THINK FAST

Competitive Programming

From Problem 2 Solution in O(1)

Number Theory

Modular Arithmetic Apps

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

Recall: Prime power divides N

```
// largest x such that p^x divides n
int maxpowInFact(ll n, ll p) {  //O(logn)
    int power = 0;
    for(ll i = p; i <= n && i > 0; i = i * p)
        power += n/i;

    return power;
}
```

Wilson Theorem

- (p-1)! %p = p-1 = -1 where p is prime number
- Here is 1 trick to use it
- Compute n! % p such that n < p but **very close** to it: E.g. 25! % 29
- Remember: 25! = 28! / (28 * 27 * 26)
- From Wilson: 28! = -1
- Then just compute this expression using mod inverse. Just 3 steps!
- Let's see a direct usage for it

Factorial % P (excluded)

- Given n, compute n! % p, after removing every p from the n!
- E.g. F(n = 10, p = 5)
- = 1 * 2 * 3 * 4 * **1** * 6 * 7 * 8 * 9 * **2**
- Notice, 5 -> 1 and 10 -> 2 after removing 5's
- Why do so? In some factorial computations, you need answer %p and you know numerator and denominator cancelled the all ps

Find 38! % 5

1	2	3	4
6	7	8	9
11	12	13	14
16	17	18	19
21	22	23	24
26	27	28	29
31	32	33	34
36	37	38	

5 = 5 * 1
10 = 5 * 2
15 = 5 * 3
20 = 5 * 4
25 = **5 * 5**
30 = 5 * 6
35 = 5 * 7

1) %5 every row is: 1 2 3 4, repeated [38 / 5] = 7 times and reminder 38%5 = 3
2) For 2nd table, remove one of the 5's, then remaining is new sub-problem

Find 38! % 5

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	

1
2
3
4
5
6
7

1) Then $(4!\% 5)^7$ and also 3! % 7, 2nd table is subproblem 7! % 5
2) Answer $4!^7 * 3! * F(7, 5)$
3) From Wilson's : $(4!\% 5)^7 = -1$ and $(4!\% 5)^8 = 1$
4) In other words, n/p let just determine the sign

Factorial % P (excluded)

```
//if p > n, then just lp to n and calc f(n)
// O(p*logn)
// (1 * 2 * 3.....n) = (1 2 3 .... p-1 p p+1... 2p 2p+1....p*p...n)
// Get Ps out: p, 2p, 3p....p*p...p*p*p.....n/p
//      Remove 1 p: 1 2 3 ..... n/p [Same subproblem]
// Mod on others: 1 2 3 .... p-1 1 2 3 .... p-1 ..... 1 2 ... n%p
//
// = ((p-1)!)^(n/p) * (n%p)! * f(n/p, p)
ll factModP(ll n, ll p) {   // after excluding all p's
    ll res = 1;
    while (n > 0) {
        for (ll i = 1; i <= n % p; i++)
            res = (res * i) % p;
        n /= p;
        // we should evaluate A=(1*2...p-1)%p =(p-1)! %p, and then find A^(n/p)
        // From Wilson's Theorem, (p-1)! %p = -1,
        // then if n^p is odd just flip answer and add mod
        if (n % 2 != 0)
            res = p - res;
    }
    return res;
}
```

Combinations

```
ll nCkModP_general(ll n, ll k, ll p) {
    ll anyPow = maxpowInFact(n, p) - maxpowInFact(k, p) - maxpowInFact(n-k, p);
    if(anyPow)
        return 0;   // then nCk divisible by p

    ll up = factModP(n, p);
    ll down = ( factModP(k, p) * factModP(n-k, p) )%p;
    ll downInv = pow(down, p - 2, p);

    return (up*downInv)%p;
}
```

Combinations: Lucas Theorem

$$\binom{m}{n} \equiv \prod_{i=0}^{k} \binom{m_i}{n_i} \pmod{p},$$

where

$$m = m_k p^k + m_{k-1} p^{k-1} + \cdots + m_1 p + m_0,$$

and

$$n = n_k p^k + n_{k-1} p^{k-1} + \cdots + n_1 p + n_0$$

- Lucus is O(plogn), great for small p (precompute nCk e.g. using dynamic programming)
- Its generalization handles: p^x

Combinations: Lucas Theorem

```
// A generalized version can handle p^x
// E.g. http://www.dms.umontreal.ca/~andrew/PDF/BinCoeff.pdf
ll nCkModP_lucas(ll n, ll k, ll p) {
    ll res = 1, np, kp;
    while (n > 0 || k > 0) {
        np = n % p, kp = k % p, n /= p, k /= p; // find numbers n, k in base p
        ll up = 1, down = 1;
        for (ll i = np - kp + 1; i <= np; ++i)
            up = (up * i) % p;
        for (ll i = 1; i <= kp; ++i)
            down = (down * i) % p;
        res = (res * ((up * pow(down, p - 2, p)) % p)) % p; // a^(p-2)%p = modInv(a, p)
    }
    return res;
}
```

Combinations nCk % m

- What about nCk % m where m is not prime number?
- Assume can’t do it just with DP
- Factorize m to prime powers:
  - e.g.: 12 = {4, 3}
- Compute nCk with p^x (e.g. using generalized Lucas theorem)
- Using Chinese remainder theorem to combine the overall results

Catalan number

- Many problems turns out to be Catalan number: 1, 1, 2, 5, 14, 42, 132, 429, 1430

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\,n!} = \binom{2n}{n} - \binom{2n}{n+1}$$

- It has a recurrence too: $C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k}$
- Compute C % P?
- Same logic as nCk or even use 2 nCk
- There are many apps...must read
- E.g. # of n pairs of balanced parentheses (())()

Catalan number % P (excluded)

```
ll catlan(ll N, ll K, ll M) {
    ll a = nCkModK(2 * N, N, M);
    ll b = nCkModK(2 * N, N - 1, M);
    return (a - b + M) % M;
}
```

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً

Problems

- 10007, 10303