二叉树前序遍历

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left
), right(right) {}
 * };
 */
class Solution {
public:
    vector<int> preorderTraversal(TreeNode* root) {
        stack<TreeNode*> st;
        vector<int> res;
        TreeNode* cur=root;
        while(cur||!st.empty())
        {
            while(cur)
            {
                //访问最左路径保存并入栈
                res.push_back(cur->val);
                st.push(cur);
                cur=cur->left;
            }
            //取最左路径最后一个节点的最右路径（访问左路径时最后遇到的右子树）
            cur=st.top();
            st.pop();
            cur=cur->right;
        }
        return res;
    }
};
```

二叉树的前序遍历

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left
), right(right) {}
 * };
 */
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
        stack<TreeNode*> st;
        vector<int> res;
        TreeNode* cur=root;
        while(cur||!st.empty())
        {
            while(cur)//找左 不访问
            {
                st.push(cur);
                cur=cur->left;
            }
            cur=st.top();
            res.push_back(cur->val);
            st.pop();
            cur=cur->right;//找右  右移
        }
        return res;
    }
};
```

二叉树的后续遍历

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left
), right(right) {}
 * };
 */
class Solution {
public:
    vector<int> postorderTraversal(TreeNode* root) {
        stack<TreeNode*> st;
        vector<int> res;
        TreeNode* cur=root;
        TreeNode* prev;
        while(cur||!st.empty())
        {
            while(cur)
            {
                st.push(cur);
                cur=cur->left;
            }
            TreeNode* top=st.top();
            //没有右子树，或右子树访问完成（上一次访问节点指向右子树根结点）则可
以访问
            if(top->right==prev||top->right==nullptr)
            {
                res.push_back(top->val);
                st.pop();
                prev=top;
            }
            else
            {
                //否则，栈顶元素存在右子树且右子树第一次访问  更新当前节点指向栈
顶元素的右子树
                cur=top->right;
            }
        }
        return res;
    }
};
```