面试题7：重建二叉树

2018/8/2 15:21:31

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。

C++

```
/**
 * 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 lenth = pre.size();

        //① 空树
        if(lenth <= 0 || &pre == nullptr || &vin == nullptr)
            return nullptr;
        return Core(&pre[0],&pre[0] + lenth - 1,&vin[0],&vin[0] + lenth - 1);
    }

    //S第一个数，E最后一个数
    TreeNode* Core(int* preS , int* preE , int* vinS , int* vinE){
        //初始化根节点
        int rootvalue = *preS;
        TreeNode* root = new TreeNode(rootvalue);{

            //② 只有一个根节点
            if(preS ==preE){
                if(vinS == vinE && *vinS == *preS){
                    return root;
                }else{
                    throw exception();
                }
            }

            //求leftlength
            //先找vinroot
            int* vinroot = vinS; //bug1:int* vinroot = preS;
            while(vinroot <= vinE && *vinroot != rootvalue) //这里的二叉树一定不含重复数字不然就不能这样找vinroot了
                ++ vinroot;

            //③ 如果vin中没有root，报错
            if(vinroot == vinE && *vinroot != rootvalue)
                throw exception();

            int leftlength = vinroot - vinS; //bug2:int leftlength = vinroot - preS;

            //左子树
            if(leftlength > 0) //bug3网络判断递归条件，不判断会内存超限
            root->left = Core(preS + 1,preS + leftlength,vinS,vinroot - 1);

            //右子树
            if(preE - preS > leftlength) //bug3忘判断递归条件
            root->right = Core(preS +leftlength + 1,preE,vinroot + 1,vinE);

            return root;
        }
    }
};
```

Java

```
/**
 * Definition for binary tree
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Solution {
    public TreeNode reConstructBinaryTree(int [] pre,int [] in) {
        TreeNode root = reConstructBinaryTree(pre,0,pre.length-1,in,0,in.length-1);
        return root;
    }
```

```java

    private TreeNode reConstructBinaryTree(int [] pre , int startPre , int endPre , int [] in,int startIn , int endIn){
        if(startPre > endPre || startIn > endIn)
            return null;
        TreeNode root = new TreeNode(pre[startPre]);
        for(int i = startIn ; i <= endIn ; i++)
            if(in [i] == pre[startPre]){
                root.left = reConstructBinaryTree(pre,startPre + 1,startPre+i-startIn,in,startIn,i-1);
                root.right = reConstructBinaryTree(pre,i-startIn + startPre + 1 , endPre , in , i + 1 ,endIn);
                    break;
            }
        return root; //数组的索引与指针的思想
    }
}
```