https://cp-algorithms.com/combinatorics/binomial-coefficients.html#calculation-in-o1

## PREREQUISITES:

Computing binomial coefficients

## PROBLEM:

Alice picks a random permutation $P$ of size $N$ and a random position $i$ between $1$ and $N$.
Then, as long as $i < N$ and $P_{i+1} < P_i$, she moves to $i+1$.
What is the expected number of positions she will visit?

## EXPLANATION:

There are $N \cdot N!$ possible starting states: choose a permutation, then choose a position.

For each $1 \leq i \leq N$, let $S_i$ denote the number of starting states that result in Alice visiting exactly $i$ positions.
Then, by definition of expected value, the answer is

$\frac{1}{N \cdot N!} \sum_{i=1} i \cdot S_i$

Now let's look at how to compute $S_i$.
Suppose we fix $i$. What is needed to visit exactly $i$ positions?

Well, suppose we also fix our starting position, say $k$. Then, we must have

▾ Case 1

Consider the case when $k+i-1 = N$.
Let's fix what the last $i$ elements are. This can be done in $\binom{N}{i}$ ways.
Now there is exactly one way to arrange them, since they must be in descending order.
Further, the other $N-i$ elements can be arranged in $(N-i)!$ ways.

This gives us a total of $\binom{N}{i}(N-i)!$ ways.

▾ Case 2

Consider the case when $k+i-1 < N$.
First, let's fix the starting position $k$: there are $N-i$ choices for it (it can be anything between $1$ and $N-i$).

With the starting position fixed, suppose we fixed which $i$ elements are in these $i$ positions.
However, how do we ensure that the next element is greater than the minimum of these $i$?

Simple: we simply fix all the $i+1$ elements instead!
That is, choose $i+1$ elements in $\binom{N}{i+1}$ ways. Then, choose which one of them is the last element: there are $i$ choices, since we cannot choose the minimum of the chosen $i+1$ but anything else is ok.
Now, there is only one way to arrange the remaining $i$ elements: descending order.

Finally, the unchosen $(N-i-1)$ elements can be arranged in $(N-i-1)!$ ways.

So, the number of ways in this case is $(N-i) \cdot \binom{N}{i+1} \cdot i \cdot (N-i-1)! = \binom{N}{i+1} \cdot i \cdot (N-i)!$.

Putting these together, we find

$$S_i = (N-i)! \times \left(\binom{N}{i} + \binom{N}{i+1} \cdot i\right)$$

Each term here is either a factorial or a binomial coefficient. As linked in the prerequisites section, they can all be computed in $\mathcal{O}(1)$ if factorials are precomputed.

So, each $S_i$ can be computed in $\mathcal{O}(1)$ time. Do this, then find $\sum S_i \cdot i$ and finally divide it by $N \cdot N!$ to obtain the answer.

Note that divisions need be performed under modulo, i.e, finding the multiplicative inverse and multiplying by it.

## TIME COMPLEXITY

$\mathcal{O}(N)$ per test case.

```
mod = 10**9 + 7
fac = [1]
ifac = [1]
for i in range(1, 200005):
        fac.append(i * fac[i-1])
        fac[i] %= mod
        ifac.append(pow(fac[i], mod-2, mod))
def C(n, r):
        if r < 0 or n < r: return 0
        return (fac[n] * ifac[r] * ifac[n-r])%mod

for _ in range(int(input())):
        n = int(input())
        ans = 0
        for i in range(1, n+1):
                ans += i * fac[n-i] * (C(n, i) + i*C(n, i+1))
                ans %= mod
        ans *= pow(fac[n] * n, mod-2, mod)
        print(ans % mod)
```