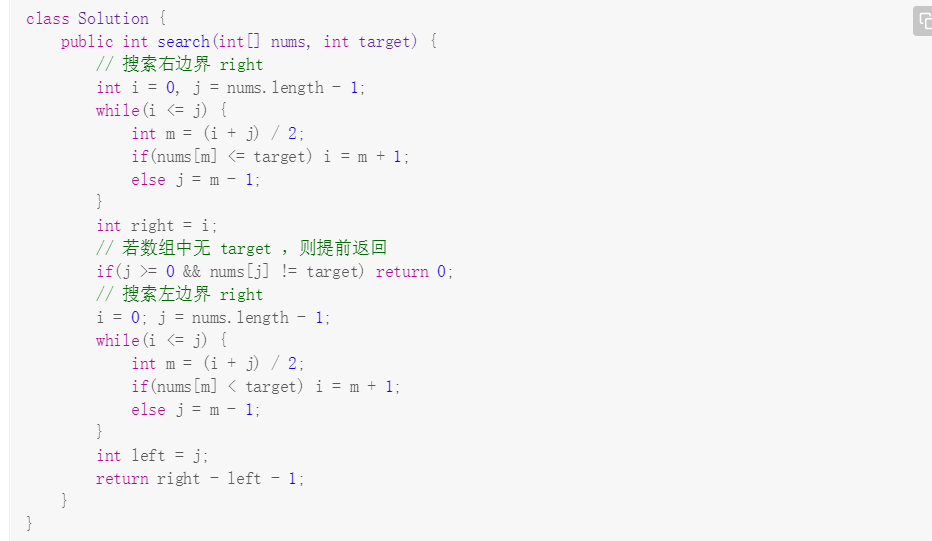
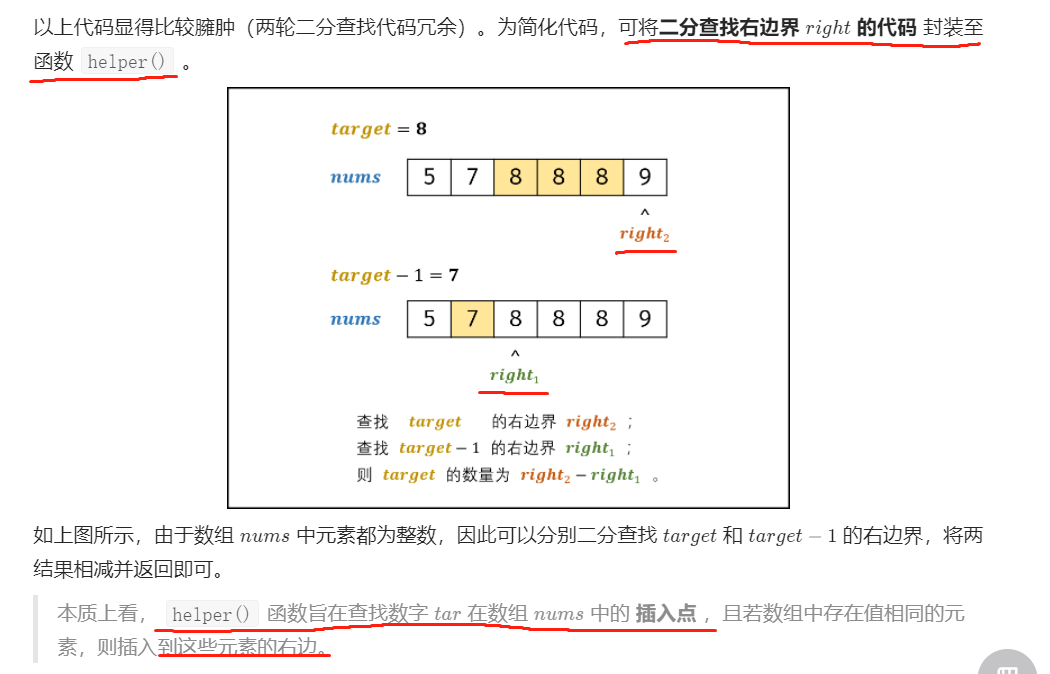
# 第一题



## 代码--常规二分



## 二分查找--插入点



这个封装函数很好,收集到util二分查找常用函数中

## 代码--查找插入点(helper方法)



注:也可以用普通二分法,加上一些判断条件即可,并不难!

具体分析此二分函数(找插入点):

1. while终止条件:

跳出while时lo=hi+1 对应逻辑:

lo指向”**第一个比target大的位置**”,若有多个值为target的元素,则lo指向这些元素的右边

hi指向”???”

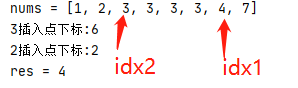
1. 区间缩小:

nums[mid] <=target, (包括等于),所找位置在区间[mid+1,hi]中,执行lo=mid+1

nums[mid] >target,所找位置在区间[lo,mid-1],执行 hi=mid-1

1. 细节,如果插入点在最后一个(index=len),即附加在原数组最后一位,逻辑可行!

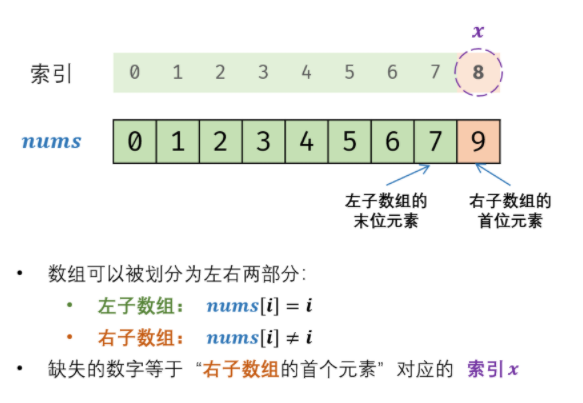
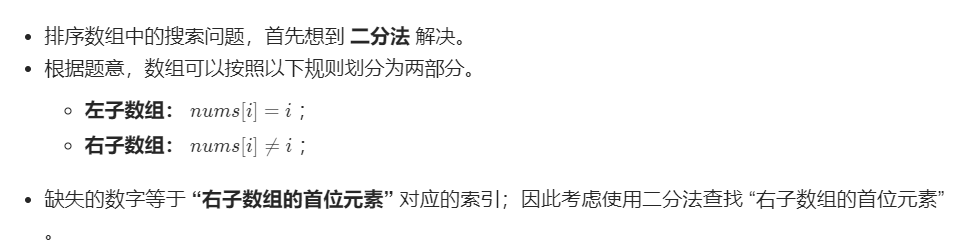
执行结果:

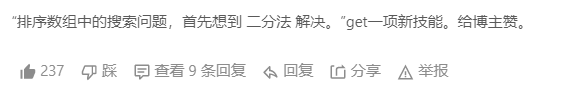
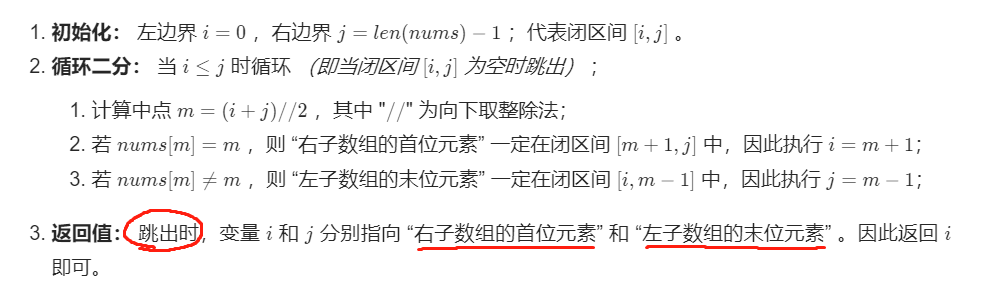
# 第二题

## 题目分析

**目标--右子数组的首位元素!!!**



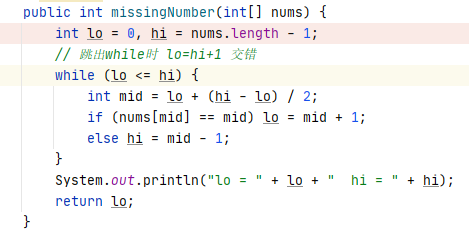
## 算法流程



## 代码

代码细节还是需要注意

二分法就是挺繁琐的-----加一减一, 跳出while的条件



## 总结

总结代码的细节

1. while的终止条件, 此处为 lo<=hi ,即跳出while时 **lo=hi+1**

对应的逻辑:lo指向”右子数组首位” hi指向”左子数组末位”,二者满足lo=hi+1

1. 区间缩小的写法:

事实上区间缩小的**变数是很多**的, 比如缩小到mid-1 / mid / mid+1 (判断mid的归属)

此处依旧要对应题目的逻辑,看**算法流程**图:

* nums[mid]==mid, mid属于左子数组,lo(右子数组首位)在区间[mid+1,hi]中,执行 lo=mid+1
* nums[mid]!=mid, mid属于右子数组,hi(左子数组末位)在区间[lo,mid-1]中,执行hi=mid-1

乍一看并没有处理mid,但其实是**符合逻辑**的!