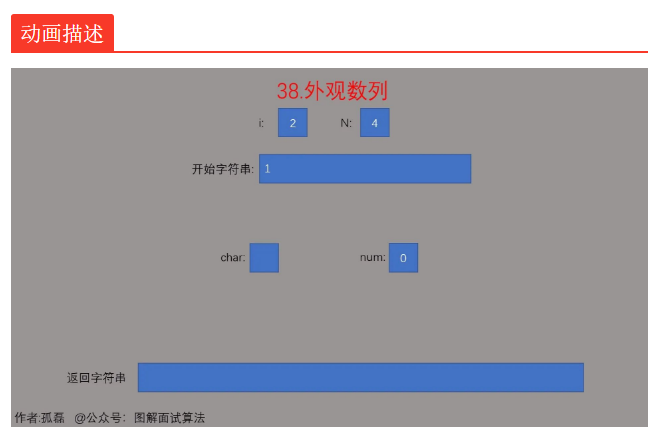
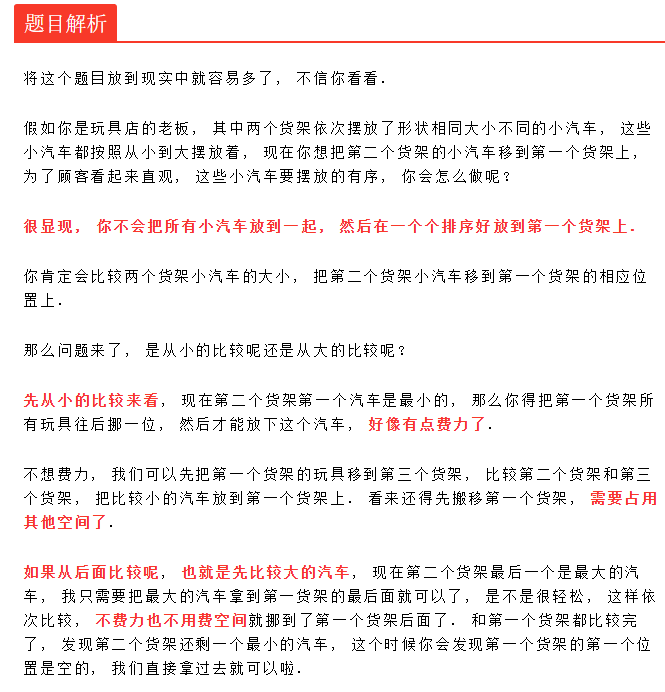
# 简单

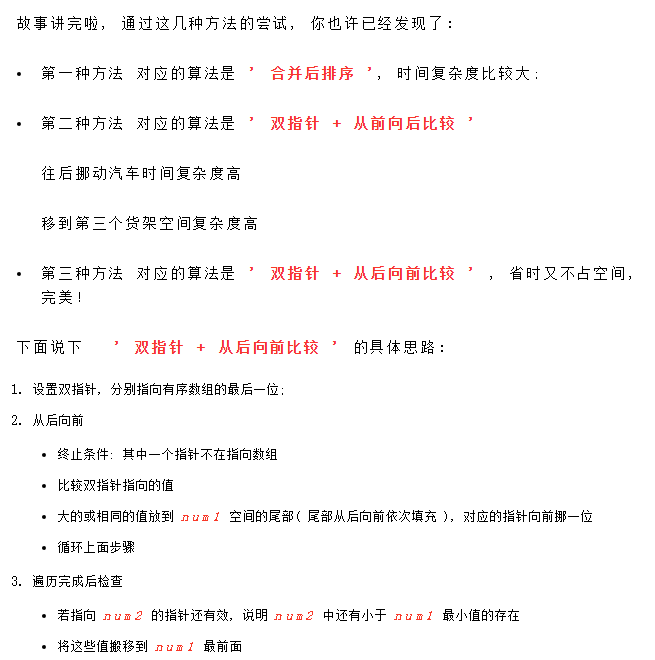
## 38\_外观数列



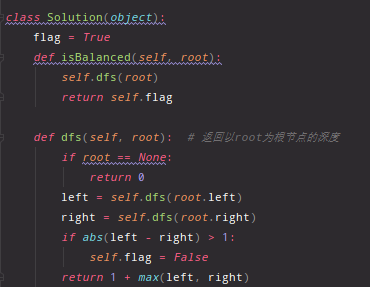


## 88\_合并两个有序数组





110\_平衡二叉树



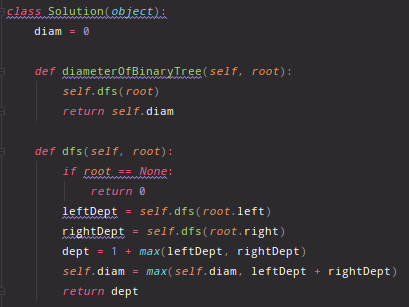
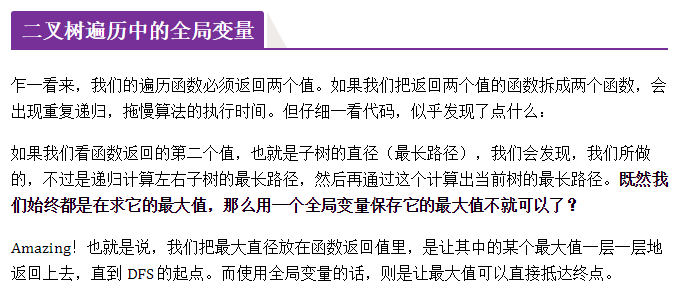
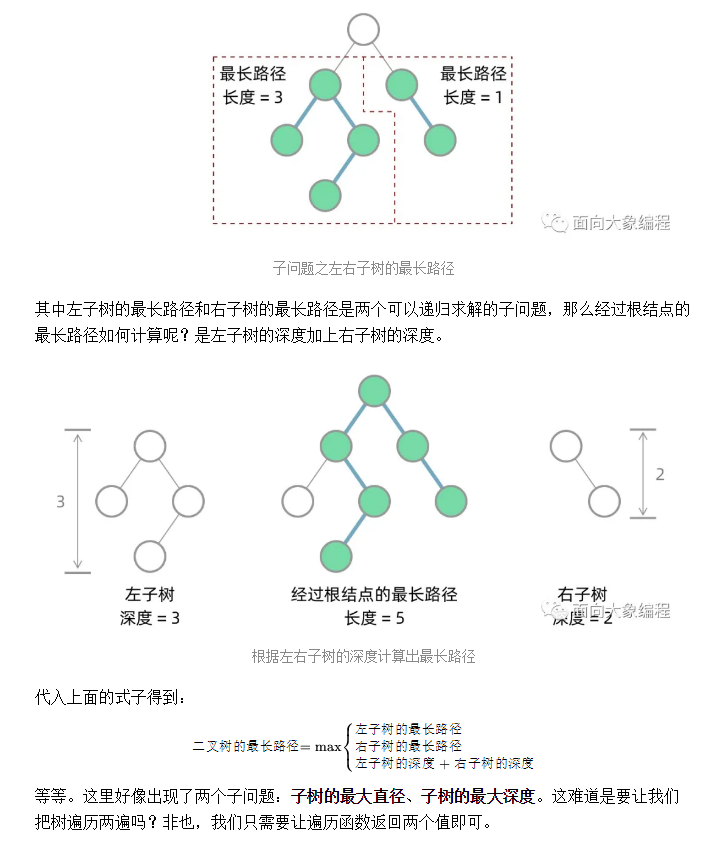
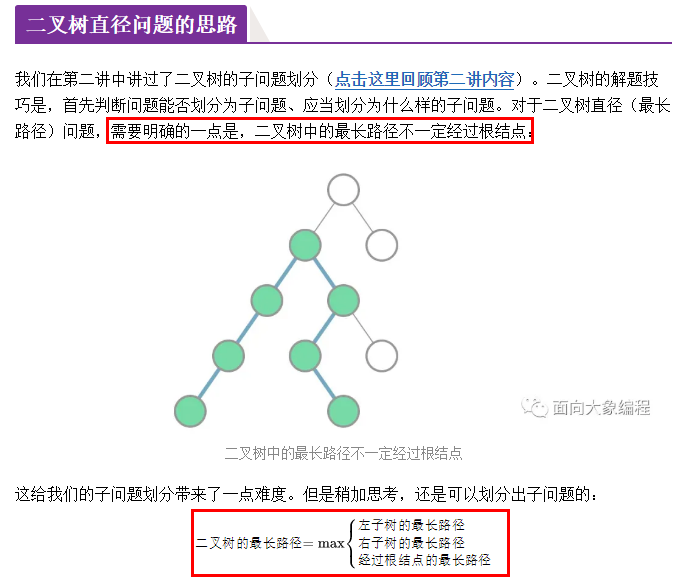
## 290\_单词规律



改进：不需要hashset，只需要讲Map中的values值取出比较即可。



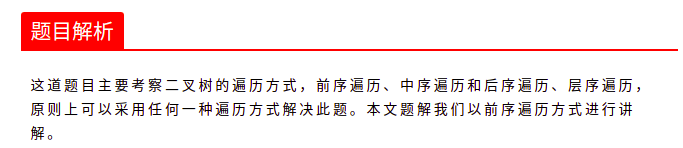
## 543\_二叉树的直径

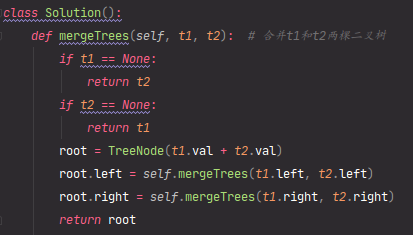


## 563\_二叉树的坡度



## 617\_合并二叉树



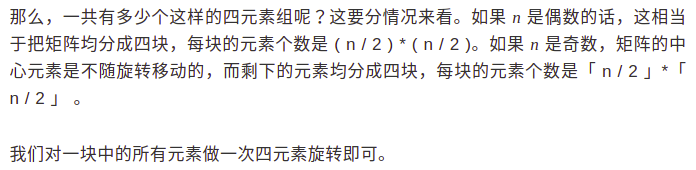


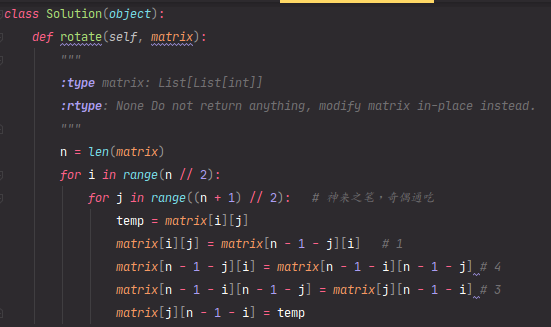
# 中等

## 48\_旋转图像



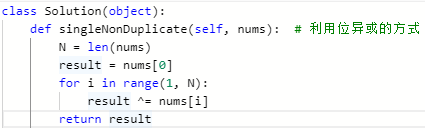


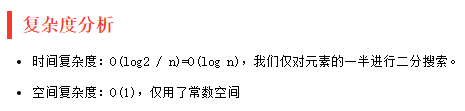
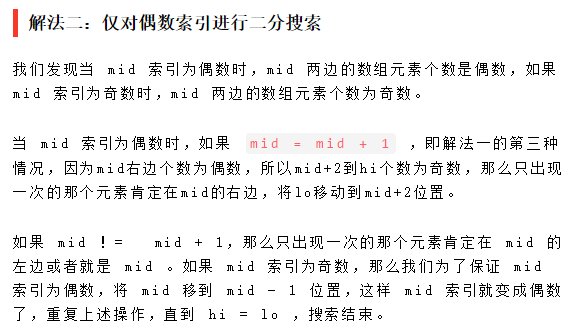




## 540\_有序数组中的单一元素

解法一：





二分了N次，直到只剩下一个元素，就是我们要找的那个数:

2 ^ N = n ----> N = logn

## 1372\_二叉树中的最长交错路径

题目不难，难得是如何只进行一次深度遍历，解决方法是：

当该往左节点时，那么就不会往右节点了，那么这个时候右节点就可以重新计算最长交错路径，达到一次遍历，两次运用的目的。

# 困难

## LCP 14\_切分数组

用gcd（a, b）> 1来判断太过耗时间，只需求出b的所有质数u[]，看a%u[i] == 0是否满足，满足就表明最大公约数大于1，这是题目设计好了的，就是要你用质数筛。

## 23\_合并k个排序链表

结合归并排序，转化为合并两个排序链表的算法。

## 124\_二叉树的最大路径和



应该划分为：（倒数第三句）

1. root.val + 左子树的最大路径和（因为右子树为负）
2. root.val + 右子树的最大路径和（因为左子树为负）
3. Root.val + 左右最大路径和（左右子树都非负）

## [1373\_二叉搜索子树的最大键值和](https://leetcode-cn.com/problems/maximum-sum-bst-in-binary-tree/)

我的思路：取一个全局变量，从上往下记录最大键值和self.sum，当不满足二叉树的性质时，self.sum清0，从不满足二叉树性质的那个节点所在层从新计算。

阶梯思路：上面的思路错误，我的想法是，从上往下，这样是大错特错的，因为加入底下不是二叉搜索树，那上面更不会是，所以应该采用的是自底向上，这里的思路很巧，返回三个值，以root为根节点的二叉搜索树的路径和，该二叉搜索树的最小值和最大值，这时应熟练处理，当root为None和当root为根节点的二叉树不是二叉搜索树时的情况，将题目条件运用的非常巧妙。

