Basic Math

Topics :

1. Number System (Base Conversion)
2. Logarithm
3. Ceil & Floor
4. Non-Rounding Fractional Number
5. Number Series / Sequence
6. Big Integer
7. GCD & LCM
8. Bit manipulation
9. Misc

Number System (Base Conversion)

1. Base to Decimal :

Suppose there is number x in base B, then $x = d_n d_{n-1} .. d_3 d_2 d_1 d_0$, where $0 \le d_i < B$.

The number x in decimal will be $d_n \times B^n + d_{n-1} \times B^{n-1} + \ldots + d_1 \times B^1 + d_0 \times B^0$.

For example :

(AB12) in Base 16 = (??) in base 10

B

$A \times 16^3 + B \times 16^2 + 1 \times 16^1 + 2 \times 16^0$

$= (43794)_{10}$

2. Decimal to Base :

$x\%B$ will give us the last digit of x in base B. We can get the remaining digits by repeatedly dividing x by B and taking its last digit until x becomes 0.

$x = x : d_0 = x\%B$

$x = \lfloor \frac{x}{B} \rfloor : d_1 = x\%B$

$\ldots$

For example :

(14) in Base 10 = (??) in base 2

$14 \% 2 = 0$

$7 \% 2 = 1$

$3 \% 2 = 1$

$1 \% 2 = 1$

Code

1. Base to Decimal : $X = (AB12)_{16}$

```cpp
int baseTodecimal ( string x, int base )
{
    int res = 0;
    int len = x.length();

    int coef = 1;
    for ( int i = len - 1; i >= 0; i-- ) {
        int d;
        if(x[i] >= '0' && x[i] <= '9') d = x[i] - '0';
        else d = (x[i] - 'A') + 10;
        res += d * coef;
        coef *= base;
    }
    return res;
}
```

2. Decimal to Base :

```cpp
string symbol = "0123456789ABCDEF";

string decimalTobase(int x, int base)
{
    string res = "";
    while(x)
    {
        int r = x % base;
        res = res + symbol[r];
        x/=base;
    }
    if(res=="")
        res = symbol[0];
    reverse(res.begin(), res.end());
    return res;
}
```

Logarithm

If $b^y = x$, then $y = \log_b x$

Base Change Rule, $\log_b(a) = \dfrac{\log_x(a)}{\log_x(b)}$

$$\log_3(a) = \frac{\log_{10}(a)}{\log_{10}(3)}$$

log10(x) -> returns double/ long double value.

'x' can be int, float, double, long double, long long...

$$x = 10^{\log_{10}(y)}$$

Ceil & Floor

ceil(x) -> rounds up (returns double value)

floor(x) -> rounds down (returns double value)

if, $x = \dfrac{a}{b}$

$$\frac{a+(b-1)}{b}$$

$$\left\lceil \frac{5}{2} \right\rceil = 2$$

$5/3$

$$\lfloor 2.5 \rfloor = 2$$

$$\left(\frac{5+(3-1)}{3} \right) \Rightarrow \frac{5+2}{3}$$

Non-Rounding Fractional Number

```c
double num = 2020.99998888888;
char buffer[50];
int n;
sprintf(buffer, "%f", num); /// sprintf(char *str, format, variable)
printf("%s\n", buffer);

sscanf(buffer, "%d", &n); /// string to number
printf("%d\n", n);
```

$$\frac{19999}{1000}$$

19.999

Number Series & Sequence

visit site : https://oeis.org/ :p

1. Arithmetic progression
2. Geometric progression

Big Integer

1 8 9 3

7 2 1

1 6 1 4

c = 1

12

a + b

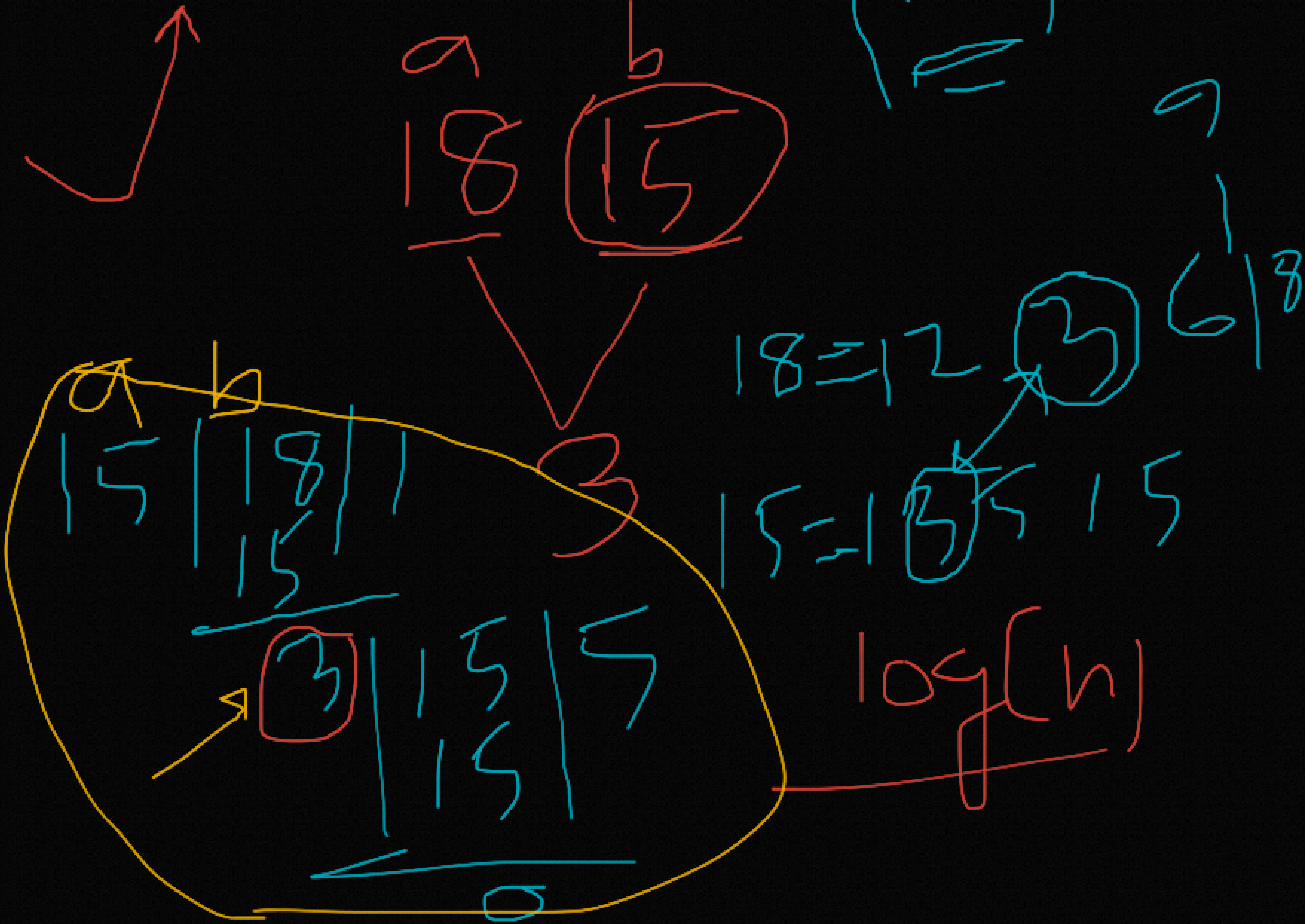
JAVA / PY

15 25

GCD & LCM

Basic Euclidean Algorithm for GCD

```
int gcd(int a, int b)
{
    if (a == 0)
        return b;
    return gcd(b % a, a);
}
```



a * b = gcd(a, b) * lcm(a, b)

Bit Manipulation

OR :

5 = 101
3 = 011 10
7 = 111

AND :

5 = 101
3 = 011
1 = 001

XOR :

5 = 101
3 = 011
6 = 110

5 = 101
5 = 101
00

(A + B)
(A & B)

1. a[l] ^ a[l+1] ^ a[l+1] ^ ... ^ a[r] = b[r] ^ b[l-1] (here b[i] is prefix xor till 'i')
2. (A + B) = (A ^ B) + 2 * (A & B)

Misc

1. (3 + 6 - 2 + 5) = ??
2. (2 + 6 / 2 + 4) = ??
3. double pow (double base, double exponent);

ceil(pow(5, 2))

$5^2 = 25.0000$

25.00001

24.9999

Modular Arithmetic

Topics :

1. Addition
2. Subtraction
3. Multiplication
4. Division (Inverse MOD)
5. Exponentiation (Big MOD)

Modular Addition

Number Line : -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
Number Line (mod 5) : 2 1 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 ...

If we use mod n then we will get number 0 to n-1.

5 (mod 3) ≡ 8 (mod 3)

$$\boxed{-2+5}$$

$= 3$

-5 (mod 2) ≡ 1 (mod 2)

not "-1"

(-5)%2 = -1 + 2 = 1

```cpp
inline void norm(ll &a) { a %= mod; (a < 0) && (a += mod); }
```

(a + b) % MOD = (a % MOD + b % MOD) % MO

(a + b + c + ...) % MOD

= (a % MOD + b % MOD + c % MOD + ...) % MO

```cpp
inline ll modAdd(ll a, ll b) { a %= mod, b %= mod; norm(a), norm(b); return (a + b) % mod; }
```

Modular Subtraction

(a - b) % MOD = (a % MOD - b % MOD) % MO

```
inline ll modSub(ll a, ll b) { a %= mod, b %= mod; norm(a), norm(b); a -= b; norm(a); return a; }
```

Modular Multiplication

(a * b) % MOD = (a % MOD * b % MOD) % MO

```
inline ll modMul(ll a, ll b) { a %= mod, b %= mod; norm(a), norm(b); return (a * b) % mod; }
```

3 x 4

(4 + 4 + 4)

Modular Division

$\frac{5}{2} = 2.5$

$\left(\frac{a}{b}\right) \% \text{ mod} \neq \left(\frac{a \% \text{ mod}}{b \% \text{ mod}}\right) \% \text{ mod}$

$\left(\frac{1}{b}\right) \% \text{ m}$

$\left(\frac{a}{b}\right) \% \text{ mod} = (a b^{-1}) \% \text{ mod}$

[From modular multiplication]

$= (a \% \text{ mod} * b^{-1} \% \text{ mod}) \% \text{ mod}$

Fermat's little theorem

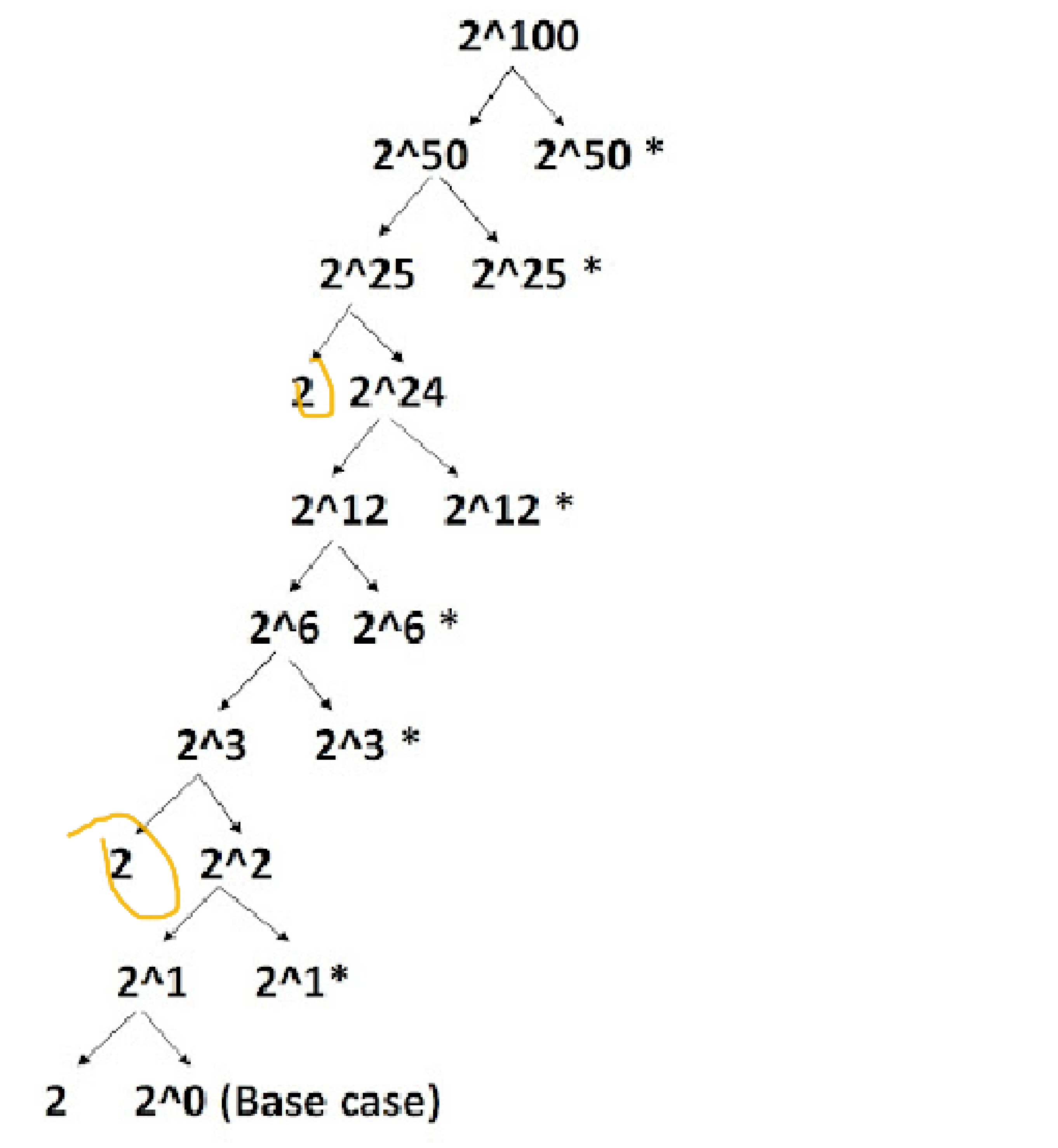
$a^{p-1} \equiv 1 \pmod{p} \rightarrow a^{p-2} \cdot a \equiv 1 \pmod{P} \rightarrow a^{P-2} \equiv a^{-1} \pmod{P}$

$b \rightarrow x \quad b \quad \% \text{ mod}$

here, a & p are co-primes.

Big Mod

Big Mod (Modular Exponentiation)




```
#define i64 long long
i64 M;
i64 F(i64 N,i64 P)
{
    if(P==0) return 1;
    if(P%2==0)
    {
        i64 ret=F(N,P/2);
        return ((ret%M)*(ret%M))%M;
    }
    else return ((N%M)*(F(N,P-1)%M))%M;

}
```

Code of Modular Inverse

```cpp
inline ll modInverse(ll a) { return modPow(a, mod - 2); }
inline ll modDiv(ll a, ll b) { return modMul(a, modInverse(b)); }
```

7 - 111

6 = 0110

Code of Big Mod

```cpp
inline ll modPow(ll b, ll p) { ll r = 1; while (p) { if (p & 1LL) r = modMul(r, b); b = modMul(b, b); p >>= 1LL; } return r; }
```

$$(6/3)\%0.5 = 2$$

$$\left(\frac{(6\%0.5)}{(3\%0.5)}\right)\%0.5$$

$$= \left(\frac{1}{3}\right)\%.5 = 0\%.5$$

$$= 0$$

10^5

$$\left(\frac{n!}{r!\,(n-r)!}\right)$$

% mod 10^9+7

0978