ICPC Howtos

Yuchen Lei

April 28, 2019

Abstract

Short snippets to solve some specific problems.

Contents

ICPC Howtos

Math

How to calculate $a^b \, mod \, m$ in $O(\log b)$ time:

```
int64_t fast_pow(int64_t a, int64_t b, int64_t m) {

    int64_t r = 1;
    for (; b; b >>= 1, a = a * a % m)
        if (b & 1)
            r = r * a % m;
    return t;
}
```

How to modulate the number of large combinations

Use Lucas's theorem.

$$Lucas \binom{n}{m} \mod p = \binom{n \mod p}{m \mod p} * Lucas \binom{n/p}{m/p} \mod p \, Lucas \binom{n}{0} = 1$$

```
int64_t Lucas(int64_t n, int64_t m) {
    return m == 0 ? 1 : C(n % p, m % p) * Lucas(n / p, m / p) % p;
}
```

How to calculate multiplicative inverse (mod n):

If n is a prime

$$a^{-1} \equiv a^{n-2} \mod n$$

```
int64_t inv(int64_t x, int64_t p) { return fast_pow(x, p, p - 2); }
```

If n is not a prime

$$a * a^{-1} \equiv 1 \mod n \Rightarrow a * a^{-1} - n * t = 1$$

So we can use extended gcd to solve it.

```
void exgcd(int64_t a, int64_t b, int64_t &x, int64_t &y)

{ b == 0 ? (x = 1, y = 0) : (exgcd(b, a % b, y, x), y -= x * (a / b)); }
int64_t inv(int64_t a, int64_t n) {
    int64_t x, y;
    exgcd(a, n, x, y);
    return x - ((x - n + 1) / n) * n;
}
```