```
1 /*
2
   给定一个二叉树,返回它的 后序 遍历。
3
4
   示例:
5
6
   输入: [1,null,2,3]
7
    1
8
9
      2
10
    3
11
12
   输出: [3,2,1]
13
14
   进阶: 递归算法很简单, 你可以通过迭代算法完成吗?
15
16
17
   来源:力扣(LeetCode)
18
   链接: https://leetcode-cn.com/problems/binary-tree-postorder-traversal
19
   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。
20 */
```

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++

```
1 /**
 2
    * Definition for a binary tree node.
 3
    * struct TreeNode {
4
         int val;
          TreeNode *left;
         TreeNode *right;
 7
          TreeNode(int x) : val(x), left(NULL), right(NULL) {}
    * };
8
    */
9
   class Solution
10
11
   {
12
13
       private:
14
            vector<int> ret_val;
            void __dfsPostOrder(TreeNode* node)
15
```

```
16
17
               if(node==NULL)
18
19
                   return ;
20
21
               __dfsPostOrder(node->left);
22
               __dfsPostOrder(node->right);
               ret_val.push_back(node->val);
23
24
           }
25
       public:
26
27
           vector<int> postorderTraversal(TreeNode* root)
28
           {
29
               ret_val.clear();
30
               __dfsPostOrder(root);
31
               return ret_val;
32
           }
33
   };
34
35
36 执行结果:
37
   通过
38
   显示详情
39 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
41 */
```

方法二:C++_迭代法

```
1 /**
 2
    * Definition for a binary tree node.
 3
    * struct TreeNode {
    *
4
          int val;
 5
          TreeNode *left;
 6
          TreeNode *right;
7
           TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8
    * };
9
    */
10
11
   class Solution
12
13
        public:
14
            vector<int> postorderTraversal(TreeNode* root)
15
16
                vector<int>
                                    ret_val;
17
                stack<TreeNode*>
                                   stnp ;
18
19
                TreeNode*
                                    curr =
                                                root;
20
                TreeNode*
                                    last
                                                NULL;
21
22
                if(root)
23
24
                    while(1)
25
26
                        while(curr)
27
                        {
```

```
28
                         stnp.push(curr);
29
                         curr = curr -> left;
                      }
30
                      /*遇到NULL回退到上一个节点*/
31
32
                      curr = stnp.top();
                      /*回退情况*/
33
34
                      if( (last==curr->right)
35
                        ||(!curr->right)
36
                      )
37
                      {
38
                         /*从右节点回退或者没有右节点*/
39
                         stnp.pop();
40
                         ret_val.push_back(curr->val);
41
                         last = curr;
                         curr = NULL;
42
43
                      }
44
                      else
45
                      {
46
                         /*从左节点回退且存在右节点*/
47
                         curr = curr->right;
48
                         last = NULL;
49
                      }
50
51
                      /*当前指针为空且栈已经为空则跳出*/
52
                      if((!curr)&&(stnp.empty()))
53
54
                         break;
55
                      }
56
                  }
57
58
              return ret_val;
59
           }
60
   };
61
62
63
   执行结果:
   通过
64
65
   显示详情
66 执行用时 :4 ms, 在所有 C++ 提交中击败了83.30% 的用户
67
   内存消耗 :9.1 MB, 在所有 C++ 提交中击败了74.48%的用户
68 */
```

AlimyBreak 2019.09.29