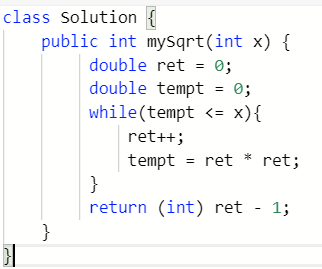
Binary Search 二分查找

1.

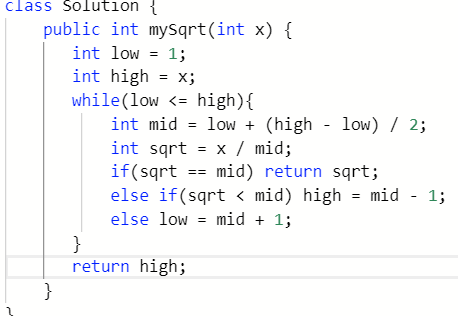


最好加一下x小于等于1直接返回x的情况

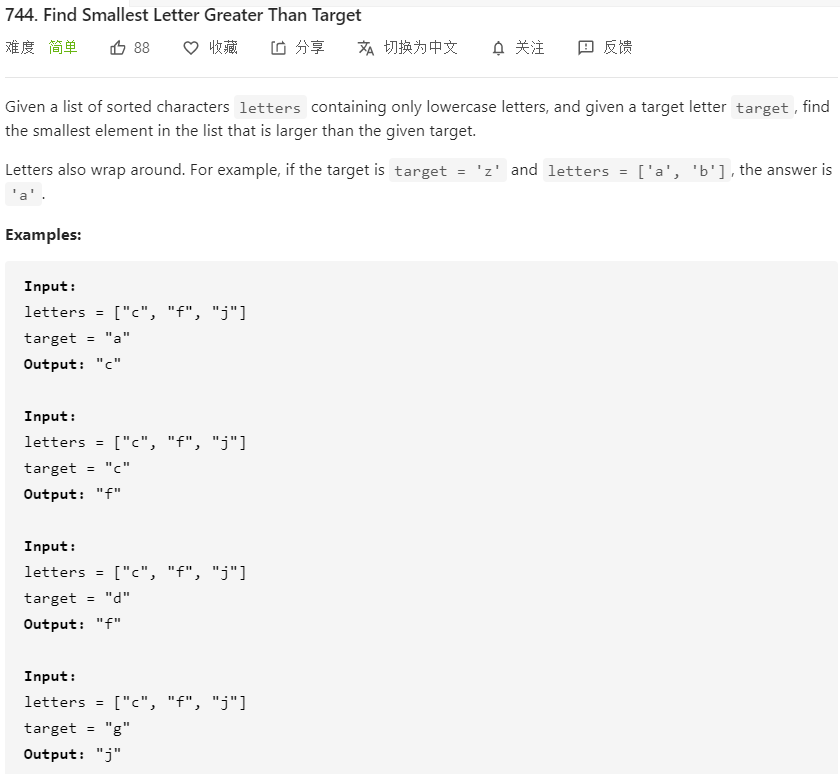
解法1：用while loop每次用一个数去不停平方返回的值直到等于大于那个数，while loop里每平方一次则返回值+1，之后那个数再等于返回值的平方



解法2：二分查找----从1到x，每次都求出一个中间数，再去看x/这个数是不是就等于这个中间数，是的话这个中间数直接返回，因为就是平方根，不然的话向左或向右去找这个数。因为最后low会超过high，所以返回high，high比low少1. EG：8去开平方，最后low是3 high是2

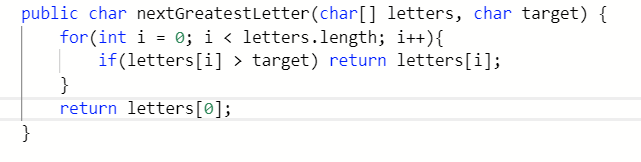


2.

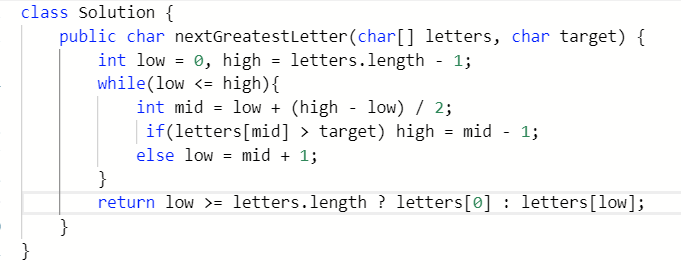


题目：找到比target这个字母比他大的数里最小的那个，如果target大于所有字母则返回list里最小的那个

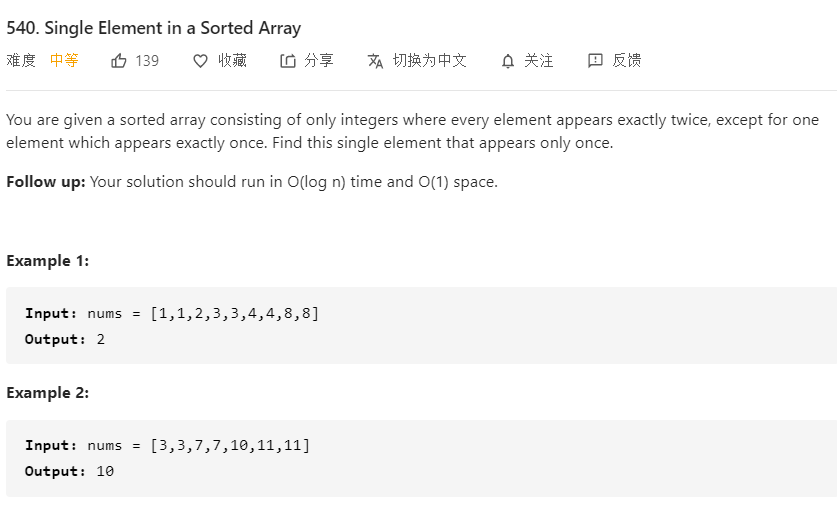
解法1：暴力遍历 时间O(N)



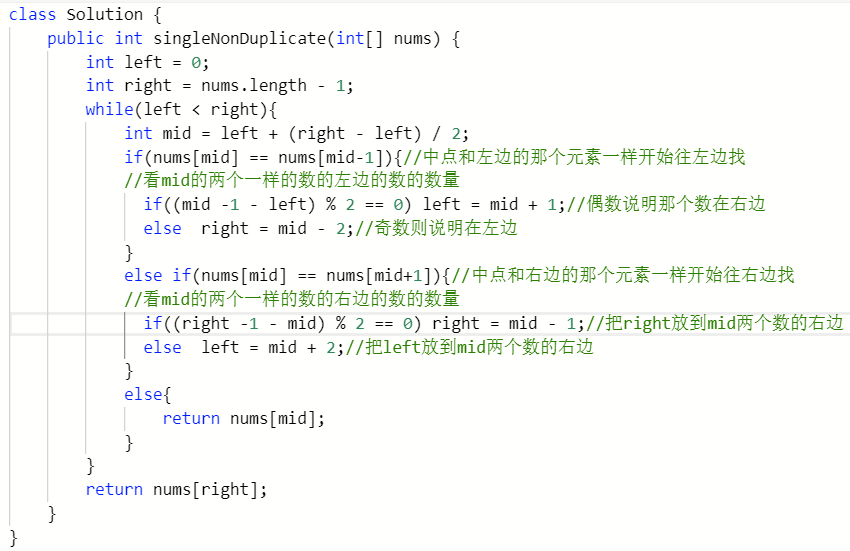
解法2：二分法，每次用mid这个index去比较list里这个index的字母比target大还是小，如果最后low大于等于这个list的list说明target比所有元素都大，则返回第一个



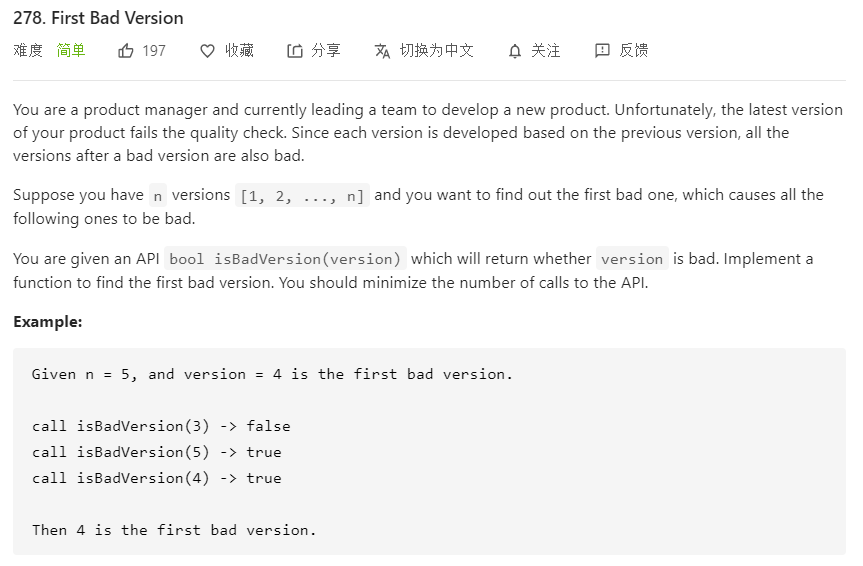
3.



解法：先从中间开始找，去看中间这个元素和左边还是和右边相同，比如如果和左边的那个元素相同，则去看左边的所有元素个数是偶数还是奇数，如果是奇数说明里面存在唯一不重复的数，则right = mid-1开始再左边找，如果是偶数说明那个数在右边那堆元素里，left = mid+2。如果中间元素和他右边那个一样则开始往右边找。如果两边都是偶数说明中间这个数就是唯一不重复的数。

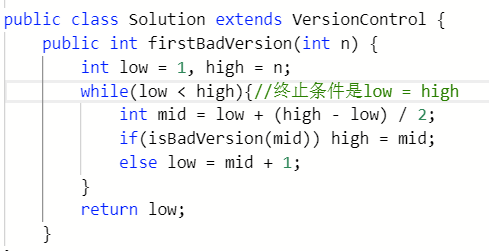


4.

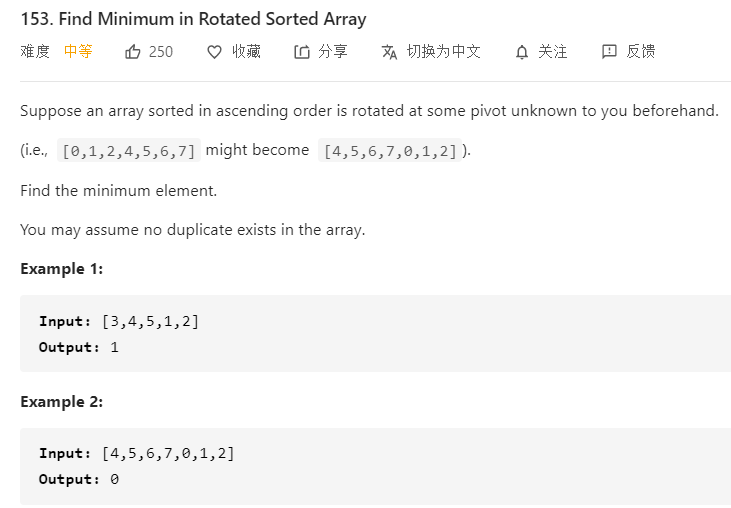


call 1-n 可能从4开始badversion则后面都是badversion

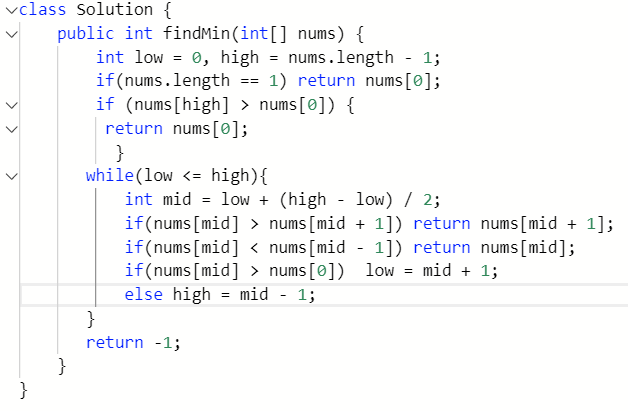
解法：用mid去看如果是badversion说明bad开始于这个之前，把high=mid，否则bad在mid之后，把left = mid+1



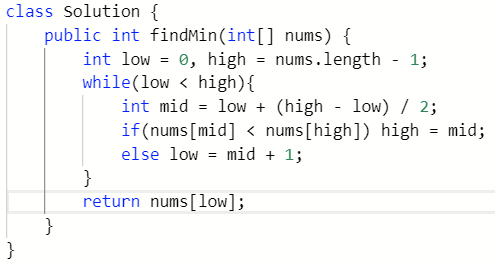
5.



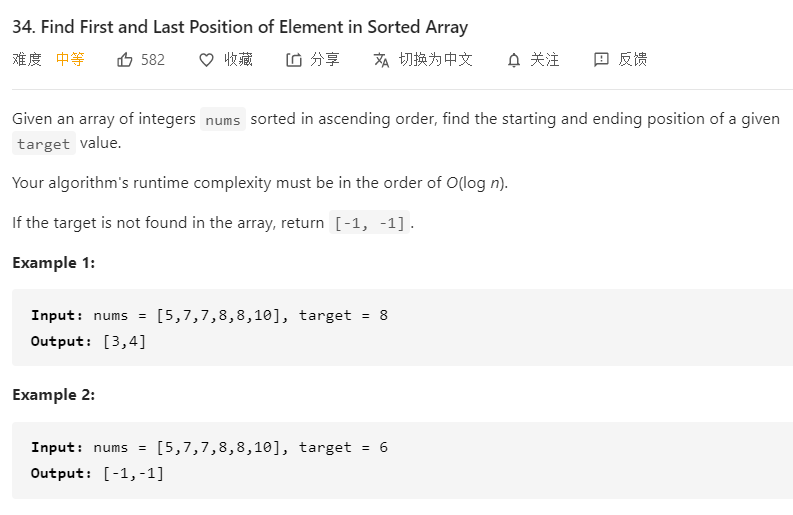
解法1：每次去找mid，如果mid的左边的数比mid大说明mid就是这个数组最小的，因为从mid开始变小，如果mid比右边数大说明mid+1是最小的数。但需要考虑数组只有一个数和数字没被rotate的情况



解法2：二分----mid小于等于high，说明mid到high都是从小到大的，则把high=mid，否则把low=mid + 1 最后返回low位置的值



6.



解法：普通二分查找 如果mid是target，则开始左右收缩，直到左右分别等于一个target然后直接返回

