673. Number of Longest Increasing Subsequence

Given an integer array `nums`, return *the number of longest increasing subsequences.*

Notice that the sequence has to be **strictly** increasing.

Example 1:

```
Input: nums = [1,3,5,4,7]
Output: 2
Explanation: The two longest increasing subsequences are [1, 3, 4, 7] and
[1, 3, 5, 7].
```

Example 2:

```
Input: nums = [2,2,2,2,2]
Output: 5
Explanation: The length of longest continuous increasing subsequence is 1,
and there are 5 subsequences' length is 1, so output 5.
```

```python
class Solution:
    def findNumberOfLIS(self, nums: List[int]) -> int:
        dp = [[1,1] for i in range(len(nums))]
        dp[0] = [1,1]
        longest = 1
        for i in range(len(nums)):
            temp = 1
            count = 0
            for j in range(i):
                if nums[j]<nums[i]:
                    temp = max(temp,dp[j][0]+1)
            for j in range(i):
                if nums[j]<nums[i] and dp[j][0]==temp-1:
                    count = count+dp[j][1]
            dp[i][0] = temp
            dp[i][1] =  max(count,dp[i][1])
            longest = max(longest,temp)

        return sum([temp[1] for temp in dp if temp[0]==longest])
```