剑指Offer（四）：重建二叉树

2017年11月22日 12:29:58 2 9,522 °C 编辑

一、前言

本系列文章为《剑指Offer》刷题笔记。

刷题平台：牛客网

书籍下载：共享资源

二、题目

输入某二叉树的前序遍历和中序遍历的结果，请重建出该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。例如输入前序遍历序列{1,2,4,7,3,5,6,8}和中序遍历序列{4,7,2,1,5,3,8,6}，则重建二叉树并返回。

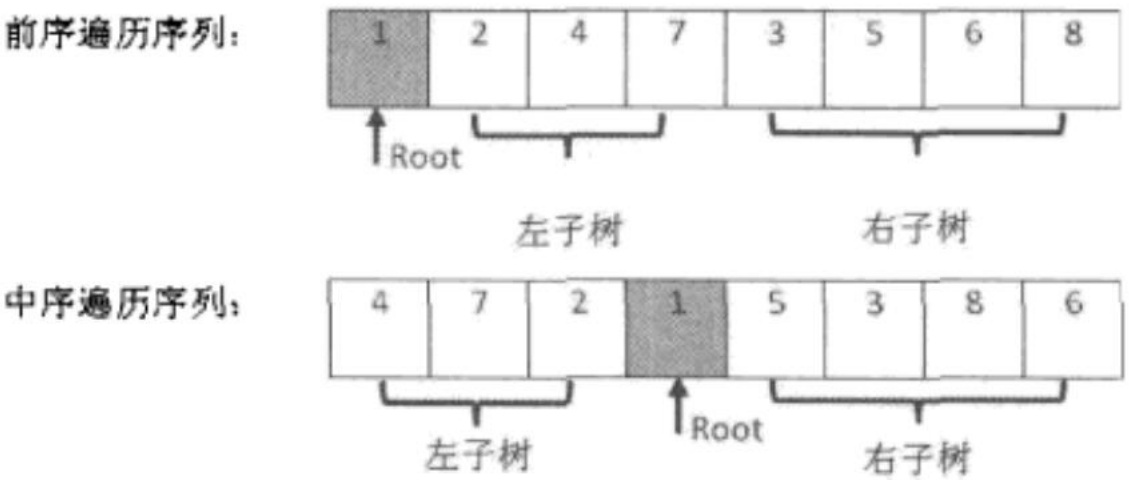
1、思路

通常树有如下几种遍历方式：

- 前序遍历：先访问根结点，再访问左子结点，最后访问右子结点。
- 中序遍历：先访问左子结点，再访问根结点，最后访问右子结点。
- 后序遍历：先访问左子结点，再访问右子结点，最后访问根结点。

本题为前序遍历和中序遍历，最少需要两种遍历方式，才能重建二叉树。

前序遍历序列中，第一个数字总是树的根结点的值。在中序遍历序列中，根结点的值在序列的中间，左子树的结点的值位于根结点的值的左边，而右子树的结点的值位于根结点的值的右边。剩下的我们可以递归来实现，具体如图：



2、代码

C++:

```
C+
```

```
/**
 * Definition for binary tree
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    TreeNode* reConstructBinaryTree(vector<int> pre,vector<int> vin) {
        if(pre.size() == 0){                    //如果为空，返回NULL
            return NULL;
        }
        //依次是前序遍历左子树，前序遍历右子树，中序遍历左子树，中序遍历右子树
        vector<int> left_pre, right_pre, left_vin, right_vin;
        //前序遍历第一个节点一定为根节点
        TreeNode* head = new TreeNode(pre[0]);
        //找到中序遍历的根节点
        int root = 0;
        //遍历找到中序遍历根节点索引值
        for(int i = 0; i < pre.size(); i++){
            if(pre[0] == vin[i]){
                root = i;
                break;
            }
        }
           //利用中序遍历的根节点，对二叉树节点进行归并
        for(int i = 0; i < root; i++){
            left_vin.push_back(vin[i]);
            left_pre.push_back(pre[i + 1]);             //前序遍历第一个为根节点
        }

        for(int i = root + 1; i < pre.size(); i++){
            right_vin.push_back(vin[i]);
            right_pre.push_back(pre[i]);
        }

        //递归，再对其进行上述所有步骤，即再区分子树的左、右子子数，直到叶节点
        head->left = reConstructBinaryTree(left_pre, left_vin);
        head->right = reConstructBinaryTree(right_pre, right_vin);
        return head;
    }
};
```

Python2.7：

```
Pytho
# -*- coding:utf-8 -*-
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
class Solution:
    # 返回构造的TreeNode根节点
    def reConstructBinaryTree(self, pre, tin):
        # write code here
        if len(pre) == 0:
            return None
        elif len(pre) == 1:
            return TreeNode(pre[0])
        else:
            root = TreeNode(pre[0])
            pos = tin.index(pre[0])
            root.left = self.reConstructBinaryTree(pre[1:pos+1], tin[:pos])
            root.right = self.reConstructBinaryTree(pre[pos+1:], tin[pos+1:])
        return root
```