根据二叉树创建字符串

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。

示例 1:

输入：二叉树: [1,2,3,4]

```
       1
     /   \
    2     3
   /
  4
```

输出："1(2(4))(3)"

解释：原本将是“1(2(4())(3())”，

在你省略所有不必要的空括号对之后，

它将是“1(2(4))(3)”。

示例 2:

输入：二叉树: [1,2,3,null,4]

```
       1
     /   \
    2     3
     \
      4
```

输出："1(2()(4))(3)"

解释：和第一个示例相似，

除了我们不能省略第一个对括号来中断输入和输出之间的一对一映射关系。

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left
), right(right) {}
 * };
 */
class Solution {
public:
    void travel(TreeNode* root,string& str)
    {
        if(root)
        {
            //转化为字符串  拼接当前节点数据
            stringstream ss;
            ss<<root->val;
            str+=ss.str();
        }
        //处理左子树
        if(root->left)
        {
            str+='(';
            travel(root->left,str);
            str+=')';
        }
        else
        {
            if(root->right)
            {
                str+="()";
            }
            else
                return;
        }
        //处理右子树
        if(root->right)
```

```
        {
            str+='(';
            travel(root->right,str);
            str+=')';
        }
    }
    string tree2str(TreeNode* root) {
        string str="";
        travel(root,str);
        return str;
    }
};
```