Codeforces 57 C Array

Combinatorics

Descirption

Calculate numbers of non-increasing and non-decreasing sequences with n elements ranged in $[1, n]$.

Solution

Reversing a non-decreasing array lead to a non-increasing one, thus we only need to calculate the first type.

Regarding that we choose i numbers, and increasingly use them in the array, that means split the whole array into i parts, the same as put $i-1$ boards in $n-1$ blanks between the elements. So the number is C_{n-1}^{i-1}. And we can choose these i numbers in C_n^i ways, total numbers of non-decreasing arrays is $\sum\limits_{i=1}^{n}\binom{n}{i}\binom{n-1}{i-1}$.

Multiply the ans by 2, but there're some re-counted situations. When all the elements are equal, it's both non-inc and non-dec, so subtract the ans by n.

Code

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>

using namespace std;

typedef long long ll;

const int maxn = 1e+5 + 5;
const ll p = 1e+9 + 7;

int n;
ll fac[maxn], inv[maxn], ans;

ll quick_power(ll base, ll index) {
  ll ret = 1;
  while (index) {
    if (index & 1) ret = ret * base % p;
    base = base * base % p;
    index >>= 1;
  }
  return ret;
}
```

```
inline ll C(int nn, int mm) {
  return fac[nn] * inv[mm] % p * inv[nn - mm] % p;
}

int main() {
  scanf("%d", &n);
  fac[0] = inv[0] = fac[1] = inv[1] = 1ll;
  for (int i = 2; i <= n; ++i) {
    fac[i] = fac[i - 1] * (ll)i % p;
    inv[i] = quick_power(fac[i], p - 2);
  }
  for (int i = 1; i <= n; ++i){
    ll tmp = C(n, i) * C(n - 1, i - 1) % p;
    ans = (ans + tmp) % p;
  }
  ans = (ans * 2ll % p - n + p) % p;
  cout << ans << endl;
  return 0;
}
```

Summary

Begin solving combinatorics problems today. Need to improve the ability of recognize the model behind the problem. At first, I didn't realize the way of choosing and using numbers, which leads to the failure of solving problem.