题目地址(673. 最长递增子序列的个数)

https://leetcode-cn.com/problems/number-of-longest-increasing-subsequence/

入选理由

1. 这是 DP 问题的另一个经典类型 - LIS（最长上升子序列），而今天这个题目是这个系列最难的之一。 如果这个不会，可以先看下我的 LIS 专题（自己搜吧）。

标签

- 动态规划

难度

- 中等

题目描述

给定一个未排序的整数数组，找到最长递增子序列的个数。

示例 1:

输入: [1,3,5,4,7]
输出: 2
解释: 有两个最长递增子序列，分别是 [1, 3, 4, 7] 和[1, 3, 5, 7]。
示例 2:

输入: [2,2,2,2,2]
输出: 5
解释: 最长递增子序列的长度是1，并且存在5个子序列的长度为1，因此输出5。
注意: 给定的数组长度不超过 2000 并且结果一定是32位有符号整数。

前置知识

- 动态规划

公司

- 暂无

思路

这道题其实就是**最长上升子序列（LIS）**的变种题。如果对 LIS 不了解的可以先看下我之前写的一篇文章**穿上衣服我就不认识你了？来聊聊最长上升子序列**[1]，里面将这种题目的套路讲得很清楚了。

回到这道题。题目让我们求最长递增子序列的个数，而不是通常的**最长递增子序列的长度**。 因此我想到使用另外一个变量记录**最长递增子序列的个数**信息即可。类似的套路有**股票问题**，这种问题的套路在于只是单独存储一个状态以无法满足条件，对于这道题来说，我们存储的单一状态就是**最长递增子序列的长度**。那么一个自然的想法是**不存储最长递增子序列的长度，而是仅存储最长递增子序列的个数**可以么？这是不可以的，因为**最长递增子序列的个数** 隐式地条件是你要先找到最长的递增子序列才行。

如何存储两个状态呢？一般有两种方式：

- 二维数组 dp[i][0] 第一个状态 dp[i][1] 第二个状态
- dp1[i] 第一个状态 dp2[i] 第二个状态

使用哪个都可以，空间复杂度也是一样的，使用哪种看你自己。这里我们使用第一种，并且 dp[i][0] 表示 以 nums[i] 结尾的最长上升子序列的长度，dp[i][1] 表示 以 nums[i] 结尾的长度为 dp[i][0] 的子序列的个数。

明确了要多存储一个状态之后，我们来看下状态如何转移。

LIS 的一般过程是这样的：

```
for i in range(n):
    for j in range(i + 1, n):
        if nums[j] > nums[i]:
            # ...
```

这道题也是类似，遍历到 nums[j] 的时候往前遍历所有的 满足 i < j 的 i。

- 如果 nums[j] <= nums[i]， nums[j] 无法和前面任何的序列拼接成**递增子序列**
- 否则说明我们可以拼接。但是拼接与否取决于拼接之后会不会更长。如果更长了就拼，否则不拼。

上面是 LIS 的常规思路，下面我们加一点逻辑。

- 如果拼接后的序列更长，那么 dp[j][1] = dp[i][1] （这点容易忽略）
- 如果拼接之后序列一样长， 那么 dp[j][1] += dp[i][1]。
- 如果拼接之后变短了，则不应该拼接。

关键点解析

- 最长上升子序列问题
- dp[j][1] = dp[i][1] 容易忘记

代码

代码支持: Python

```
class Solution:
    def findNumberOfLIS(self, nums: List[int]) -> int:
        n = len(nums)
        # dp[i][0] ->  LIS
        # dp[i][1] -> NumberOfLIS
        dp = [[1, 1] for i in range(n)]
        ans = [1, 1]
        longest = 1
        for i in range(n):
            for j in range(i + 1, n):
                if nums[j] > nums[i]:
                    if dp[i][0] + 1 > dp[j][0]:
                        dp[j][0] = dp[i][0] + 1
                        # 下面这行代码容易忘记，导致出错
                        dp[j][1] = dp[i][1]
                        longest = max(longest, dp[j][0])
                    elif dp[i][0] + 1 == dp[j][0]:
                        dp[j][1] += dp[i][1]
        return sum(dp[i][1] for i in range(n) if dp[i][0] == longest)
```

复杂度分析

令 N 为数组长度。

- 时间复杂度：$O(N^2)$
- 空间复杂度：$O(N)$

扩展

这道题也可以使用线段树来解决，并且性能更好，不过由于不算是常规解法，因此不再这里展开，感兴趣的同学可以尝试一下。

最后推荐一个类似的题目，都是套路题，而且可以和前面的专题有所联系，大家可以用它来拔拔高
https://github.com/azl397985856/leetcode/blob/master/problems/2008.maximum-earnings-from-taxi.md

更多题解可以访问我的 LeetCode 题解仓库：https://github.com/azl397985856/leetcode 。 目前已经 45K star 啦。

关注公众号力扣加加，努力用清晰直白的语言还原解题思路，并且有大量图解，手把手教你识别套路，高效刷题。

参考资料

[1] 穿上衣服我就不认识你了？来聊聊最长上升子序列: *https://lucifer.ren/blog/2020/06/20/LIS/*

上一页 下一页