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。

思路，最长递增子序列其实一般是按动态规划来做的，假设a1.到an中最长子序列为n，那么新增一个an+1，到an+1的最长序列，为an+1之前的比an+1小的数字的最长序列+1，这时最长子序列可能变为n+1或者保持不变，并且。这是求单最长子序列的思路。该题是需要统计最长子序列路径的个数。那么，我们先统计到该点最长子序列的路径个数，到该点最长子序列路径个数为，个数相等的点的路径个数的和。然后动态统计相加即可得到答案。

代码

```
class Solution {
public:
    int findNumberOfLIS(vector<int>& nums) {
        vector<int> number(nums.size(),1);
        vector<int> rootnumber(nums.size(),1);
        int max=1;
        int num=0;
        for(int i=1;i<nums.size();i++){
            int maxnum=1;
            int rootnum=1;
            for(int j=i-1;j>=0;j--){
                if(nums[i]>nums[j]){
                    if(number[i]+number[j]>maxnum){
                        maxnum=number[i]+number[j];
                        rootnum=rootnumber[j];
                    }else if(number[i]+number[j]==maxnum){
                        rootnum+=rootnumber[j];
                    }
```

```
                }
            }
            number[i]=maxnum;
            rootnumber[i]=rootnum;
            if(maxnum>max){
                max=maxnum;
            }
        }
        for(int i=0;i<nums.size();i++){
            if(number[i]==max){
                num+=rootnumber[i];
            }
        }
        return num;
    }
};
```