# 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：

```
    3
   / \
  9  20
    /  \
   15   7
```

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left
), right(right) {}
 * };
 */
class Solution {
public:

    TreeNode* _build(vector<int>& postorder, vector<int>& inorder,int& post
idx,int start,int end)
    {
        if(start>end)//不包含任何元素
            return nullptr;
        TreeNode* cur=new TreeNode(postorder[postidx]);
        //左右子树区间
        int curidx=start;
```

```
        for(;curidx<=end;curidx++)
        {
            if(inorder[curidx]==postorder[postidx])
                break;//找到分界点 curidx
        }
        if(curidx<end)
        {
            cur->right=_build(postorder,inorder,--postidx,curidx+1,end);
        }
        else
            cur->right=nullptr;
        if(start<curidx)
        {
            cur->left=_build(postorder,inorder,--postidx,start,curidx-1);
        }
        else
            cur->left=nullptr;

        return cur;
    }

    TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {
        int postidx=postorder.size()-1;
        return _build(postorder,inorder,postidx,0,inorder.size()-1);
    }
};
```