257 Binary Tree Paths

2018年4月7日 16:39

Question:

Given a binary tree, return all root-to-leaf paths.
For example, given the following binary tree:

```
  1
/ \
2   3
\
 5
```

All root-to-leaf paths are:
["1->2->5", "1->3"]

来自 <https://leetcode.com/problems/binary-tree-paths/description/>

给定一个二叉树，返回从根节点到叶节点的所有路径。

Solution for Python3:

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution:
    def binaryTreePaths(self, root):
        """
        :type root: TreeNode
        :rtype: List[str]
        """
        if not root:
           return []
        List = []
        self.dfs(root, '', List)
        return List

    def dfs(self, root, L, List):
       L += str(root.val)
       if not root.left and not root.right:
           List.append(L)
           return;
       L += '->'
       if root.left:
           self.dfs(root.left, L, List)
       if root.right:
           self.dfs(root.right, L, List)

class Solution2:
    def binaryTreePaths(self, root):
        """
        :type root: TreeNode
        :rtype: List[str]
        """
        if not root:
           return []
        if not root.left and not root.right:
           return [str(root.val)]
        return [str(root.val) + '->' + i for i in self.binaryTreePaths(root.left)] + [str(root.val) + '->' + i for i in self.binaryTreePaths(root.right)]
```

Solution for C++:

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution1 {
public:
    vector<string> binaryTreePaths(TreeNode* root) {
        if (!root) {
            return vector<string> {};
        }
        vector<string> v;
        dfs(root, "", v);
        return v;
    }

    void dfs (TreeNode* root, string s, vector<string>& v) {
        s += to_string(root->val);
        if (!root->left && ! root->right) {
            v.push_back(s);
            return;
        }
        s += "->";
        if (root->left) {
            dfs(root->left, s, v);
        }
        if (root->right) {
            dfs(root->right, s, v);
        }
    }
};
```

```cpp


class Solution2 {
public:
    vector<string> binaryTreePaths(TreeNode* root) {
        if (!root) {
            return vector<string> {};
        }
        if (!root->left && !root->right) {
            return vector<string> {to_string(root->val)};
        }
        vector<string> v;
        for (string s : binaryTreePaths(root->left)) {
            v.push_back(to_string(root->val) + "->" + s);
        }
        for (string s : binaryTreePaths(root->right)) {
            v.push_back(to_string(root->val) + "->" + s);
        }
        return v;
    }
};
```

Appendix:

C++ 最后一个递归解法

1) 如果是空节点，则返回空数组；
2) 如果是叶子节点，则将当前节点值变成字符串放进一个新的数组返回；
3) 当前节点下先创建空数组，然后遍历左节点返回的数组，该数组包含左子节点一下所有含有叶子节点的子串。从数组中循环取出该子串，并加上当前节点的数值字符串组成新子串放进数组。右子节点也如此。最后返回该数组。