重建二叉树

# 剑指Offer—重建二叉树(根据前序和中序重建二叉树)

## 题目描述

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。

示例：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根节点。



## 思路分析

首先剖析前序遍历和中序遍历的特点：

### **前序遍历**：第一个元素肯定是根元素；后面的元素分成两部分，前面连续的一部分为当前根节点的左子树的前序遍历，后面连续的一部分为当前根节点的右子树的前序遍历。如2,4,7为1节点的左子树的前序遍历；3,5,6,8为1节点的右子树的前序遍历。

### **中序遍历**：根节点元素肯定在中间位置，根节点之前的就是当前根节点的左子树的中序遍历，根节点之后的就是当前根节点的右子树的中序遍历，因此只需要找出根节点，就可以分成两部分，进而递归下去。如1节点为根节点，则4,7,2为左子树的中序遍历，5,3,8,6为右子树的中序遍历。

总体思路：前序遍历的第一个元素为根节点，根据该元素在中序遍历中找出根节点的位置，然后递归的方式寻找当前节点的左子树和右子树。

## Java代码

/\*\*

\* 递归调用该方法

\*/

public TreeNode reConstructBinaryTree(int [] pre,int [] in) {

if(pre == null||in == null) return null;//防止pre或in为null

return reConstructBinaryTree(pre,0,pre.length-1,in,0,in.length-1);

}

/\*\*

\* 递归方法

\*/

public TreeNode **reConstructBinaryTree**(int[] pre,int preStart,int preEnd,int[] in,int inStart,int inEnd){

//递归终止条件

if(preStart> preEnd||inStart > inEnd || preEnd-preStart != inEnd-inStart) return null;

TreeNode root = new TreeNode(pre[preStart]);

int rootIndex = inStart;

//在中序遍历结果中寻找当前子树的根节点索引

while(rootIndex <= inEnd && in[rootIndex] != pre[preStart]) rootIndex++;

//递归调用

**root.left** = reConstructBinaryTree(pre,preStart+1,preStart+(rootIndex-inStart),in,inStart,rootIndex-1);

**root.right** = reConstructBinaryTree(pre,preStart+(rootIndex-inStart)+1,preEnd,in,rootIndex + 1,inEnd);

return root;

}