LeetCode 第 145 号问题：二叉树的后序遍历

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步博客：https://www.algomooc.com*

题目来源于 LeetCode 上第 145 号问题：二叉树的后序遍历。题目难度为 Hard，目前通过率为 25.8% 。

题目描述

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?

题目解析

用**栈(Stack)**的思路来处理问题。

后序遍历的顺序为**左-右-根**，具体算法为：

- 先将根结点压入栈，然后定义一个辅助结点 head
- while 循环的条件是栈不为空
- 在循环中，首先将栈顶结点t取出来
- 如果栈顶结点没有左右子结点，或者其左子结点是 head，或者其右子结点是 head 的情况下。我们将栈顶结点值加入结果 res 中，并将栈顶元素移出栈，然后将 head 指向栈顶元素
- 否则的话就看如果右子结点不为空，将其加入栈
- 再看左子结点不为空的话，就加入栈

动画描述

代码实现

```
public class Solution {
    public List<Integer> postorderTraversal(TreeNode root) {
    List<Integer> res = new ArrayList<Integer>();
    if(root == null)
        return res;
    Stack<TreeNode> stack = new Stack<TreeNode>();
    stack.push(root);
    while(!stack.isEmpty()){
```

```
        TreeNode node = stack.pop();
        if(node.left != null) stack.push(node.left);//和传统先序遍历不一样，先将左结点入栈
        if(node.right != null) stack.push(node.right);//后将右结点入栈
        res.add(0,node.val);                          //逆序添加结点值
    }
    return res;
   }
}
```