二叉树 #a

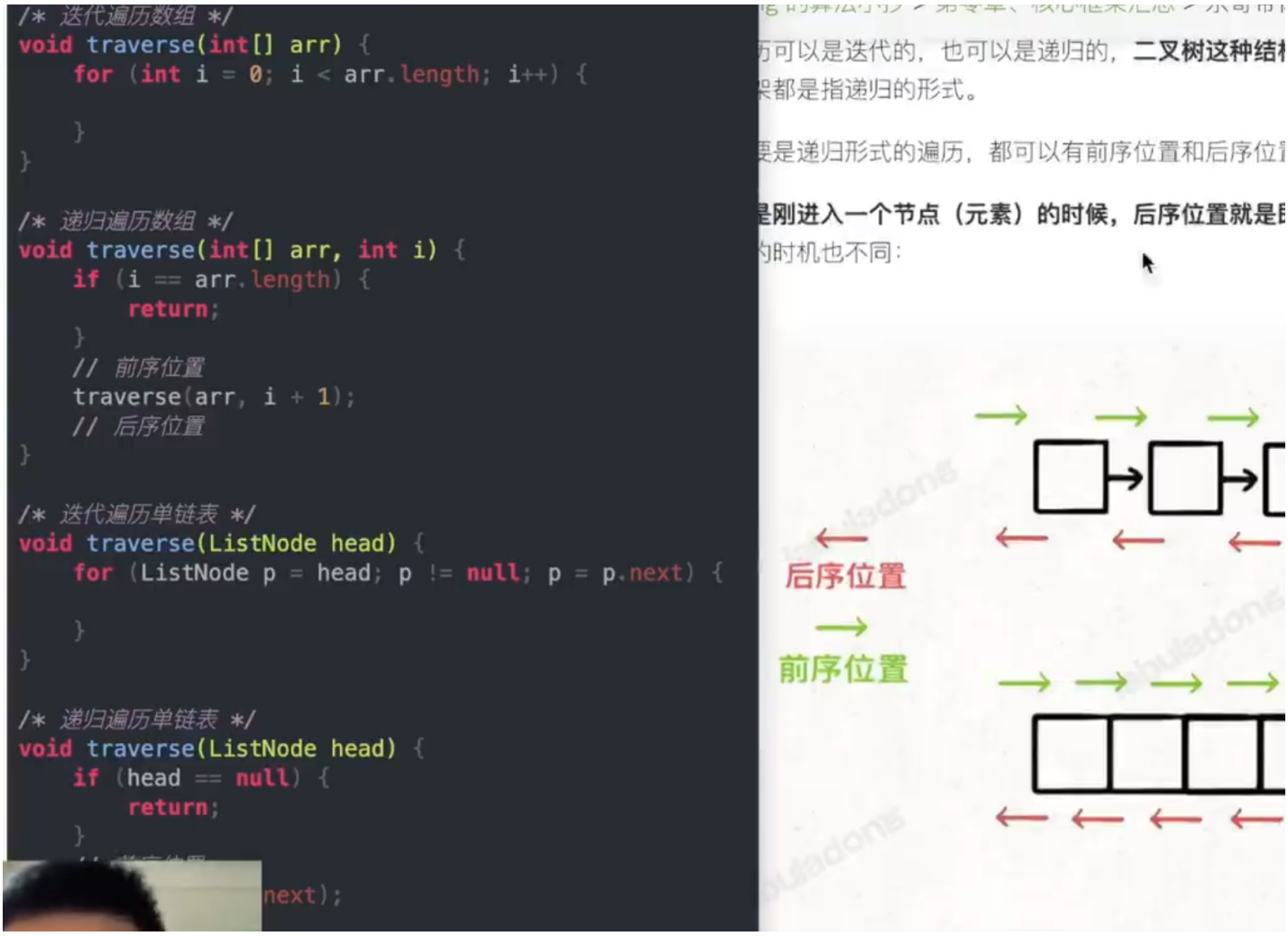
leetcode

no		desc
1	二叉树直径	通过最大深度后序位置获取直径
2	翻转二叉树	翻转
3	填充右侧节点	填充next域
4	展开二叉树	拉平二叉树

sumup

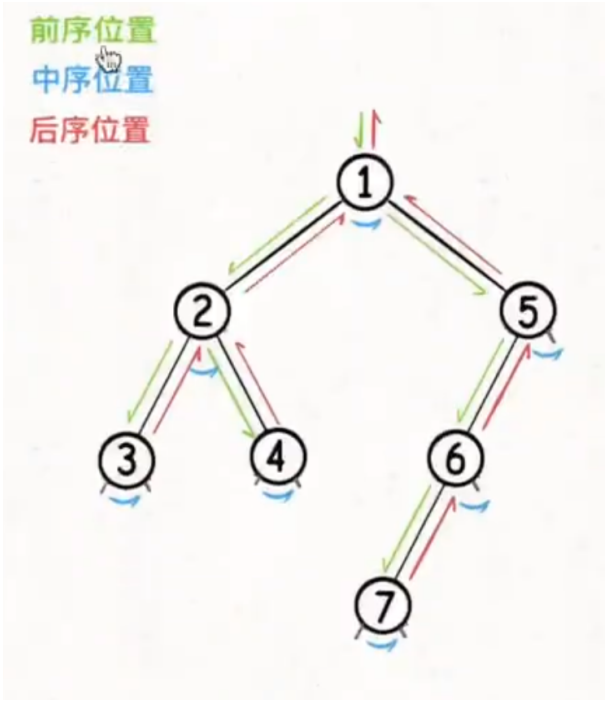
- ==遍历一遍二叉树得到答案, 可以的话用traverse()配合外部变量遍历
- ==是否可以通过递归, 通过子问题(子树)推导原问题答案, 写出递归函数定义, 充分利用函数返回值, 分解问题的思维模式

> 单独一个节点, 需要做什么, 需要在前中后哪个位置做, 其他节点不用多想, **递归函数**会执行相同操作



& 代码写到哪里, 递归遍历, 就可以有前序位置和后序位置

😆 *遍历的本质*



- **前序**: 进入一个节点的时刻运行
- **后序**: 离开一个节点的时刻运行
- **中序**: 遍历完该节点的所有左子树, 即将开始该节点的右子树时刻运行

> 前序位置意味着当前节点只能使用函数参数的数据, 后序位置意味着不仅可以获取参数数据, 还可以获取子树通过**函数返回值**传递回来的数据

```python
# 前序: 打印每层的层级
def traverse(node, level):
    if node is None: return

    print(node.val, level)
    traverse(node.left, level + 1)
    traverse(node.right, level + 1)


# 后序: 打印每层左右子树的节点数
def traverse_son(node):
    """ 返回子树的节点个数 """
    if node is None: return 0

    left_num = traverse_son(node.left)
    right_num = traverse_son(node.right)

    print(node.val, left_num, right_num)

    return left_num + right_num + 1  # 给上一层的是当前层左右和加上自己
```