106. Construct Binary Tree from Inorder and Postorder Traversal

题目描述：https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal/

> 给定一棵二叉树的中序遍历和后序遍历，求这棵二叉树。

解题思路：

后序遍历的最后一个节点是根节点，然后从中序遍历可以得到根节点左右的分别是左右子树，然后迭代求解。

代码1：

自己写的，Memory Limits Exceeded

```
/**
 * 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:
    int findloc(vector<int> v, int n){
        for(int i = 0; i < v.size(); i++){
            if(n == v[i])
                return i;
        }
        return -1;
    }
    vector<int> getv(vector<int> v, int s, int e){
        vector<int> r;
        for(int i = s; i <= e; i++){
            r.push_back(v[i]);
        }
        return r;
    }
    TreeNode* buildTree(vector<int>& in, vector<int>& post) {
        if(in.size()<=0)
            return NULL;
        TreeNode *root = new TreeNode(post[post.size()-1]);
        int insplit = findloc(in, root->val);
        if(insplit < 0)
            return root;
        vector<int> lin; lin = getv(in, 0, insplit-1);
        vector<int> rin; rin = getv(in, insplit+1, in.size()-1);
        vector<int> lpo; lpo = getv(post, 0, insplit-1);
        vector<int> rpo; rpo = getv(post, insplit, post.size()-2);
        root->left = buildTree(lin, lpo);
        root->right = buildTree(rin, rpo);
        lin.clear();rin.clear();lpo.clear();rpo.clear();
        return root;
    }
};
```

代码2:

使用pos记录两个vector访问到哪儿里

```
/**
 * 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:
    TreeNode* CreateTree(vector<int> &in, vector<int> &post, int is, int ie, int p
s, int pe){
        if(pe < ps){
            return NULL;
        }
        int pos = 0;
        TreeNode *root = new TreeNode(post[pe]);
        for(int i = is; i <= ie; i++){
            if(in[i] == root->val){
                pos = i;
                break;
            }
        }
        int llength = pos - is;
        int rlength = ie - pos;
        root->left = CreateTree(in, post, is, pos - 1, ps, ps + llength - 1);
        root->right = CreateTree(in, post, pos + 1, ie, pe - rlength, pe - 1);

        return root;
    }
    TreeNode* buildTree(vector<int>& in, vector<int>& post) {
        return CreateTree(in, post, 0, in.size()-1, 0, post.size()-1);
    }

};
```