL) {
            res.push_back(num);
            return;
        }
        if(root->left != NULL) {
            dfs(root->left, res, num);
        }
        if(root->right != NULL) {
            dfs(root->right, res, num);
        }
        return;
    }
    int sumNumbers(TreeNode* root) {
        vector<int> res;
        dfs(root, res, 0);
        int c = 0;
        for(int i = 0; i < res.size(); i++) {
            c += res[i];
        }
        return c;
    }
};
```

代码 **postOrder:**

```
class Solution {
public:
    int sumNumbers(TreeNode* root) {
        if(!root) return 0;
        stack<TreeNode* > s;
        int sum = 0;
        int curSum = 0;
        TreeNode *pre = NULL;
        TreeNode *p = root;
        while(p || !s.empty()) {
            if (p) {
                s.push(p);
                curSum *= 10;
                curSum += p->val;
                p = p->left;
            } else {
                TreeNode* top = s.top();
                if(top->right == NULL || top->right == pre) {
                    s.pop();
                    pre = top;
                    if(top->left == NULL && top->right == NULL) {
                        sum += curSum;
                    }
                    curSum = curSum / 10;
                } else {
                    p = top->right;
                }
            }
        }
        return sum;
    }
};
```

代码 存储层次迭代：

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    int sumNumbers(TreeNode* root) {
        int res = 0;
        if(root == NULL) return res;
        stack<pair<TreeNode*, int> > s;
        s.push(pair<TreeNode*, int>(root, 0));
        while(!s.empty()) {
            TreeNode* p = s.top().first;
            int val = s.top().second * 10 + p->val;
            s.pop();
            if(p->left == NULL && p->right == NULL) {
                res += val;
            }
            if(p->left != NULL) {
                s.push(pair<TreeNode*, int> (p->left, val));
            }
            if(p->right != NULL) {
                s.push(pair<TreeNode*, int> (p->right, val));
            }
        }
        return res;
    }
};
```