673. Number of Longest Increasing Subsequence

Attempted

Medium | Topics | Companies

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

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

Example 1:

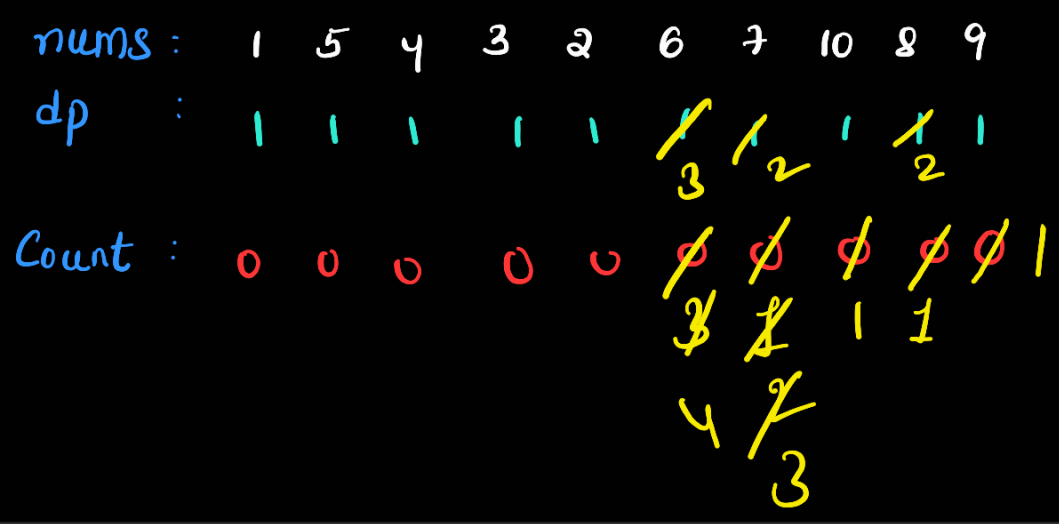
```
Input: nums = [1,3,5,4,7]
Output: 2
Explanation: The two longest increasing subsequences are [1, 3, 4, 7] and [1, 3, 5, 7].
```

Example 2:

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

Constraints:

- `1 <= nums.length <= 2000`
- `-10^6 <= nums[i] <= 10^6`




```cpp
class Solution {
public:
    int findNumberOfLIS(vector<int>& nums) {
        int n = nums.size();
        vector<int> dp(n, 1);
        vector<int> count(n, 1);

        int maxi = 1;
```

```
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < i; j++) {
                if (nums[i] > nums[j]) {
                    if (1 + dp[j] > dp[i]) {
                        dp[i] = 1 + dp[j];
                        count[i] = count[j];
                    } else if (1 + dp[j] == dp[i]) {
                        count[i] = count[i] + count[j];
                    }
                }
            }
            maxi = max(maxi, dp[i]);
        }

        int ans = 0;

        for (int i = 0; i < n; i++) {
            if (dp[i] == maxi)
                ans = ans + count[i];
        }

        return ans;
    }
};
```

$T(n) : O(n^2)$

$S(n) : O(2n)$