0674. 最长连续递增序列

ITCharge 大约 4 分钟

- 标签：数组
- 难度：简单

题目链接

- 0674. 最长连续递增序列 - 力扣

题目大意

描述：给定一个未经排序的数组 $nums$。

要求：找到最长且连续递增的子序列，并返回该序列的长度。

说明：

- **连续递增的子序列**：可以由两个下　　和 r（$l < r$）确定，如果对于每个 $l \leq i < r$，都有 $nums[i] < nums[i + 1] $，那么子序列 $[nums[l], nums[l + 1], ..., nums[r - 1], nums[r]]$ 就是连续递增子序列。
- $1 \leq nums.length \leq 10^4$。
- $-10^9 \leq nums[i] \leq 10^9$。

示例：

- 示例 1：

```py
输入：nums = [1,3,5,4,7]
输出：3
解释：最长连续递增序列是 [1,3,5]，长度为 3。尽管 [1,3,5,7] 也是升序的子序列，但它不是连续的，因为 5 和 7 在原数组里被 4 隔开。
```

- 示例 2：

```py
输入：nums = [2,2,2,2,2]
输出：1
解释：最长连续递增序列是 [2], 长度为 1。
```

解题思路

思路 1：动态规划

1. 定义状态

定义状态 $dp[i]$ 表示为：以 $nums[i]$ 结尾的最长且连续递增的子序列长度。

2. 状态转移方程

因为求解的是连续子序列，所以只需要考察相邻元素的状态转移方程。

如果一个较小的数右侧相邻元素为一个较大的数，则会形成一个更长的递增子序列。

对于相邻的数组元素 $nums[i-1]$ 和 $ıs[i]$ 来说：

- 如果 $nums[i-1] < nums[i]$，则 $nums[i]$ 可以接在 $nums[i-1]$ 后面，此时以 $nums[i]$ 结尾的最长递增子序列长度会在「以 $nums[i-1]$ 结尾的最长递增子序列长度」的基础上加 1，即 $dp[i] = dp[i-1] + 1$。
- 如果 $nums[i-1] >= nums[i]$，则 $nums[i]$ 不可以接在 $nums[i-1]$ 后面，可以直接跳过。

综上，我们的状态转移方程为：$dp[i] = dp[i-1] + 1$，$nums[i-1] < nums[i]$。

3. 初始条件

默认状态下，把数组中的每个元素都作为长度为 1 的最长且连续递增的子序列长度。即 $dp[i] = 1$。

4. 最终结果

根据我们之前定义的状态，$dp[i]$ 表示为：以 $nums[i]$ 结尾的最长且连续递增的子序列长度。则为了计算出最大值，则需要再遍历一遍 dp 数组，求出最大值即为最终结果。

思路 1：动态规划代码

```py
class Solution:
    def findLengthOfLCIS(self, nums: List[int]) -> int:
        size = len(nums)
        dp = [1 for _ in range(size)]

        for i in range(1, size):
            if nums[i - 1] < nums[i]:
                dp[i] = dp[i - 1] + 1

        return max(dp)
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$。一重循环遍历的时间复杂度为 $O(n)$，最后求最大值的时间复杂度是 $O(n)$，所以总体时间复杂度为 $O(n)$。
- **空间复杂度**：$O(n)$。用到了一维数组存储状态，所以总体空间复杂度为 $O(n)$。

思路 2：滑动窗口（不定长度）

1. 设定两个指针：$left$、$right$，分别指向滑动窗口的左右边界，保证窗口内为连续递增序列。使用 $window\_len$ 存储当前窗口大小，使用 max_len 维护最大窗口长度。
2. 一开始，$left$、$right$ 都指向 0。
3. 将最右侧元素 $nums[right]$ 加入当前连续递增序列中，即当前窗口长度加 1（`window_len += 1`）。
4. 判断当前元素 $nums[right]$ 是否满足连续递增序列。
5. 如果 $right > 0$ 并且 $nums[right - 1] \geq nums[right]$，说明不满足连续递增序列，则将 $left$ 移动到窗口最右侧，重置当前窗口长度为 1（`window_len = 1`）。
6. 记录当前连续递增序列的长度，并更新最长连续递增序列的长度。
7. 继续右移 $right$，直到 $right \geq len(nums)$ 结束。
8. 输出最长连续递增序列的长度 max_len。

思路 2：代码

```py
class Solution:
    def findLengthOfLCIS(self, nums: List[int]) -> int:
        size = len(nums)
        left, right = 0, 0
        window_len = 0
        max_len = 0

        while right < size:
            window_len += 1

            if right > 0 and nums[right - 1] >= nums[right]:
                left = right
                window_len = 1

            max_len = max(max_len, window_len)
            right += 1

        return max_len
```

思路 2：复杂度分析

- **时间复杂度**：$O(n)$。
- **空间复杂度**：$O(1)$。