重建二叉树

输入某二叉树的前序遍历和中序遍历的结果，请重建该二叉树，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}

解题思路：

- 前序遍历中，第一个数字总是树的根节点的值，在中序遍历中，根节点的值在序列的中间，左子树的节点的值位于根节点的值的左边，而右子树的节点的值位于根节点的值的右边。
- 首先在中序遍历中找到根节点 1 的下标为 4， 则左子树有 3 个，右子树有 4 个，则前序遍历中 1 后的 2，4，7 为其左子树中的值，3，5，6，8 则为其右子树的值。
- 这样我们就分别找到了根节点的左右子树的值，然后再用同样的方法继续下去，也就是递归的方法就可以构建出这个二叉树。

```
public class ConstructBinaryTree_07 {

    public static void main(String args[]) {
        // 前序遍历顺序
        int[] preOrder = {1, 2, 4, 7, 3, 5, 6, 8};
        // 中序遍历
        int[] inOrder = {4, 7, 2, 1, 5, 3, 8, 6};
        TreeNode root = reConstructBinaryTree(preOrder, inOrder);
        prePrint(root);
        System.out.println();
        inPrint(root);
    }

    // 缓存中序遍历数组每个值对应的索引
    private static Map<Integer, Integer> indexForInOrders = new HashMap<>();

    public static TreeNode reConstructBinaryTree(int[] pre, int[] in) {
        for (int i = 0; i < in.length; i++)
            indexForInOrders.put(in[i], i);
        return reConstructBinaryTree(pre, 0, pre.length - 1, 0);
    }

    private static TreeNode reConstructBinaryTree(int[] pre, int preL, int preR, int inL)
{
        if (preL > preR)
            return null;
        TreeNode root = new TreeNode(pre[preL]);
        int inIndex = indexForInOrders.get(root.value);
        // 计算中序遍历中左子树的个数
        int leftTreeSize = inIndex - inL;
        // 递归构建节点的左子树
        root.left = reConstructBinaryTree(pre, preL + 1, preL + leftTreeSize, inL);
        // 递归构建节点的右子树
        root.right = reConstructBinaryTree(pre, preL + leftTreeSize + 1, preR, inL +
```

```
leftTreeSize + 1);
        return root;
    }

    private static void prePrint(TreeNode node) {
        if (node != null) {
            System.out.print(node.value + " ");
            prePrint(node.left);
            prePrint(node.right);
        }
    }

    private static void inPrint(TreeNode node) {
        if (node != null) {
            inPrint(node.left);
            System.out.print(node.value + " ");
            inPrint(node.right);
        }
    }
}

class TreeNode {
    int value;
    TreeNode left;
    TreeNode right;

    public TreeNode(int value) {
        this.value = value;
    }
}
```