LeetCode 第 34 号问题：在排序数组中查找元素的第一个和最后一个位置

题目来源于 LeetCode 上第 34 号问题：find-first-and-last-position-of-element-in-sorted-array。题目难度为 中等。

题目描述

给定一个按照升序排列的整数数组 **nums**，和一个目标值 **target**。找出给定目标值在数组中的开始位置和结束位置。

你的算法时间复杂度必须是 **O(log n)** 级别。

如果数组中不存在目标值，返回 [-1, -1]。

示例:

```
输入: nums = [5,7,7,8,8,10], target = 8
输出: [3,4]
```

```
输入: nums = [5,7,7,8,8,10], target = 6
输出: [-1,-1]
```

题目解析

题目中要求了时间复杂度为O(log n),这就很清楚要使用二分查找法了。

首先定义两个指针变量，分别存储左右两个位置的索引。首先去找目标值的最左面的索引,通过循环为了防止元素丢失,每次保留最右面的元素,左侧的指针移动时+1。在循环结束的时候判断一下数组中是否包括目标值,不包括的话直接退出。 右面的跟左侧相同，只不过正好相反。

动画描述

代码实现

```
// 34. 下一个排列
// https://leetcode-cn.com/problems/find-first-and-last-position-of-element-in-sorted-array/
// 时间复杂度：O(n)
// 空间复杂度：O(1)
class Solution {
    public int[] searchRange(int[] nums, int target) {
        int[] res = new int[] { -1, -1 };
        int left = 0;
        int right = nums.length - 1;
        int l = left;
        int r = right;
        while (left < right) {
            int mid = (left + right) / 2;
            if (nums[mid] < target) {
                left = mid + 1;
```

```
            } else {
                right = mid;
            }
        }
        if (left>right||nums[left]!=target) {
            return new int[]{-1,-1};
        }
        while (l < r) {
            int mid = (l + r) / 2 + 1;
            if (nums[mid] > target) {
                r = mid - 1;
            } else {
                l = mid;
            }
        }
        if (left > right || left > r) {
            return new int[] { -1, -1 };
        } else {
            return new int[] { left, r };
        }
    }
}
```