

LeetCode 第 94 号问题：二叉树的中序遍历

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 94 号问题：二叉树的中序遍历。题目难度为 Medium

题目描述

给定一个二叉树，返回它的 **中序** 遍历。

示例:

输入: [1,null,2,3]

```
  1
   \
    2
   /
  3
```

输出: [1,3,2]

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

第一种方法: 递归

二叉树的中序遍历相信大家已经很熟悉了.操作流程就是 **左 -> 打印 -> 右**.

那就按照 **左 -> 打印 -> 右** 这种顺序遍历树就可以了，递归函数实现

- 终止条件:当前节点为空时
- 函数内: 递归的调用左节点，打印当前节点，再递归调用右节点

参考代码

```
// lang=javascript
var inorderTraversal = function(root) {
    let res = [];
    handle(root,res);
    return res;
};
function handle(root, res) {
    if (root !== null) {
        handle(root.left, res);
        res.push(root.val);
        handle(root.right, res);
    }
}
```

复杂度分析

- 时间复杂度: $O(n)$,

- 空间复杂度: $O(h)$, h 是树的高度

第二种方法: 迭代

这题的真正难点在于如何用非递归的方式实现。

递归的调用过程是不断往左边走, 当左边走不下去了, 就打印节点, 并转向右边, 然后右边继续这个过程, 是函数自己调用自己, 一层层的嵌套下去, 操作系统/虚拟机自动帮我们用了**栈**来保存了每个调用的函数, 现在我们需要自己模拟这样的调用过程。

栈的特性是**后进先出**, 那么我们将遍历左子树的节点压栈, 当找不到左子树时, 栈顶就是最底层的左子树, 出栈打印出来; 接着转向右子树父节点, 继续遍历父节点的左子树并压栈, 循环如此。

因此遍历的过程就是:

1. 压栈根节点
2. 遍历左子树, 压栈, 直到左子树为空
3. 出栈栈顶元素, 打印
4. 转向右子树, 重复 1, 2, 3步骤

动画理解



Animation2

参考代码

```
// lang=javascript
var inorderTraversal = function(root) {
    let res = [],
        stack = [],
        node = root;
    while (stack.length > 0 || node !== null) {
        while (node) {
            stack.push(node);
            node = node.left;
        }
        node = stack.pop();
        res.push(node.val);
        node = node.right;
    }
    return res;
}
```

复杂度分析

- 时间复杂度: $O(n)$
- 空间复杂度: $O(n)$



qrcode