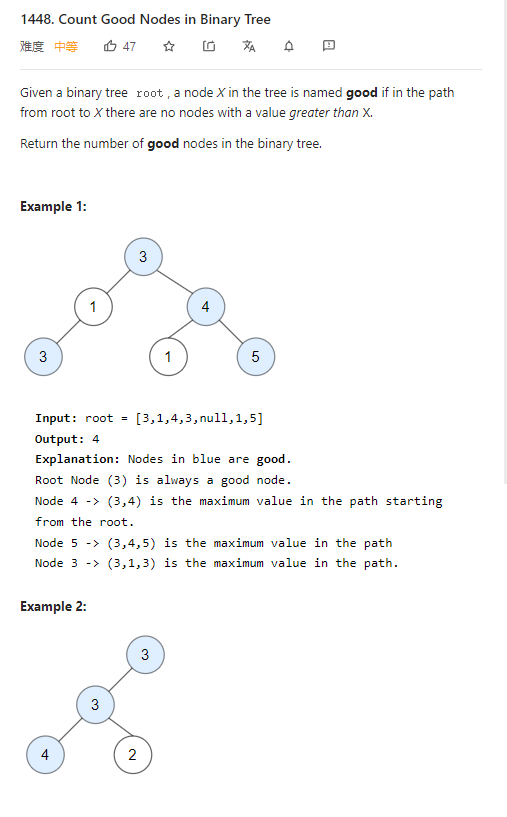
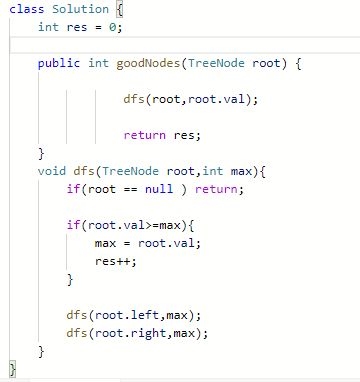
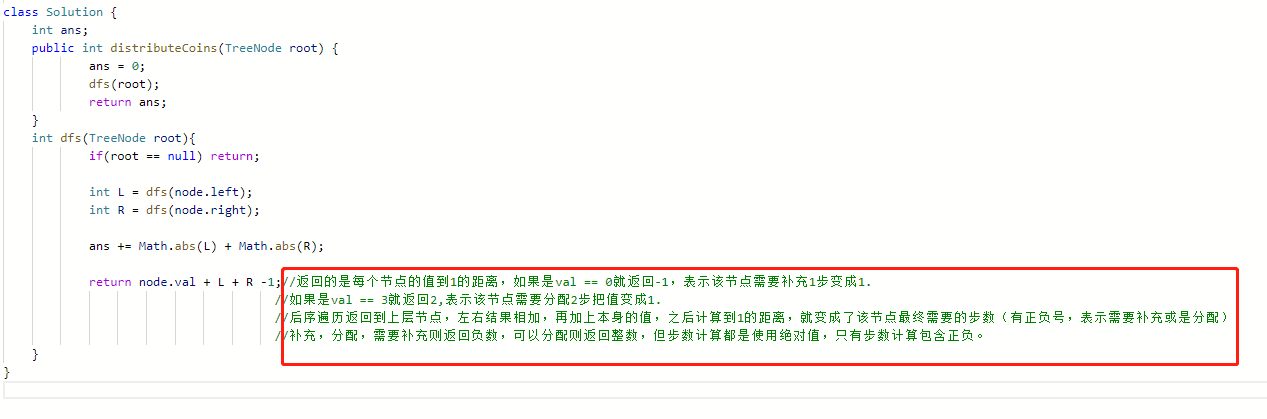
1. 吃亏不要紧，好好体会先序，中序，后序遍历模型的使用:





1. 二叉树分硬币,计算总步数:

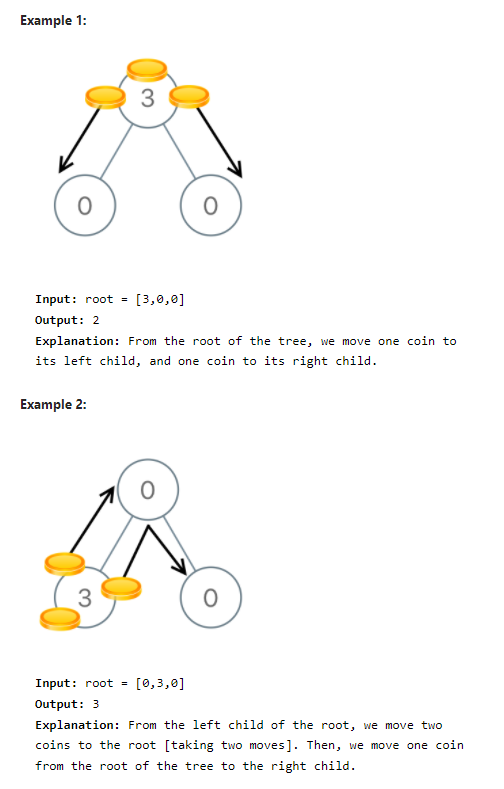
这题主要在于怎么去想，每个节点的val都可以计算需要的步数。



理解后序遍历的很好例子。

看懂红框里面的东西就懂了。

最好是理解。



1. <https://leetcode-cn.com/problems/design-hashmap/solution/she-ji-ha-xi-ying-she-by-leetcode-soluti-klu9/>

Hashmap的实现

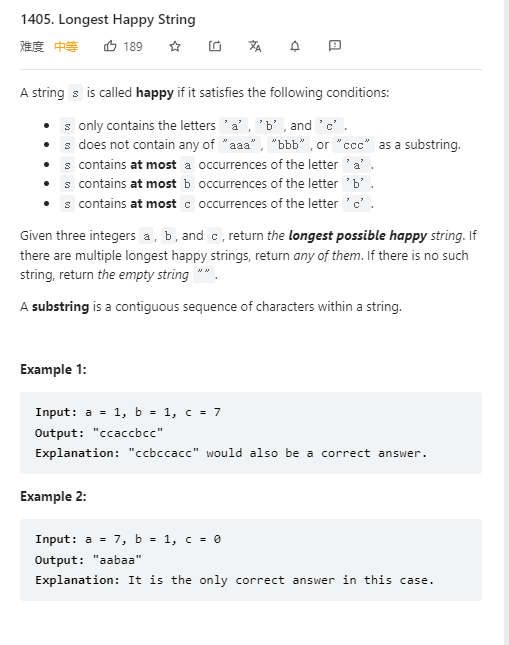
1. 快乐数:

很直观的用次数从大到小使用元素，就可以排出答案，解法也很直接:

技巧就是如何将数字转成字母:

(char)(Xx+’a’) 字母转数字是减，正好相反。

还有就是优先队列的妙用，用数组int[]构造类似hashmap的key-value结构，方便优先队列排序。使用很灵活。

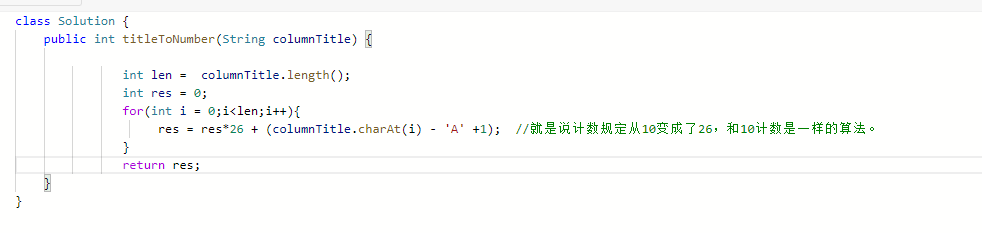




多写几遍肯定会。 984和这题一模一样，甚至少一个字母。

5.26计数制度：（见13的逆推过程）





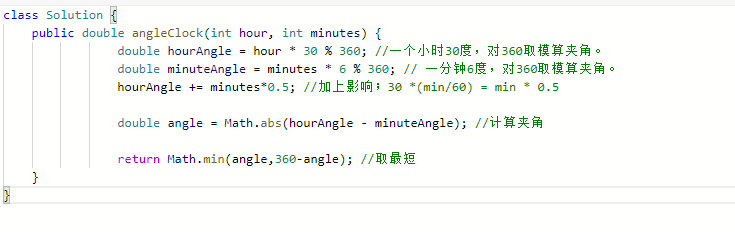
1. 时钟夹角：



很暴力直接的一道题。

计算各自的夹角然后算就完了。

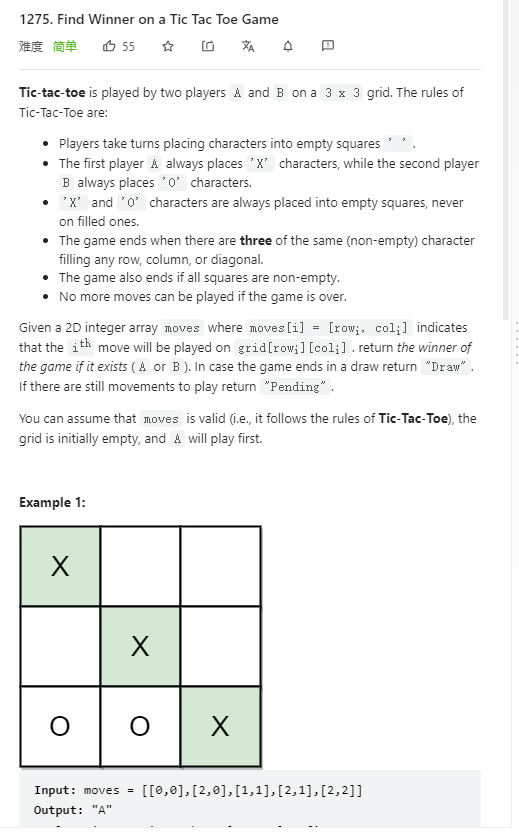
其实就是数学题。

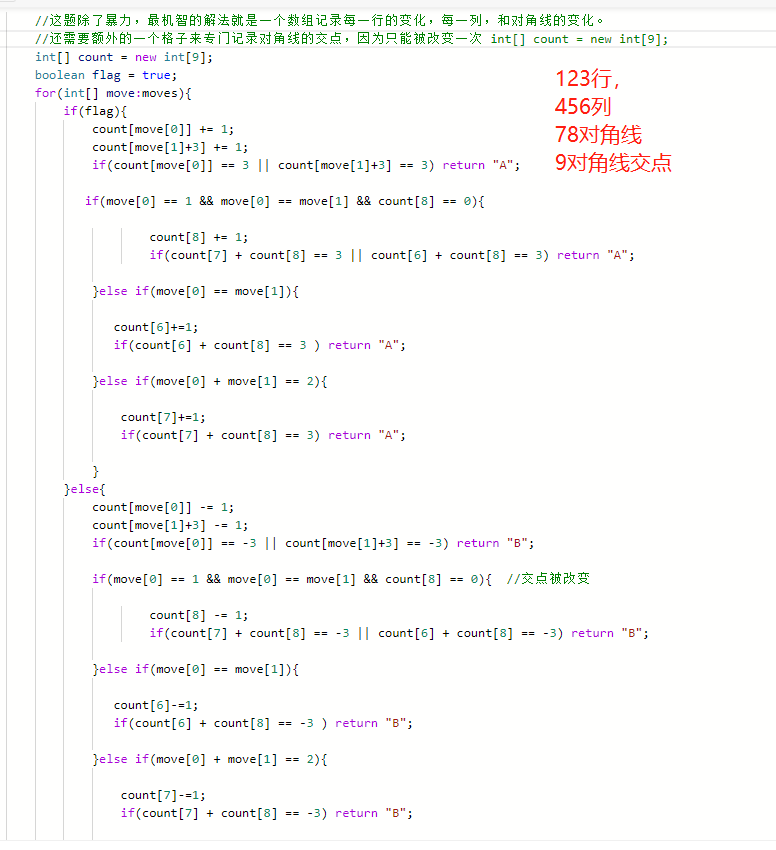


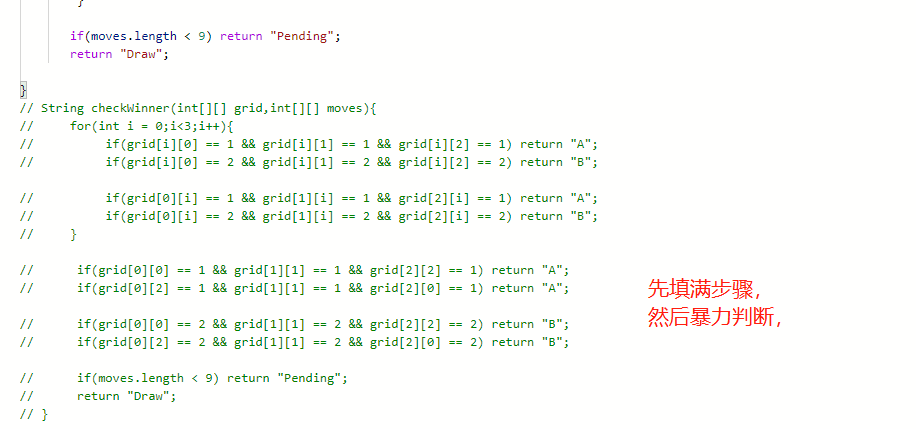
1. 井字棋判输赢

其实是一道很典型的解法，用一个数组来记录行，列，对角线，以及对角线交点的变化。

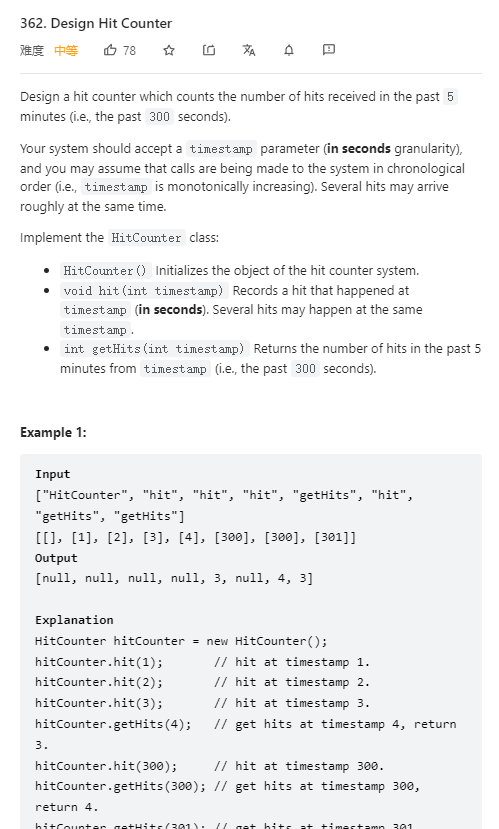
这样做时间复杂度会低一些，但是相比暴力，会略复杂。







1. 前缀和+TreeMap 的api妙用:



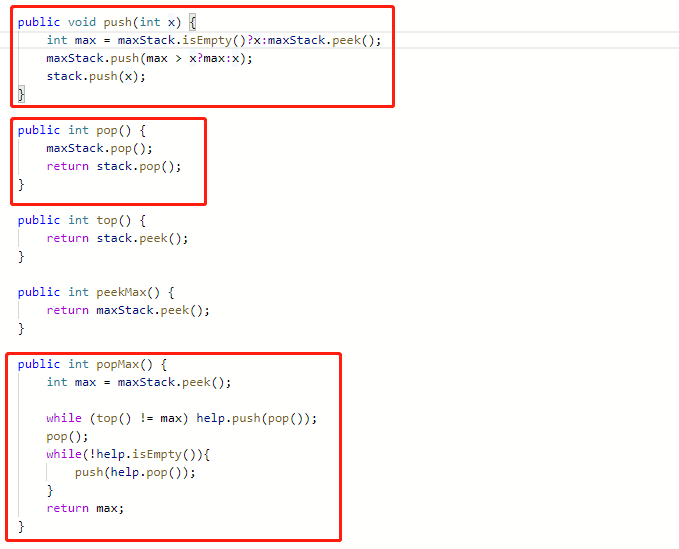


1. 典中典之最大栈:

还是双栈的实现，主要是要记住，这里的双栈push()和pop()是共进退原则！！！

这个最大栈的精髓在于，记录了出栈或者进栈的时候，当前栈的最大值是多少。

是用来记录那个时刻的状态。

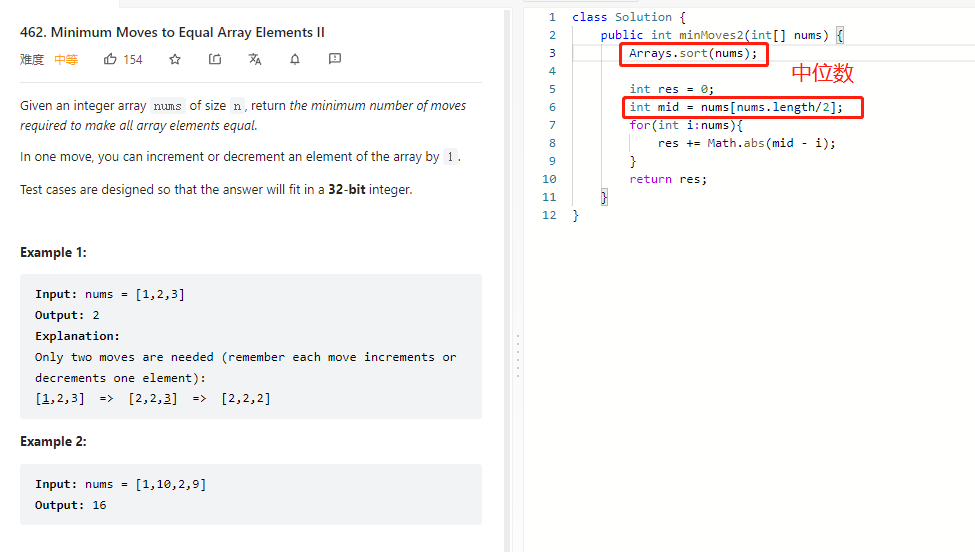


MaxStack只加最大的往上磊。

在后面取得当前栈最大值时也是直接用api共进退即可。

9.最小步骤使数组内全部一致（一眼记概念）。

其实是道数学题，记住用中位数去进行计算，（而不是平均数）就行了。



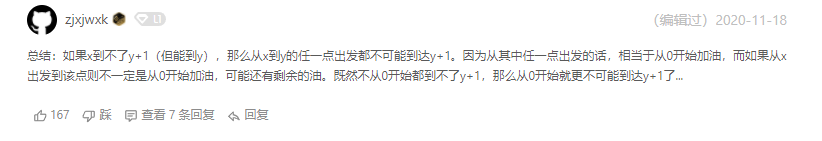
10.加油站推论

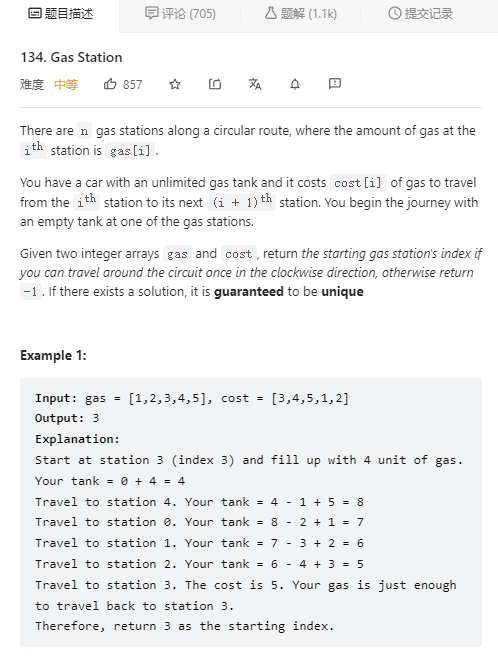
题目如下：加油和cost同时进行，问起点。

这里有个数学推论，如果x作为起点到y可以，但y+1不行的话，

那么x到y中间所有的点都不能到y+1.

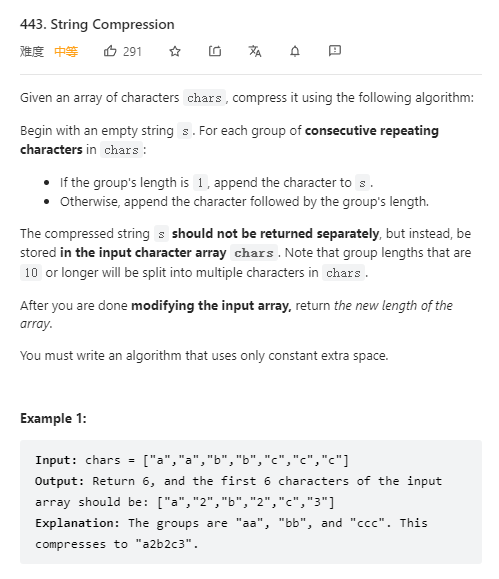
也就是可以忽略掉x-y的所有点，直接用y+1作为新尝试点。







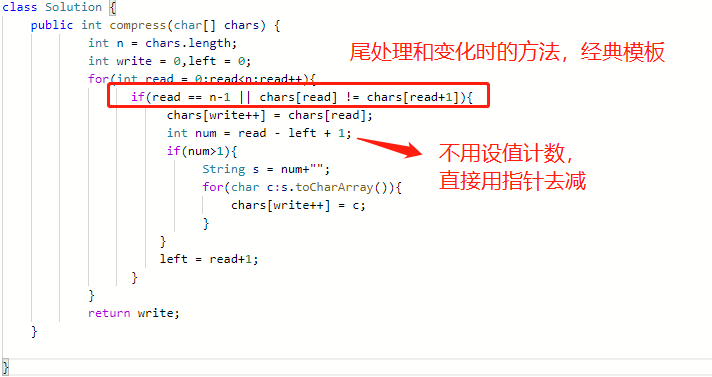
1. 连续相同字母的计数方式:



这题是纯模拟题，关键在于几个指针（write，left，read）的用法。

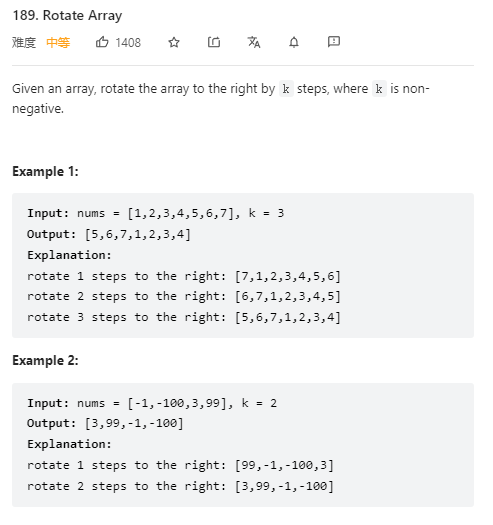
对于相同连续字母计数，

原来是设值计数的，其实根本不用，直接拿指针去减。



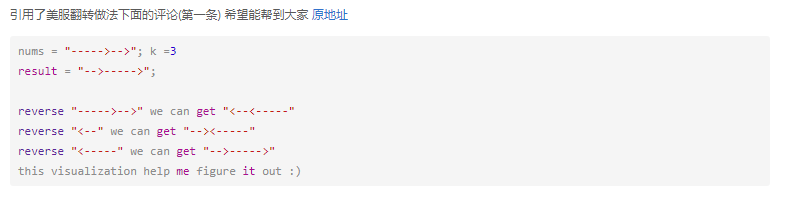
12.

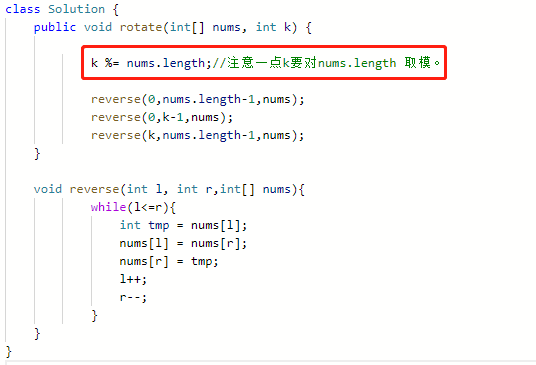
反转数组：



很经典的题，解法多种，除了新开数组覆盖，

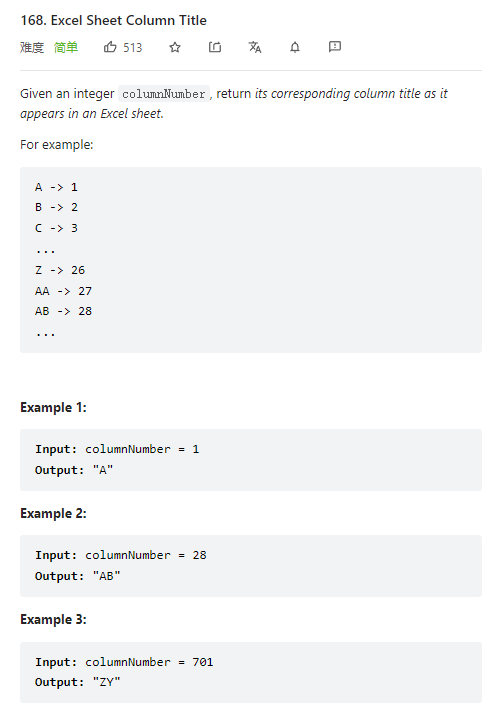
还有一种很经典的原地翻转的解法:

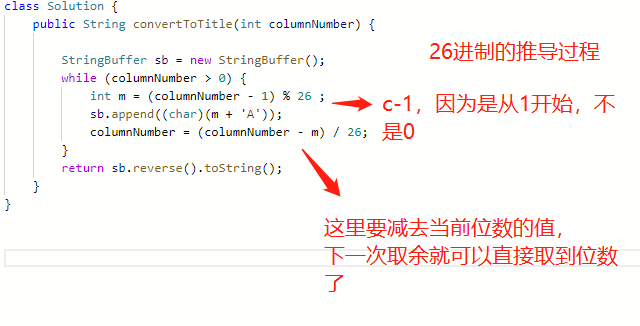




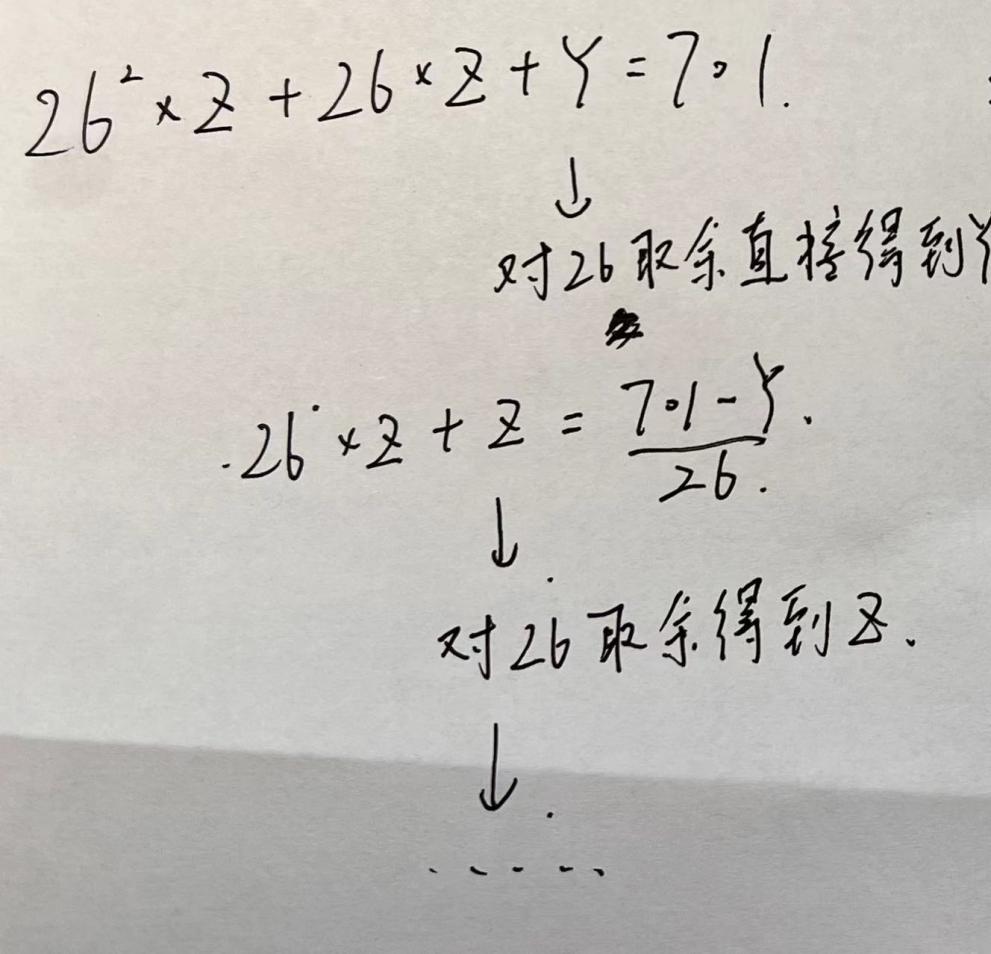
k取模，防止走多圈重复。

13.（见5，正推）

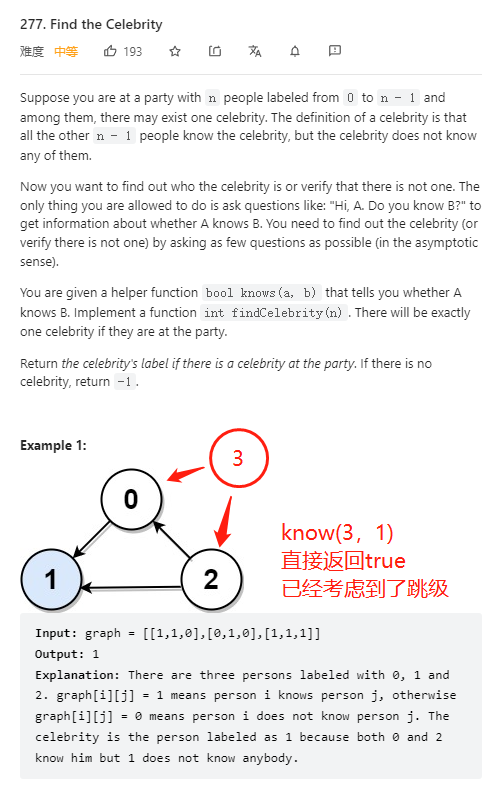




也可以不-1，但后面如果m = 0的话，要改成m = 26.(当前为Z)



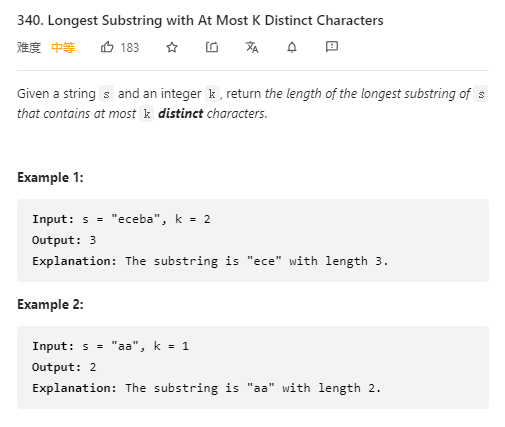
1. 图的塔顶存在判断:



塔顶:1就是塔顶，没有任何出口，所有元素都可以到达1.



1. 经典滑动窗口:



答案随着循环边走边变。

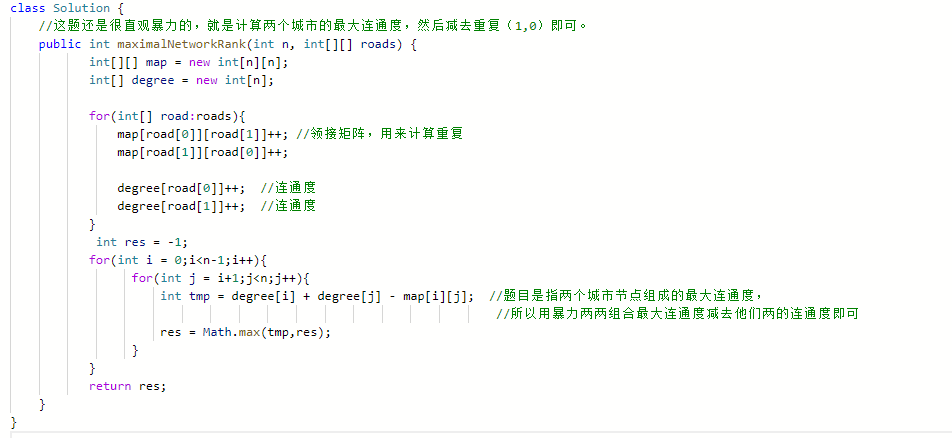


1. 图的最大连接度数题，领接矩阵：



这种算图的连接数题，一般都用的领接矩阵+度数计算求解。

还是很直观暴力的。



稍微记一下题型即可，这类题的通用做法就是这样，算度数。