Sum Root of Leaf Numbers

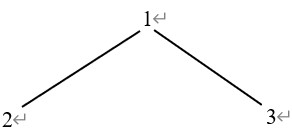
*Description*

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.

*Example 1:*

Input: [1, 2, 3]



Output: 25

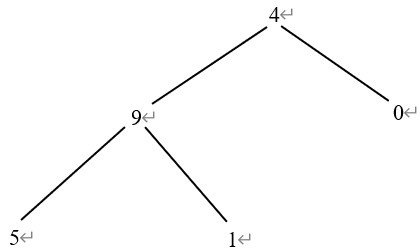
Explanation:

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

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

Therefore, sum = 12 + 13 = 25.

*Example 2:*



Output: 1026

Explanation:

The root-to-leaf path 4->9->5 represents the number 495.

The root-to-leaf path 4->9->1 represents the number 491.

The root-to-leaf path 4->0 represents the number 40.

Therefore, sum = 495 + 491 + 40 = 986 + 40 = 1026.

*Code (Non – Recursive):*

struct TreeNode {

int val;

TreeNode \*left;

TreeNode \*right;

TreeNode(int x) : val(x), left(NULL), right(NULL) {}

};

class Solution {

public:

int sumNumbers(TreeNode\* root) {

if (root == NULL)

return 0;

int sum = 0;

stack<TreeNode \*> node\_stack;

TreeNode \* node = root;

while (node || !node\_stack.empty()) {

if (node) {

node\_stack.push(node);

if (node->left) {

node->left->val += node->val \* 10;

}

node = node->left;

} else {

node = node\_stack.top();

node\_stack.pop();

if (node->right) {

node->right->val += node->val \* 10;

}

if (node->left == NULL && node->right == NULL) {

sum += node->val;

}

node = node->right;

}

}

return sum;

}

};

*Code (Recursive):*

struct TreeNode {

int val;

TreeNode \*left;

TreeNode \*right;

TreeNode(int x) : val(x), left(NULL), right(NULL) {}

};

class Solution {

public:

int getdfs(TreeNode \* root) {

int sum = dfs(root, 0);

return sum;

}

private:

int dfs(TreeNode \* root, int sum) {

if (root == NULL) return 0;

if (root->left == NULL && root->right == NULL)

sum += root->value;

return dfs(root->left, sum \* 10 + root->val) + dfs(root->right, sum \* 10 + root->val);

}

}