606 Construct String from Binary Tree

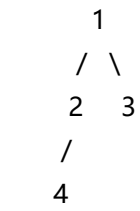
2018年4月16日 20:01

You need to construct a string consists of parenthesis and integers from a binary tree with the preorder traversing way.

The null node needs to be represented by empty parenthesis pair "()". And you need to omit all the empty parenthesis pairs that don't affect the one-to-one mapping relationship between the string and the original binary tree.

Example 1:

Input: Binary tree: [1,2,3,4]

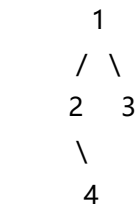


Output: "1(2(4))(3)"

Explanation: Originallay it needs to be "1(2(4())(3()())",
but you need to omit all the unnecessary empty parenthesis pairs.
And it will be "1(2(4))(3)".

Example 2:

Input: Binary tree: [1,2,3,null,4]



Output: "1(2()(4))(3)"

Explanation: Almost the same as the first example,
except we can't omit the first parenthesis pair to break the one-to-one mapping relationship between the input and the output.

来自 <https://leetcode.com/problems/construct-string-from-binary-tree/description/>

你需要采用前序遍历的方式，将一个二叉树转换成一个由括号和整数组成的字符串。

空节点则用一对空括号 "()" 表示。而且你需要省略所有不影响字符串与原始二叉树之间的一对一映射关系的空括号对。

Solution for Python3:

```
class Solution1:
    def tree2str(self, t):
```

```
            """
            :type t: TreeNode
            :rtype: str
            """
            if not t:
                return ''
            if not t.left and not t.right:
                return str(t.val)
            if not t.right:
                return str(t.val) + '(' + self.tree2str(t.left) + ')'
            return str(t.val) + '(' + self.tree2str(t.left) + ')(' +  self.tree2str(t.right) + ')'

class Solution2:
    def tree2str(self, t):
            """
            :type t: TreeNode
            :rtype: str
            """
            if not t:
                return ''
            deq = collections.deque([t])
            visited = set()
            res = ''
            while deq:
                t = deq[-1]
                if t in visited:
                    deq.pop()
                    res += ')'
                else:
                    visited.add(t)
                    res += '(' + str(t.val)
                    if not t.left and t.right:
                        res += '()'
                    if t.right:
                        deq.append(t.right)
                    if t.left:
                        deq.append(t.left)
            return res[1:-1]


```

```
22
23
    /**
1    * Definition for a binary tree node.
     * struct TreeNode {
2    *     int val;
     *     TreeNode *left;
3    *     TreeNode *right;
     *     TreeNode(int x) : val(x), left(NULL), right(NULL)
4    {}
     * };
5    */
    class Solution1 {
6    public:
        string tree2str(TreeNode* t) {
7           if (!t)
                return "";
8           if (!t->left && !t->right)
                return to_string(t->val);
9           if (!t->right)
                return to_string(t->val) + "(" +
10   tree2str(t->left) + ")";
            return to_string(t->val) + "(" + tree2str(t->
11   left) + ")(" + tree2str(t->right) + ")";
        }
12   };

13   class Solution2 {
     public:
14      string tree2str(TreeNode* t) {
            if (!t)
15              return "";
            stack<TreeNode*> sta;
16          sta.push(t);
            set<TreeNode*> visited;
17          string res;
            while (!sta.empty()) {
18              t = sta.top();
                if (visited.count(t)) {
```

```
                    sta.pop();
                    res += ")";
                } else {
                    visited.insert(t);
                    res += "(" + to_string(t->val);
                    if (!t->left && t->right)
                        res += "()";
                    if (t->right)
                        sta.push(t->right);
                    if (t->left)
                        sta.push(t->left);
                }
            }
            return res.substr(1, res.length() - 2);
        }
    };
```

40

41

42

43

44

45

46

47

48

49

50