二叉树遍历问题

问题：
在二叉树中，我们知道存在三种遍历，前序，中序和后序遍历。而且中序遍历对于重构二叉树来说至关重要，知道前序或者后序，还必须得知道中序才能构造一棵二叉树。那么给出一棵树的前序(后序)以及中序遍历数组，请求出这棵树的后序(前序)遍历数组。

解法：
思想很简单，就是利用二叉树的遍历特性：

- 前序数组的第一个元素是二叉树的根节点；
- 后序数组的最后一个元素是二叉树的根节点；
- 根据前序和后序的根节点元素，可以在中序中确定左子树和右子树的元素个数，从而递归构造二叉树；

看代码：

```
1. public void constrcutFromPre(int[] preOrder, int[] inOrder) {
2.     int length1 = preOrder.length;
3.     int length2 = inOrder.length;
4.     helper(0, length1-1, 0, length2-1, preOrder, inOrder);
5. }
6. 
7. public void helper(int pStart, int pEnd, int iStart, int iEnd, int
   [] preOrder, int[] inorder) {
8.     int root = preOrder[pStart];
9.     int index = -1;
10.    for (int i = iStart; i <= iEnd; i++) {
11.        if (inOrder[i] == root) {
12.            index = i;
13.        }
14.    }
15.    helper(pStart + 1, index - iStart + pStart, iStart, index - 1,
   preOrder, inOrder);
16.    helper(index - iStart + pStart + 1, pEnd, index + 1, iEnd, pre
   Order, inOrder);
17.    System.out.println(root);//因为后序输出
18. }
```

```
1. public void constrcutFromPre(int[] postOrder, int[] inOrder) {
2.     int length1 = postOrder.length;
3.     int length2 = inOrder.length;
4.     helper(0, length1-1, 0, length2-1, postOrder, inOrder);
5. }
6. 
7. public void helper(int pStart, int pEnd, int iStart, int iEnd, int
   [] postOrder, int[] inorder) {
8.     int root = postOrder[pEnd];
9.     int index = -1;
10.    for (int i = iStart; i <= iEnd; i++) {
11.        if (inOrder[i] == root) {
12.            index = i;
13.        }
14.    }
15.    System.out.println(root);//因为前序输出
16.    helper(pStart, index - iStart + pStart - 1, iStart, index - 1,
   postOrder, inOrder);
17.    helper(index - iStart + pStart, pEnd, index + 1, iEnd, postOrd
   er, inOrder);
18. }
```