题目地址（35. 搜索插入位置）

https://leetcode-cn.com/problems/search-insert-position

入选理由

1. 今天是二分，双指针中的一种类型。 专题篇会单独对二分进行详细整理，包教包会的那种 ^_^

2. 难度是简单，大家一定要打卡哦

标签

- 双指针
- 二分

难度

- 简单

题目描述

给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，返回它将会被按顺序插入的位置。

你可以假设数组中无重复元素。

```
示例 1:

输入: [1,3,5,6], 5
输出: 2
示例 2:

输入: [1,3,5,6], 2
输出: 1
示例 3:

输入: [1,3,5,6], 7
输出: 4
示例 4:
```

```
输入: [1,3,5,6], 0
输出: 0
```

暴力法

思路

一次遍历枚举数组，寻找第一个大于等于 target 的值即可。

代码

代码支持：JS

```
var searchInsert = function (nums, target) {
  for (let i = 0; i < nums.length; i++) {
    if (nums[i] >= target) {
      return i;
    }
  }
  return nums.length;
};
```

复杂度分析

令 n 为数组长度

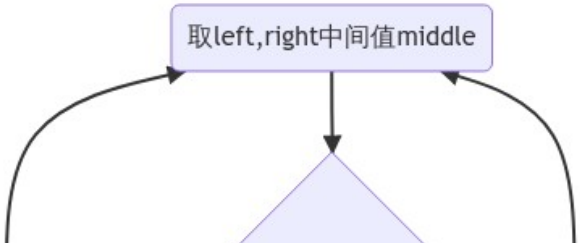
- 时间复杂度: $O(n)$
- 空间复杂度: $O(1)$

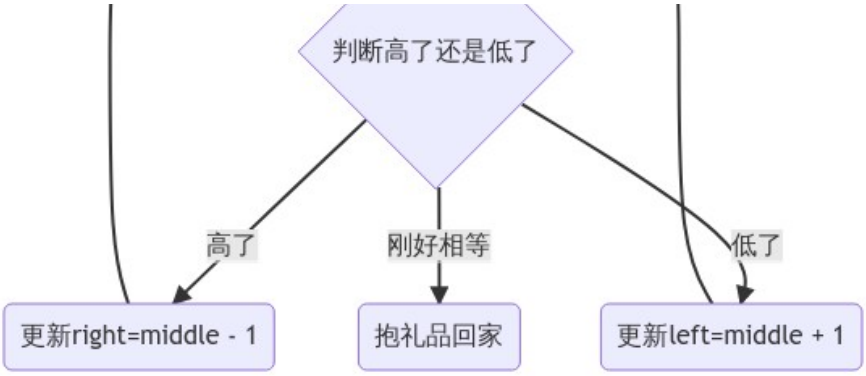
二分

思路

举个例子，不知道看官们有没有看过这样一档电视节目，给一个商品，找观众去猜价格，如果猜中了，商品就作为礼品给观众，观众猜一个价格后，主持人会告诉你高了还是低了。 如果你对这个商品的价值一无所知，你会怎么做?

我们会给一个中间值，通过主持人给的信息更新中间值，更新流程如下





这就是典型的二分法，可以直接套讲义模板

代码

代码支持：Java，JS，Python，CPP

Java Code:

```
class Solution {
    public int searchInsert(int[] nums, int target) {
        if (nums.length == 0) return -1;

        int left = 0;
        int right = nums.length - 1;

        while (left <= right) {

            int mid = left + (right - left) / 2;
            if (target > nums[mid]) {
                left = mid + 1;
            } else if (target < nums[mid]) {
                right = mid - 1;
            } else if (target == nums[mid]){
                return mid;
            }
        }

        return left;
    }
}
```

JS Code:

```
var searchInsert = function (nums, target) {
  let left = 0,
    right = nums.length - 1;
```

```
  while (left <= right) {
    const middle = (left + right) >> 1;
    const middleValue = nums[middle];
    if (middleValue === target) {
      return middle;
    } else if (middleValue < target) {
      left = middle + 1;
    } else {
      right = middle - 1;
    }
  }
  return left;
};
```

Python Code:

```
class Solution:
    def searchInsert(self, nums: List[int], target: int) -> int:
        l = 0
        r = len(nums) - 1

        while(l<=r):
            mid = (l+r)//2
            if nums[mid] == target:
                return mid
            if nums[mid] < target:
                l = mid + 1
            else:
                r = mid - 1
        return l
```

CPP Code:

```
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int l = 0, r = nums.size();

        while (l <= r) {
            int mid = l + (r - l)/2;
            if (nums[mid] == target) return mid;

            if (nums[mid] < target) l = mid + 1;
            else
                r = mid - 1;
```

```
        }

        return l;
    }
};
```

复杂度分析

令 n 为数组长度

- 时间复杂度: $O(logn)$
- 空间复杂度: $O(1)$

上一页 下一页