## 题目描述：

## 输入某二叉树的前序遍历和中序遍历的结果，请重建出该二叉树。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}，则重建二叉树并返回。

## 解题思路：

递归的思想：

先创建一个根节点：root

然后，需要将左子树和右子树的连接到根节点上去、

所有求出左子树和右子树的先序遍历序列和中序遍历序列

通过函数的递归调用，求出整个子树上

root->left = reConstructBinaryTree(pre\_left,vin\_left);

root->right = reConstructBinaryTree(pre\_right, vin\_right);

## 算法描述：

    class Solution {

    public:

        struct TreeNode\* reConstructBinaryTree(vector<int> pre,vector<int> in) {

            int inlen=in.size();

            if(inlen==0)

                return NULL;

            vector<int> left\_pre,right\_pre,left\_in,right\_in;

            //创建根节点，根节点肯定是前序遍历的第一个数

            TreeNode\* head=new TreeNode(pre[0]);

            //找到中序遍历根节点所在位置,存放于变量gen中

            int gen=0;

            for(int i=0;i<inlen;i++)

            {

                if (in[i]==pre[0])

                {

                    gen=i;

                    break;

                }

            }

            //对于中序遍历，根节点左边的节点位于二叉树的左边，根节点右边的节点位于二叉树的右边

            //利用上述这点，对二叉树节点进行归并

            for(int i=0;i<gen;i++)

            {

                left\_in.push\_back(in[i]);

                left\_pre.push\_back(pre[i+1]);//前序第一个为根节点

            }

            for(int i=gen+1;i<inlen;i++)

            {

                right\_in.push\_back(in[i]);

                right\_pre.push\_back(pre[i]);

            }

            //和shell排序的思想类似，取出前序和中序遍历根节点左边和右边的子树

            //递归，再对其进行上述所有步骤，即再区分子树的左、右子子数，直到叶节点

           head->left=reConstructBinaryTree(left\_pre,left\_in);

           head->right=reConstructBinaryTree(right\_pre,right\_in);

           return head;

        }

    };

自己的答案：

/\*\*

\* Definition for binary tree

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode(int x) : val(x), left(NULL), right(NULL) {}

\* };

\*/

class Solution {

public:

TreeNode\* reConstructBinaryTree(vector<int> pre,vector<int> vin) {

int len = vin.size();

int pos;

if(pre.empty() || vin.empty())

return NULL;

TreeNode \* root = new TreeNode(pre[0]);

for(int i=0; i<len; i++)

{

if(vin[i] == pre[0])

{

pos = i;

break;

}

}

vector<int> pre\_left,pre\_right,vin\_left,vin\_right;

for(int i=0; i<pos;i++)

{

pre\_left.push\_back(pre[i+1]);

vin\_left.push\_back(vin[i]);

}

for(int i=pos+1; i<len;i++)

{

pre\_right.push\_back(pre[i]);

vin\_right.push\_back(vin[i]);

}

root->left = reConstructBinaryTree(pre\_left,vin\_left);

root->right = reConstructBinaryTree(pre\_right,vin\_right);

return root;

}

};