34 在排列数组中查找元素的第一个和最后一个位置

Label：二分法、数组
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。
如果数组中不存在目标值 target，返回 [-1, -1]。

进阶：你可以设计并实现时间复杂度为 O(log n) 的算法解决此问题吗？

输入：nums = [5,7,7,8,8,10], target = 8
输出：[3,4]
输入：nums = [], target = 0
输出：[-1,-1]

- 遍历

```
class Solution {
    public int[] searchRange(int[] nums, int target) {
        if (nums.length == 0) return new int[]{-1, -1};
        else if (nums.length == 1) {
            return nums[0] == target ?  new int[]{0, 0} : new int[]{-1,-1};
        }

        int[] re = new int[2];
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] == target) {
                re[0] = i;
                while (i < nums.length - 1  && nums[i] == target) {
                    i++;
                }
                re[1] = nums[i] == target ? i : --i;
                return re;
            } else if (nums[i] > target) {
                return new int[]{-1,-1}; // 提前剪枝
            }
        }
        return new int[]{-1,-1};
    }
}
```

- 二分法

```
class Solution {
    public int[] searchRange(int[] nums, int target) {
        int leftIdx = binarySearch(nums, target, true);
        int rightIdx = binarySearch(nums, target, false) - 1;
        if (leftIdx <= rightIdx && rightIdx < nums.length && nums[leftIdx] ==
target && nums[rightIdx] == target) {
            return new int[]{leftIdx, rightIdx};
        }
        return new int[]{-1, -1};
    }

    public int binarySearch(int[] nums, int target, boolean lower) {
        int left = 0, right = nums.length - 1, ans = nums.length;
        while (left <= right) {
            int mid = (left + right) / 2;
            if (nums[mid] > target || (lower && nums[mid] >= target)) {
                right = mid - 1;
                ans = mid;
            } else {
                left = mid + 1;
            }
        }
        return ans;
    }
}
```