剑指offer第4题

```java
package niuke;

import javax.swing.RootPaneContainer;

/**
* @author tfxidian E-mail: tfxidian@163.com
* @version 创建时间：2018年10月10日 下午9:21:15
* 类说明
* 根据先序遍历和中序遍历重建二叉树//重建二叉树之后很容易就可以后序遍历，因此此题也可以用来求解根据先序遍历和中序遍历求后序遍历
*/
public class ReConstructBinaryTree {


      public static class TreeNode {
          int val;
          TreeNode left;
          TreeNode right;
          TreeNode(int x) { val = x; }
      }

    public static void main(String[] args) {

        int pre[] = {1,2,4,7,3,5,6,8};
        int in[] =  {4,7,2,1,5,3,8,6};
        TreeNode treeNode;
        treeNode = reConstructBinaryTree(pre, in);
        preTree(treeNode);
        System.out.println("in order");
        inTree(treeNode);
        System.out.println("post order");
        postTree(treeNode);
    }

    public static void  preTree(TreeNode treeNode) {
        if (treeNode != null) {
            System.out.println(treeNode.val);
            preTree(treeNode.left);
            preTree(treeNode.right);
        }
    }

```

```
    public static void inTree(TreeNode treeNode) {
        if (treeNode!= null) {
            inTree(treeNode.left);
            System.out.println(treeNode.val);
            inTree(treeNode.right);
        }
    }

    public static void postTree(TreeNode treeNode) {
        if (treeNode!= null) {
            preTree(treeNode.left);
            preTree(treeNode.right);
            System.out.println(treeNode.val);
        }
    }
    public static TreeNode reConstructBinaryTree(int [] pre,int [] in) {

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

    }

    public static TreeNode reConstructBinaryTree(int[] pre, int startpre,
int endpre, int [] in , int startin, int endin) {

        if (pre.length!= in.length) {
            return null;
        }
        if (startpre>endpre|| startin> endin) {
            return null;
        }
        TreeNode rootNode = new TreeNode(pre[startpre]);
        int index;
        for (index = startin; index <= endin; index++) {
            if (pre[startpre] == in[index]) {
                break;
            }
        }
        rootNode.left = reConstructBinaryTree(pre, startpre+1, index-
startin+startpre, in, startin, index-1);
        rootNode.right = reConstructBinaryTree(pre, index-
startin+startpre+1, endpre, in, index+1, endin);
        return rootNode;
    }

}


```