# 53. 数字在排序数组中出现的次数

## 题目链接

[牛客网](https://www.nowcoder.com/practice/70610bf967994b22bb1c26f9ae901fa2?tpId=13&tqId=11190&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking&from=cyc_github)

## 题目描述

Input:
nums = 1, 2, 3, 3, 3, 3, 4, 6
K = 3

Output:
4

## 解题思路

只要能找出给定的数字 k 在有序数组第一个位置和最后一个位置，就能知道该数字出现的次数。

先考虑如何实现寻找数字在有序数组的第一个位置。正常的二分查找如下，在查找到给定元素 k 之后，立即返回当前索引下标。

public int binarySearch(int[] nums, int K) {
 int l = 0, h = nums.length - 1;
 while (l <= h) {
 int m = l + (h - l) / 2;
 if (nums[m] == K) {
 return m;
 } else if (nums[m] > K) {
 h = m - 1;
 } else {
 l = m + 1;
 }
 }
 return -1;
}

但是在查找第一个位置时，找到元素之后应该继续往前找。也就是当 nums[m]>=k 时，在左区间继续查找，左区间应该包含 m 位置。

private int binarySearch(int[] nums, int K) {
 int l = 0, h = nums.length;
 while (l < h) {
 int m = l + (h - l) / 2;
 if (nums[m] >= K)
 h = m;
 else
 l = m + 1;
 }
 return l;
}

查找最后一个位置可以转换成寻找 k+1 的第一个位置，并再往前移动一个位置。

public int GetNumberOfK(int[] nums, int K) {
 int first = binarySearch(nums, K);
 int last = binarySearch(nums, K + 1);
 return (first == nums.length || nums[first] != K) ? 0 : last - first;
}

需要注意以上实现的查找第一个位置的 binarySearch 方法，h 的初始值为 nums.length，而不是 nums.length - 1。先看以下示例：

nums = [2,2], k = 2

如果 h 的取值为 nums.length - 1，那么在查找最后一个位置时，binarySearch(nums, k + 1) - 1 = 1 - 1 = 0。这是因为 binarySearch 只会返回 [0, nums.length - 1] 范围的值，对于 binarySearch([2,2], 3) ，我们希望返回 3 插入 nums 中的位置，也就是数组最后一个位置再往后一个位置，即 nums.length。所以我们需要将 h 取值为 nums.length，从而使得 binarySearch 返回的区间更大，能够覆盖 k 大于 nums 最后一个元素的情况。