Number Theory Lecture 8

Sunday, 14 July 2024 5:01 PM

H.W:- Find $^nC_r \% p$ given n and r, $p = 10^9+7$

limits:- T testcases

$$\boxed{\begin{array}{c} 1 \le T \le 10^5 \\ 1 \le r \le n \le 10^5 \end{array}}$$

$$\boxed{^nC_r} = \frac{n!}{(n-r)!\, r!} \% p$$

$$= \Big(\underbrace{(n!) \% p}_{n \times (n-1) \times (n-2) \cdots 1,\ O(n)} * \underbrace{[(n-r)! \% p]}_{O(n-r)}^{-1} \% p * \underbrace{[r! \% p]^{-1} \% p}_{O(n+\log p)} \Big) \% p$$

$O(\log p)$

$a^{p-2} \% p$

$O(n + \log p)$

$O(n + \log p)$

$$^nC_r = \frac{n!}{(n-r)!\, r!}$$

$$^5C_2 = \frac{5!}{2!\, 3!} = \frac{120}{2 \times 6} = 10$$

$$\underbrace{^{5000}C_{2000} \% p}_{\boxed{< p}} = \frac{5000!}{2000!\, 3000!} \% p$$

$\frac{a}{b} \% p = (a \times b^{-1}) \% p = a \% p * b^{-1} \% p$

$5000! \% p$ ✓

$p = 7$

$5^{-1} = 3$

$\frac{4}{5} \bmod 7$

$2000! \% p * 3000! \% p$ ✓

✓

$\frac{a}{b} \% p$

$= (4 * 5^{-1}) \bmod 7$

$= (4 * 3) \bmod 7$

$= 12 \bmod 7$

$= \boxed{5}$

$a^{p-2} \% p$

$= a^{-1} \% p$

$N = 60$ $\quad p = 7$

$D = 12$

$\boxed{\frac{60}{12} = 5}$

$N = \frac{4}{5} \% 7$

$D =$

$(a * b * c) \% p$

$\equiv (((a * b) \% p) * c) \% p$

```
ll pow(int a, int b) {
    if(b==0)    return 1;
    ll ans = pow(a, b/2);
    ans = (ans*ans)%MOD;
    if(b%2 == 1)  ans = (ans*a)%MOD;
    return ans;
}
ll mod_inv(int a) {
    return pow(a, MOD-2);
}
ll fact(int n) {
    ll ans = 1;
    FOR(i, 2, n+1)  ans = (ans*i)%MOD;
    return ans;
}
ll ncr(int n, int r) {
    ll nf = fact(n);
    ll nrf = mod_inv(fact(n-r));
    ll rf = mod_inv(fact(r));
    return (rf*((nf*nrf)%MOD))%MOD;
}
void solve() {
    int n,r;
    cin >> n >> r;
    cout << ncr(n, r) << endl;
}
```

=x=

$T = O(N + \log P)$

$= O(N)$

$P = 10^9 + 7$

$N, T \le 10^5$

T test cases → $T = O(T \cdot N)$

$\simeq \underline{10^{10}}$ won't work.

Precomputation ?? $\hookrightarrow$ Factorial. ✓

factorial $\to O(N)$

$1 \to n \quad O(N^2)$

$O(N)$

$f(i) = (i * f(i-1)) \% p$

$T \to {}^nC_r = \dfrac{n!}{(n-r)!\, r!}$

$n! \to O(1)$, $(n-r)! \to O(1)$, $r! \to O(1)$

$\log p$

$O(N + T \log p)$

$N, T \leq 10^5$

$fi = 0!^{-1}, 1!^{-1}, 2!^{-1} \ldots \quad N!^{-1}$

$\hookrightarrow$ Precompute factorial inverses $\quad O(N \log p)$

$O(N \log p + T)$

$f[i] = (i * f[i-1]) \% p$

$fi[i] = ((i+1) * fi[i+1]) \% p$

$a!^{-1}$

$\frac{1}{a!} \bmod p$

$(a+1)!^{-1} \bmod p$

$\frac{1}{(a+1)!} \bmod p$

$a!^{-1} \bmod p = (a+1) \cdot (a+1)!^{-1} \bmod p$

$\frac{a+1}{(a+1)!} \bmod p$

$= \frac{1}{a!} \bmod p$

$= a!^{-1} \bmod p$

$T = O(N + \log p + T)$

$\approx \boxed{O(N+T)}$

```cpp
const int MOD = 1e9+7;
const int N = 1e5;
ll f[N+1];
ll fi[N+1];
ll pow(int a, int b) {
    if(b==0)    return 1;
    ll ans = pow(a, b/2);
    ans = (ans*ans)%MOD;
    if(b%2 == 1)  ans = (ans*a)%MOD;
    return ans;
}
ll mod_inv(int a) {
    return pow(a, MOD-2);
}
ll fact(int n) {
    ll ans = 1;
    FOR(i, 2, n+1)  ans = (ans*i)%MOD;
    return ans;
}
void precompute() {
    f[0] = 1;
    FOR(i,1,N+1) f[i] = (i*f[i-1])%MOD;
    fi[N] = mod_inv(f[N]);
    RFOR(i,N-1,0) fi[i] = ((i+1)*fi[i+1])%MOD;
}
ll ncr(int n, int r) {
    ll nf = f[n];
    ll nrf = fi[n-r];
    ll rf = fi[r];
    return (rf*((nf*nrf)%MOD))%MOD;
}
void solve() {
    int n,r;
    cin >> n >> r;
    cout << ncr(n, r) << endl;
}
```

O(n) precomputation, able to calculate ${}^nC_r$ in O(1) time

what if you want to reuse all binomial coefficients repeatedly?

${}^nC_r$ for all $0 \le r \le n \le N$

(n) → ${}^nC_0, {}^nC_1, {}^nC_2 \ldots {}^nC_n$ → $O(n)$ $O(n^2)$ ✓

$1 \le n, r \le 10^3$

${}^nC_r = {}^nC_{n-r}$

✓ ${}^nC_r = {}^{n-1}C_r + {}^{n-1}C_{r-1}$

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

```
N = 10
MOD = 1000000007
ncr = [[0 for _ in range(N+1)] for _ in range(N+1)]
ncr[0][0] = 1
for i in range(1, N+1):
    ncr[i][0] = 1
    ncr[i][i] = 1
    for j in range(1, i):
        ncr[i][j] = (ncr[i-1][j] + ncr[i-1][j-1])%MOD
print(*ncr)
```

https://leetcode.com/problems/powx-n/

```python
def pow(a, b):
    if b==0:
        return 1.0
    ans = pow(a, b//2)
    ans = ans*ans
    if b%2 != 0:
        ans = ans*a
    return ans
class Solution:
    def myPow(self, x: float, n: int) -> float:
        if n>=0:
            return pow(x, n)
        else:
            return 1.0/pow(x, -n)
```

```cpp
double pow(double a, long b) {
    if(b==0)
        return 1.0;
    double ans = pow(a, b/2);
    ans = ans*ans;
    if(b%2==1)  ans = ans*a;
    return ans;
}
class Solution {
public:
    double myPow(double x, int n) {
        long nn = n;
        if(n>=0) return pow(x, nn);
        else return 1.0/pow(x,-nn);
    }
};
```

https://leetcode.com/problems/count-anagrams/

too $\qquad \dfrac{3!}{2!\,1!} = 3$

hot $\qquad \dfrac{3!}{1!\,1!\,1!} = 6$

aa $\qquad \dfrac{2!}{2!} = \boxed{1}$

Hello World

$\dfrac{5!}{2!} \times 5!$ ✓

too hot

$3 \times 6 = \boxed{18}$

cat dog aab bab

$3! \times 3! \times \dfrac{3!}{2!} \times \dfrac{3!}{2!}$

aabbc abcbc

$\dfrac{5!}{2!\,2!} * \dfrac{5!}{1!\,2!\,2!}$

% P

```python
N = 100000
MOD = 1000000007
f = [1 for _ in range(N+1)]
fi = [1 for _ in range(N+1)]
def pow(a, b):
    if b==0:
        return 1
    ans = pow(a, b//2)
    ans = (ans*ans)%MOD
    if b%2!=0:
        ans = (ans*a)%MOD
    return ans
for i in range(1, N+1):
    f[i] = (i*f[i-1])%MOD
fi[N] = pow(f[N], MOD-2)
for i in range(N-1, -1, -1):
    fi[i] = ((i+1)*fi[i+1])%MOD
def ana(w):
    c = [0 for _ in range(26)]
    for ch in w:
        c[ord(ch)-ord('a')] += 1
    ans = f[len(w)]
    for n in c:
        ans = (ans*fi[n])%MOD
    return ans
class Solution:
    def countAnagrams(self, s: str) -> int:
        a = list(s.split())
        ans = 1
        for w in a:
            ans = (ans*ana(w))%MOD
        return ans
```