```
/*
给定一个二叉树，返回它的 后序 遍历。

示例:

输入: [1,null,2,3]
   1
    \
     2
    /
   3

输出: [3,2,1]

进阶: 递归算法很简单，你可以通过迭代算法完成吗?

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/binary-tree-postorder-traversal
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 递归法:按照左右根的顺序依次递归即可.
- 迭代法:
  - 基本思想:用栈保存先前走过点路径,以便可以在访问完子树后,可以利用栈中的信息,回退到当前节点的父亲节点.
- 后序遍历是先访问左、右子树，再访问根节点,而在迭代过程中，并不知道是从左子树回退到根节点,还是从右子树回退到根节点.
- 若是从左子树回退到根节点,此时就应该去访问右子树,而如果从右子树回退到根节点，此时就应该访问根节点.
- 在后序遍历时应当保存前一个访问到的节点信息,以便在退栈时可以知道是左子树返回还是右子树返回.

C++

```
/**
 * 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
{

    private:
        vector<int> ret_val;
        void __dfsPostOrder(TreeNode* node)
```

```
        {
            if(node==NULL)
            {
                return ;
            }
            __dfsPostOrder(node->left);
            __dfsPostOrder(node->right);
            ret_val.push_back(node->val);
        }

    public:
        vector<int> postorderTraversal(TreeNode* root)
        {
            ret_val.clear();
            __dfsPostOrder(root);
            return ret_val;
        }
};

/*
执行结果:
通过
显示详情
执行用时 :4 ms, 在所有 C++ 提交中击败了83.30% 的用户
内存消耗 :9.2 MB, 在所有 C++ 提交中击败了43.66%的用户
*/
```

方法二:C++_迭代法

```
/**
 * 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:
        vector<int> postorderTraversal(TreeNode* root)
        {
            vector<int>         ret_val ;
            stack<TreeNode*>    stnp    ;

            TreeNode*           curr    =   root;
            TreeNode*           last    =   NULL;

            if(root)
            {
                while(1)
                {
                    while(curr)
                    {
```

```
                        stnp.push(curr);
                        curr = curr -> left;
                    }
                    /*遇到NULL回退到上一个节点*/
                    curr = stnp.top();
                    /*回退情况*/
                    if(  (last==curr->right)
                       ||(!curr->right)
                    )
                    {
                        /*从右节点回退或者没有右节点*/
                        stnp.pop();
                        ret_val.push_back(curr->val);
                        last    =   curr;
                        curr    =   NULL;
                    }
                    else
                    {
                        /*从左节点回退且存在右节点*/
                        curr = curr->right;
                        last = NULL;
                    }

                    /*当前指针为空且栈已经为空则跳出*/
                    if((!curr)&&(stnp.empty()))
                    {
                        break;
                    }
                }
            }
            return ret_val;
        }
};

/*
执行结果:
通过
显示详情
执行用时 :4 ms, 在所有 C++ 提交中击败了83.30% 的用户
内存消耗 :9.1 MB, 在所有 C++ 提交中击败了74.48%的用户
*/
```

AlimyBreak
2019.09.29