1. Binary Tree Postorder Traversal

Given a binary tree, return the *postorder* traversal of its nodes’ values.

**Example:**

Input: [1,null,2,3]  
 1  
 \  
 2  
 /  
 3  
  
Output: [3,2,1]

**Follow up:** Recursive solution is trivial, could you do it iteratively?

**解**

每个节点需要经过两次，分别找左子树和右子树 step1： 将根节点BT赋值给p step2： 如果p非空，则p进栈，将进栈标志设置为第一次，再将p指向p节点的左孩子；重复直到p空 step3： 若栈非空，从栈顶弹出元素赋值给p，如果p是第一次进栈，则将p进栈，置进栈标志为第二次进栈，p指向p的右孩子及节点；否则则访问节点p，**同时将p强行置空**(下一轮直接进入step3，弹出栈顶元素，即为根节点） step4： 重复2 3,直到栈空树也空

class Solution {  
public:  
 vector<int> postorderTraversal(TreeNode\* root) {  
 stack<pair<TreeNode\*, int>>s;  
 TreeNode \*p = root;  
 vector<int>path;  
 while(p || !s.empty()){  
 while(p){  
 s.push(make\_pair(p, 1));  
 p = p->left;  
 }  
 if(!s.empty()){  
 pair tmp = s.top();  
 s.pop();  
 p = tmp.first;  
 if(tmp.second == 1){  
 s.push(make\_pair(p, 2));  
 p = p->right;  
 }else{  
 path.push\_back(p->val);  
 p = NULL;  
 }  
 }  
 }  
 return path;  
 }  
};