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，给出

前序遍历 preorder = [3,9,20,15,7]

中序遍历 inorder = [9,3,15,20,7]

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    unordered_map<int,int> pos;//记录节点位置
    TreeNode* buildTree(vector<int> pre,vector<int> vin) {
        int n=pre.size();//获取 pre 前序遍历 树长度
        //记录中序序列位置信息
        for(int i=0;i<n;i++)
        {
            pos[vin[i]]=i;
        }
        return dfs(pre,vin,0,n-1,0,n-1);//深度优先遍历搜索 给定左边界和右边界
0，n-1
    }
     TreeNode* dfs(vector<int> pre,vector<int> vin,int pl,int pr,int vl,int
 vr)
     {
         //前序：根左右
         //中序：左根右
         if(pl>pr) return NULL;
         //找根节点
         TreeNode* root=new TreeNode(pre[pl]);
         //左子树长度 k  根据中序根的位置 减去中序左边界 则为左子树长度
         int k=pos[pre[pl]]-vl;
         //vl+k 为中序 根位置
         //pl 为前序 根位置
         //pl+1 到 pl+k 前序深度优先遍历 所有左节点
         //vl 到 vl+k-1;中序深度优先遍历 所有左节点
```

```
        //pl+k+1 到 pr 前序深度优先遍历 所有右节点
        //vl+k+1 到 vr;中序深度优先遍历 所有右节点
        root->left=dfs(pre,vin,pl+1,pl+k,vl,vl+k-1);
        root->right=dfs(pre,vin,pl+k+1,pr,vl+k+1,vr);

        return root;
    }
};
```