题目 重建二叉树

考点 树 热点指数 84163 通过率 22.82%

具体题目

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。

```
import java.util.*;
/**

 * Definition for binary tree
 * public class TreeNode {
 * int val;
 * TreeNode left;
 * TreeNode right;
 * TreeNode(int x) { val = x; }
 * }
    */
    public class Solution {
    public TreeNode reConstructBinaryTree(int [] pre,int [] in) {
       if(pre.length == 0||in.length == 0){
            return null;
        }
        TreeNode node = new TreeNode(pre[0]);
        for(int i = 0; i < in.length; i++){
            if(pre[0] == in[i]){
                node.left = reConstructBinaryTree(Arrays.copyOfRange(pre, 1, i+1),
Arrays.copyOfRange(in, 0, i));
                node.right = reConstructBinaryTree(Arrays.copyOfRange(pre, i+1, pre.length),
Arrays.copyOfRange(in, i+1,in.length));
            }
        }
        return node;
    }
```

1.先求出根节点（前序序列第一个元素）。
2.将根节点带入到中序遍历序列中求出左右子树的中序遍历序列。
3.通过左右子树的中序序列元素集合带入前序遍历序列可以求出左右子树的前序序列。
4.左右子树的前序序列第一个元素分别是根节点的左右儿子
5.求出了左右子树的4种序列可以递归上述步骤

```
public class Solution {
    public TreeNode reConstructBinaryTree(int [] pre,int [] in) {
        return reConBTree(pre,0,pre.length-1,in,0,in.length-1);
    }
    public TreeNode reConBTree(int [] pre,int preleft,int preright,int [] in,int inleft,int
inright){
        if(preleft > preright || inleft> inright)//当到达边界条件时候返回null
            return null;
        //新建一个TreeNode
        TreeNode root = new TreeNode(pre[preleft]);
        //对中序数组进行输入边界的遍历
        for(int i = inleft; i<= inright; i++){
            if(pre[preleft] == in[i]){
                //重构左子树，注意边界条件
                root.left = reConBTree(pre,preleft+1,preleft+i-inleft,in,inleft,i-1);
                //重构右子树，注意边界条件
```

```
                root.right = reConBTree(pre,preleft+i+1-inleft,preright,in,i+1,inright);
            }
        }
        return root;
    }
}
```

```
public class Solution {
    public TreeNode reConstructBinaryTree(int [] pre,int [] in) {
        int i=0;
        if(pre.length!=in.length||pre.length==0||in.length==0)
            return null;
        TreeNode root = new TreeNode(pre[0]);
        while(in[i]!=root.val)
            i++;
        int[] preLeft = new int[i];
        int[] inLeft = new int[i];
        int[] preRight = new int[pre.length-i-1];
        int[] inRight = new int[in.length-i-1];
        for(int j = 0;j<in.length;j++) {
            if(j<i) {
                preLeft[j] = pre[j+1];
                inLeft[j] = in[j];
            } else if(j>i) {
                preRight[j-i-1] = pre[j];
                inRight[j-i-1] = in[j];
            }
        }
        root.left = reConstructBinaryTree(preLeft,inLeft);
        root.right = reConstructBinaryTree(preRight,inRight);
        return root;
    }
}
```

```
/* 先序遍历第一个位置肯定是根节点node，
  中序遍历的根节点位置在中间p，在p左边的肯定是node的左子树的中序数组，p右边的肯定是node的右子树的中序数组
  另一方面，先序遍历的第二个位置到p，也是node左子树的先序子数组，剩下p右边的就是node的右子树的先序子数组
  把四个数组找出来，分左右递归调用即可
*/
```

```
/**
 * Definition for binary tree
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
    */
    import java.util.ArrayList;
    import java.util.List;
    public class Solution {
    public TreeNode reConstructBinaryTree(int [] pre,int [] in) {
        ArrayList<Integer> preList = new ArrayList<Integer>(pre.length);
        ArrayList<Integer> inList = new ArrayList<Integer>(in.length);
        for (int i : pre)
            preList.add(i);
        for (int i : in)
            inList.add(i);
        return getRootNode(preList, inList);
    }
```

```
    private TreeNode getRootNode(List<Integer> preList, List<Integer> inList) {
        if (preList.size() == 0)
            return null;
        int rootVal = preList.get(0);
        TreeNode root = new TreeNode(rootVal);
        int index = inList.indexOf(rootVal);
        List<Integer> leftInList = inList.subList(0, index);
        List<Integer> rightInList = inList.subList(index+1, inList.size());
        List<Integer> leftPreList = preList.subList(1, leftInList.size()+1);
        List<Integer> rightPreList = preList.subList(preList.size()
                - rightInList.size(), preList.size());


        root.left = getRootNode(leftPreList, leftInList);
        root.right = getRootNode(rightPreList, rightInList);
        return root;
    }
    }
```

```
/* 先序遍历第一个位置肯定是根节点node，
   中序遍历的根节点位置在中间p，在p左边的肯定是node的左子树的中序数组，p右边的肯定是node的右子树的中序数组
   另一方面，先序遍历的第二个位置到p，也是node左子树的先序子数组，剩下p右边的就是node的右子树的先序子数组
   把四个数组找出来，分左右递归调用即可
*/
```

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。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。 解题思路 因为是树的结构，一般都是用递归来实现。 用数学归纳法的思想就是，假设最后一步，就是root的左右子树都已经重建好了，那么我只要考虑将root的左右子树安上去即可。 根据前序遍历的性质，第一个元素必然就是root，那么下面的工作就是如何确定root的左右子树的范围。 根据中序遍历的性质，root元素前面都是root的左子树，后面都是root的右子树。那么我们只要找到中序遍历中root的位置，就可以确定好左右子树的范围。 正如上面所说，只需要将确定的左右子树安到root上即可。递归要注意出口，假设最后只有一个元素了，那么就要返回。 我的答案 import java.util.Arrays;

```
public class Solution {
    public TreeNode reConstructBinaryTree(int [] pre,int [] in) {
        //数组长度为0的时候要处理
        if(pre.length == 0){
            return null;
        }
        int rootVal = pre[0];
        //数组长度仅为1的时候就要处理
        if(pre.length == 1){
            return new TreeNode(rootVal);
        }
        //我们先找到root所在的位置，确定好前序和中序中左子树和右子树序列的范围
        TreeNode root = new TreeNode(rootVal);
        int rootIndex = 0;
        for(int i=0;i<in.length;i++){
            if(rootVal == in[i]){
                rootIndex = i;
                break;
            }
        }
        //递归，假设root的左右子树都已经构建完毕，那么只要将左右子树安到root左右即可
        //这里注意Arrays.copyOfRange(int[],start,end)是[)的区间
        root.left =
reConstructBinaryTree(Arrays.copyOfRange(pre,1,rootIndex+1),Arrays.copyOfRange(in,0,rootIndex
));
```

```
        root.right =
reConstructBinaryTree(Arrays.copyOfRange(pre,rootIndex+1,pre.length),Arrays.copyOfRange(in,ro
otIndex+1,in.length));
        return root;
    }
}
```