之前讲解的都是遍历二叉树，这次该构造二叉树了

106.从中序与后序遍历序列构造二叉树

根据一棵树的中序遍历与后序遍历构造二叉树。

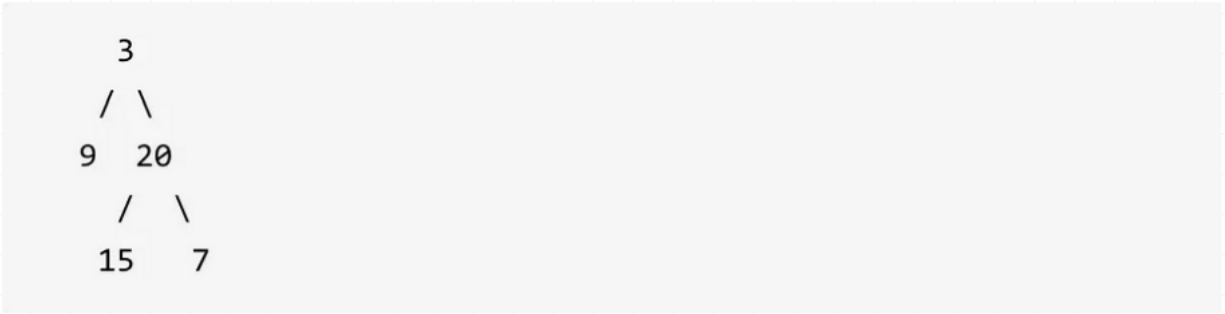
注意：你可以假设树中没有重复的元素。

例如，给出

中序遍历 inorder = [9,3,15,20,7]

后序遍历 postorder = [9,15,7,20,3]

返回如下的二叉树：



> 事先理解了通过**先序遍历序列和中序遍历序列**构造**二叉树的过程**，以及通过**后序遍历序列和中序遍历序列**构造**二叉树**的过程

> 由于通过两个序列构造的二叉树具有**一层层切割**的过程，就应该想到用到了**递归**
> 第一步:如果**数组大小**为零的话，说明是**空节点**
> 第二步:如果不为空,那么取后序数组最后一个元素作为节点元素
> 第三步:找到后序数组的最后一个元素在中序数组的位置，作为切割点
> 第四步:切割中序数组，切成中序左数组和中序右数组 （顺序别搞反了，一定是先切中序数组）
> 第五步:切割后序数组，切成后序左数组和后序右数组
> 第六步:递归处理左区间和右区间

得到以下算法框架

```
TreeNode* traversal(vector<int>& inorder,vector<int>& postorder){
 //第一步如果后序数组为空
 if(postoder.size()==0) return NULL;

 //第二部取出后序数组的最后一个元素
 int rootvalue=postorder[postorder.size()-1];
 TreeNode*root=new TreeNode(rootValue);
```

```
8
9   //叶子结点
10   if(postorder.size()==1) return root;
11
12   //第三步：找切割点
13   int delimiterIndex;
14   for(delimiterIndex=0;delimiterIndex<inorder.size();delimiterIndex++){
15   if(inorder[delimiterIndex]==rootValue) break;
16   }
17   // 第四步：切割中序数组，得到 中序左数组和中序右数组
18           // 第五步：切割后序数组，得到 后序左数组和后序右数组
19
20           // 第六步
21           root->left = traversal(中序左数组, 后序左数组);
22           root->right = traversal(中序右数组, 后序右数组);
23
24           return root;
25   }
```