~~Advanced~~ Number Theory

Today's class is not that advanced

— Priyansh Agarwal

- ✔ Factorization
- ✔ Sieve of Eratosthenes
- ✔ Prime Factorization
  - Smallest Prime Factor
  - Number of Factors
  - Sum of Factors
- ✔ Revision
  - Modular Arithmetic
  - Exponentiation
  - Euclidean Algorithm

Factorization: Find all factors of a number N

- Naive way
  - Check every number for factor from 1 to N
  - O(N)
- Factors occur in pairs
  - N = P * Q
  - Without loss of generality if P ≤ Q => P <= sqrt(N)
- Efficient way
  - Check for every P from 1 to sqrt(N). Q = N / P
  - O(sqrtN)

$N = \underline{\underline{20}}$

1, 2, 4, 5, 10, 20

N

$\underline{\underline{O(N)}}$

```
for (int i=1 ; i<=N ; i++) {
        if (N % i == 0)
                cout << i << " ";
}
```

$N \longrightarrow$ has a factor $\underline{\underline{p}}$

$\underline{\boxed{N/p}}$ is an integer

q

$q \times p = N$

$q \times p = N$

$q = \boxed{\dfrac{N}{p}}$ $p = \boxed{\dfrac{N}{q}}$

integers

+ve

for every factor p of N

there exist another factor $\rightarrow N/p$

factors always occur in pairs

$$p \times q = N$$

$$\underline{\underline{\min(p, q)}} \rightarrow \text{maximum value of this}$$

$N = 20$

$p = 1, \quad q = 20$

$\boxed{p = 2, \quad q = 10}$

$p = 4, \quad q = 5$

$p = 5, \quad q = 4$

$\boxed{p = 10, \quad q = 2}$

$p = 20, \quad q = 1$

$p = 1$, $p = 2$, $p = 4$, $p = 5$

$\downarrow$

$q = 20$, $q = 10$, $q = 5$, $q = 4$

(p, q) $\qquad$ $p \times q = N$

$(p \leq q)$

Iterate over factor $\underline{\underline{p}}$

such that $\underset{q}{\underline{\underline{N/p}}} \geq p$

$p \longrightarrow N$

N/p is also a valid factor

$p \times q = N$

$\boxed{p} \leq q$

$N = 20 \qquad p = 10$

$$\boxed{p \times q \geq p^2}$$

$p, \quad q$

$p \leq q$

$p^2 \leq qp$

$\boxed{q \times p \leq N}$, $\boxed{q > p}$

$\boxed{p^2 \leq q \times p \leq N}$

$p^2 \leq N$

$\boxed{p \leq \mathrm{sqr}(N)}$

count = 0

```
for (int i = 1 ; i <= sqr(N); i++)
        int p = i
        if( N % p == 0 ){
            int q = N / p
            if(p == q)
                count ++
            else
                count += 2 }
}
```

$N = 16$

1, 2, 4, 8, 16

$p = 4$

$q = \frac{N}{p} = \frac{16}{4} = 4$

$\boxed{p \leq q} \longrightarrow p \leq sqrt(N)$

$\boxed{q \leq p} \longrightarrow q \leq sqrt(N)$

N tell me whether N has odd no. of factor or even no. of factors

without actually finding out all the factors

factors occur in pairs

$p \rightarrow \frac{N}{p}$ is also a factor

when $p = \frac{N}{p}$ or $p^2 = N$

or $\boxed{p == \text{sqrt}(N)}$

Factorization: Find all factors of a number N

Implementation (Efficient way)

```
vector<long long> findFactors(long long n) {
    vector<long long> factors;
    for (long long d = 1; d * d <= n; d++) {
        if (n % d == 0) {
            factors.push_back(d);
            if (n / d != d) // d should be different from n / d
                factors.push_back(n / d);
        }
    }
    return factors;
}
```

Prime Factorization: Represent N as $p_1^{k1} p_2^{k2} \ldots\ldots p_m^{km}$

- Naive way
  - Find all factors of N.
  - Check each number for prime
  - Find how many times each prime divides N
- Efficient way
  - Only one of the prime factors of N can be > sqrt(N)
  - Iterate from 2 to sqrt(N)
  - Check if current number divides N. If yes, keep dividing N by that number as many times as possible
  - Invariant: any new number that will divide N now will be a prime number
  - O(sqrtN)

$\boxed{N = 20}$

$\boxed{2^2 \cdot 5^1}$

$\boxed{N = 36}$

$\boxed{2^2 \cdot 3^2}$

$\boxed{N = 30}$

$\boxed{2^1 3^1 5^1}$

N

list down all factor of

N ~~is~~ in sqrt(N) time

sqrt(N)

iterate over every factor

sqrt(N)

check for prime

if yes

$O(\sqrt{N} \cdot \sqrt{N}) = O(N)$

time

The smallest factor of N which is not 1 is found to be prime

if x divides N

N
$=$

1 x

x

~~fact~~ x is smallest factor $\neq 1$ for N

$\hookrightarrow$ claim $\rightarrow$ x is a prime

claim

$\boxed{\text{assume } x \text{ is not a prime}}$ ✓

then there exists a y st.

$y \neq 1$ & $y \neq x$ and

$\boxed{y \text{ divides } x} \rightarrow y < x$

then y such that y divides

X

we know that X divides N

$\Rightarrow$ y also divide N

and $y < X$

Find out smallest factor of N which is not 1

x

Divide N by x as many times as possible

$\boxed{N = 36}$ $x = 2$

$36/2$ $\beta 1$

$\downarrow$

$18/2$ $\boxed{2^2}$

$\downarrow$

$\boxed{9}$ $\beta 2$

$N = \underline{\boxed{x^k}} \cdot z$

separately

x cannot occur in prime factoriz

of z

1. Smallest factor of N which is not equal to 1 is prime

2. assume x^k divide N and

x^{k+1} doesn't divide

$$\boxed{\frac{N}{x^k}}$$

① X is smallest prime factor of N

$$\boxed{\frac{N}{x^k}} \rightarrow y$$

$$\boxed{y > X}$$

Smallest factor of $\boxed{\frac{N}{x^k}}$ (y)

$y > x$

y is prime

$\downarrow$

M

$N = 36, \quad x = 2, \quad \frac{N}{x^k} \rightarrow \frac{36}{2^2} = \boxed{9}$

```
for ( int i = 2 ; i <= sqrt(N); i++)
{
    if ( N % i == 0 ) {  count = 0
        while ( N % i == 0 ) {
            count ++ ;
            N = N / i;
        }
    }
}
```

$N \rightarrow \frac{N}{x^k}$ if x is smallest prime factor of N

$$N = \boxed{p_1^{k_1}} \cdot p_2^{k_2} \cdots p_m^{k_m}$$

x

$$\frac{N}{p_1^{k}} = p_2^{k_2}\, p_3^{k_3} \cdots p_m^{k_m}$$

$$\boxed{N/x^k}$$

$$N = \boxed{p_1^{k_1}} \; p_2^{k_2} \cdots p_m^{k_m}$$

$$\frac{N}{p_1^{k_1}} = p_2^{k_2} \; p_3^{k_3} \cdots p_m^{k_m}$$

N = 100

i = 2

N = N/2 → 50

count = 1

N = N/2 → $\boxed{25}$

count = 2

i = 3 , i = 4 , i = 5

```
for ( int i = 2 ; i <= sqrt(N); i++) {
    if ( N % i == 0 ) { count = 0
        while ( N % i == 0 ) {
            count ++ ;
            N = N / i ;
        }
    }
}
```

2^4 divides N

if 2 divides N

$i = 2$

$i = 4$

$i = 8$

$i = 16$

$N \rightarrow 2^4$ divides it

$N/2$

$N/2$

$N/4$

$N/2$

$\rightarrow \boxed{N/2^4}$

$N = 29$

$=$

$N = 14$

$\boxed{3}$

$N = 28 \rightarrow \boxed{5}$

$2^1 \cdot 7^1$

$2^2 \cdot 7^1$

$p_1 \cdot p_2 > N \leftarrow p_1\ p_2$

$p_1 > \operatorname{sqrt}(N)$

$p_2\ \operatorname{sqrt}(N)$

Prime Factorization: Represent N as $p_1^{k1} p_2^{k2} \ldots\ldots p_m^{km}$

N = 32

2, 4, 8, 16, 32

Implementation (Efficient way)

```
vector<long long> primeFactorization(long long n) {
    vector<long long> factorization;
    for (long long d = 2; d * d <= n; d++) {
        while (n % d == 0) {
            factorization.push_back(d);
            n /= d;
        }
    }
    if (n > 1) // checking for the only prime factor that is > sqrt(N)
        factorization.push_back(n);
    return factorization;
}
```

sqrt(N)

Sieve of Eratosthenes: Find all prime numbers from 1 to N

- Naive way
  - Iterate from 1 to N
  - Check if current number is prime: O(sqrtN) or O(logN) using Miller Rabin
  - Time: O(NrootN), Space: O(1)

- Efficient way
  - Iterate from 2 to N
  - Keep marking numbers as non primes by iterating on multiples of primes
  - If current number (X) is unmarked -> X is Prime
  - Mark all multiples of X as non-prime
  - Time: O(Nlog(logN)), Space: O(N). Proof

Precomputation of Primes

Q queries → N ↘ prime
or not

$1 \le q \le 10^5$

$1 \le N \le 10^7$

Sieve of Eratosthenes

✗	✓	✓		✓		✓	
1	2	3	4	5	6	7	8

Prime number → divisible by 1 and itself

Prime number (x) is not divisible by any number > 1 and < x

~~1~~	2	3	4	5	6	7	8	9	10	11	12	13	14
	1	1	0	1	0	1	0	0	0	1	0	1	0

$i = 2 \longrightarrow \frac{N}{2}$

$i = 3 \longrightarrow \frac{N}{3}$

$i = 4 \longrightarrow \frac{N}{4}$

$\vdots$

$$\underbrace{\frac{N}{2} + \frac{N}{3} + \frac{N}{4} - - - - \frac{N}{N}}_{\leq n \log n}$$

Harmonic Series

$i = 2$ $\quad$ $N \mid 2$

$i = 3$ $\quad$ $N \mid 3$

$i = 4$ $\quad$ α

$i = 5$ $\quad$ $N \mid 5$

$$N + \frac{N}{2} + \frac{N}{3} + \frac{N}{4} \cdots \frac{N}{N} \le N \log N$$

avg case

$$\frac{N}{2} + \frac{N}{3} + \frac{N}{4} \cdots \frac{N}{N}$$ 4

only prime $\longrightarrow$ $\frac{N}{2} + \frac{N}{3} + \frac{N}{5} \rightarrow$ 2

$$2 < 4 < x$$

$i = 2$ 4, 6, 8, 10, 12, 14, 16

$i = 3$ $\boxed{6}$, 9, 12, 15, 18, 21, 24

$\boxed{i = 5}$ 10, 15, 20, 25

$\boxed{p}$

$p \cdot 2$ $p \cdot 3$ $p \cdot 4$

$p \cdot 5$ $p \cdot 6$ $p \cdot 7$ $p \cdot 8$

$p \cdot 9$ $p \cdot 10$

$\boxed{p = 11}$ $\boxed{p \cdot 11}$

~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~

11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19

~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ ~~25~~ ~~26~~ ~~27~~

~~28~~ 29 ~~30~~ 31 ~~32~~ ~~33~~ ~~34~~ 35

$$N + \frac{N}{2} + \frac{N}{3} + \frac{N}{4} - \cdots - \frac{N}{N} \le N\log N$$

$\times$

All no's

$$\frac{N}{2} + \frac{N}{3} + \frac{N}{4} - \cdots - \frac{N}{N}$$

Primes

$$\boxed{N\log(\log N)}$$

Square of primes

$N = 10^7 \longrightarrow N \log N \rightarrow 10^7 \cdot 24$

$N \log(\log N) \longrightarrow 10^7 \cdot \log(24)$

$$\boxed{10^7 \cdot 5}$$

$N/\log N$

Sieve of Eratosthenes: Find all prime numbers from 1 to N

Implementation (Efficient way)

```cpp
vector<bool> isPrime(n + 1, true);
isPrime[0] = isPrime[1] = false;

for (long long i = 2; i <= n; i++) {
    if (isPrime[i]) {
        for (long long j = i * i; j <= n; j += i) {
            // why iterate from i * i and not 2 * i
            isPrime[j] = false;
        }
    }
}
```

Smallest Prime Factor: Precomputing SPF for every N using Sieve

- Idea
  - Iterate through all the primes and for every multiple of that prime P, see if P is its smallest prime factor or not.
  - We can use the same idea as that in sieve to find all primes and iterate over only relevant multiples of P. → $p \times p$
  - Time for precomputation: O(Nlog(logN))

- Use case
  - Finding the prime factorization of a number in O(logN) time

N find out the smallest prime factor of N

```
for( int i=2 ; i*i<=N ; i++){
        if (N% i==0)
            ans = i
}
    ans = N
```

2 3 4 5 6 7 8 9

2 3 2 5 2 7 2 3

10 11 12 13 14 15 16

2 11 2 13 2 3 2

17 18 19 20 21 22 23

17 2 19 2 3 2 23

24 25 26 27 28 29 30

2 5 2 3 2 29 2

$$N \longrightarrow p_1^{k_1}\, p_2^{k_2}\, p_3^{k_3} \cdots\cdots p_m^{k_m}$$

$$\boxed{p_1} < p_2 < p_3 < p_4 \cdots < p_m$$

$O(1)$

$$N \longrightarrow \boxed{\frac{N}{p_1^{k_1}}} \longrightarrow \boxed{\frac{N}{p_1^{k_1} \cdot p_2^{k_2}}}$$

```
while (N != 1){
    int prime = spf[N], count = 0
    while( N % prime == 0){
        N = N/prime
        count++
    }
}
```

$N \rightarrow$ dividing it by a number

$$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{4} \rightarrow \frac{N}{8} \text{ --- - -}$$

$$\frac{N}{3} \rightarrow \frac{N}{9} \rightarrow \frac{N}{27} \rightarrow \frac{N}{27 \cdot 3}$$

Efficient Prime Factorization using SPF

```
vector<pair<int, int>> primeFactorization(int x, vector<int>& spf){
    vector<pair<int, int>> ans;
    while(x != 1){
        int prime = spf[x];
        int cnt = 0;
        while(x % prime == 0){
            cnt++;
            x = x / prime;
        }
        ans.push_back({prime, cnt});
    }
    return ans;
}
void solve(){
    int maxN = 1e6;
    vector<bool> isPrime(maxN, true);
    vector<int> spf(1e6, 1e9);
    for(long long i = 2; i < maxN; i++){
        if(isPrime[i]){
            spf[i] = i;
            for(long long j = i * i; j < maxN; j += i){
                isPrime[j] = false;
                spf[j] = min(spf[j], (int)i);
            }
        }
    }
    vector<pair<int, int>> primeF = primeFactorization(36, spf);
}
```

prime factorization

almost linear

spf

$N \rightarrow$ find out how many factors

N has

$$N = p_1^{k_1} \, p_2^{k_2} \cdots\cdots p_m^{k_m}$$

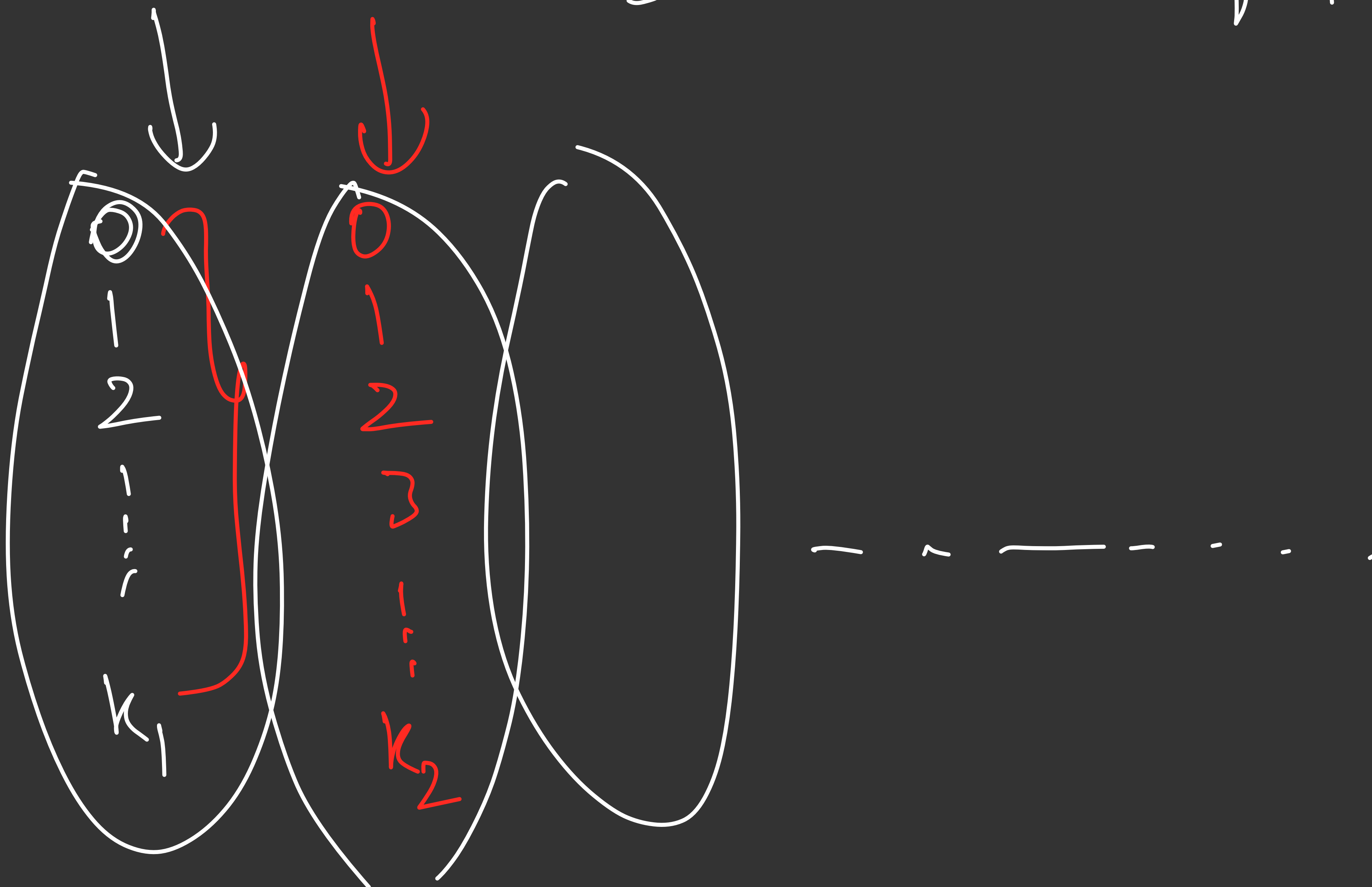
$$X \rightarrow p_1^{a_1} \, p_2^{a_2} \cdots\cdots p_m^{a_m}$$

$$\boxed{a_i \le k_i}$$

$$N = 2^2 \cdot 5^3 \cdot 7^2$$

$$X = 2^1 \cdot 5^3 \cdot 7^0$$

$p_1^{k_1} \; p_2^{k_2} \; p_3^{k_3} \; \cdots\cdots \; p_m^{k_m}$



$$(1+k_1)\cdot(1+k_2)(1+k_3)\ \ldots\ldots\ (1+k_m)$$

$$N = 10 \longrightarrow 2^1 \cdot 5^1$$

$$1 \longrightarrow 2^0 \cdot 5^0$$

$$2 \longrightarrow 2^1 \cdot 5^0$$

$$5 \longrightarrow 2^0 \cdot 5^1$$

$$10 \longrightarrow 2^1 \cdot 5^1$$

find out the sum of factors of a number N

$N = 10$ $\underbrace{1, 2, 5, 10}$ $= \underline{18}$

$N = 2^5$

$2^0, 2^1, 2^2, 2^3, 2^4, 2^5$

$N = 2^k$

$1, 2, 4, 8, 16 \text{ ----}$

$2^0 \quad 2^1 \quad 2^2 \quad 2^3 \quad 2^4 \text{ ---- } 2^k$

$$G.P \to a_0 \frac{\left(r^{n} - 1\right)}{r - 1}$$

$$\downarrow$$

$$1 \, \frac{\left(2^{k+1} - 1\right)}{2 - 1}$$

$$N \longrightarrow 2^{k}$$

$$N = 2^3 \cdot 5^1$$

$$\begin{pmatrix} 2^0 \cdot 5^0 & 2^1 \cdot 5^0 & 2^2 \cdot 5^0 & 2^3 \cdot 5^0 \\ 2^0 \cdot 5^1 & 2^1 \cdot 5^1 & 2^2 \cdot 5^1 & 2^3 \cdot 5^1 \end{pmatrix}$$

$$\left(2^0 + 2^1 + 2^2 + 2^3\right) \cdot \left(5^0 + 5^1\right)$$

$$N = 2^k \cdot 5^p$$

$$\underbrace{(2^0 + 2^1 \ldots 2^k)}_{\frac{1(2^{k+1} - 1)}{2-1}} \cdot \underbrace{(5^0 + 5^1 \ldots 5^p)}_{\frac{1(5^{p+1} - 1))}{5-1}}$$

Number and Sum of Divisors from Prime Factorization: Problem

- $\boxed{N = p_1^{e1} p_2^{e2} \ldots\ldots p_k^{ek}}$
  - Number of Divisors:

$$d(n) = (e_1 + 1) \cdot (e_2 + 1) \cdots (e_k + 1)$$

  - Sum of Divisors:

$$\sigma(n) = \frac{p_1^{e_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{e_2+1} - 1}{p_2 - 1} \cdots \frac{p_k^{e_k+1} - 1}{p_k - 1}$$

Modular Arithmetic: Link

$$(A + B) \% M = [(A \% M) + (B \% M)] \% M$$

$$(A - B) \% M = [(A \% M) - (B \% M) + M] \% M$$

$$(A * B) \% M = [(A \% M) * (B \% M)] \% M$$

$$\cancel{(A / B) \% M = [(A \% M) * (B^{-1} \% M)] \% M}$$

What is $B^{-1} \% M$ = Modular Inverse (Will be covered later)

Print the answer modulo 10^9+7

$$(5+6)\%2$$

$$((5\%2)+(6\%2))\%2$$

$$(1+0)\%2 = 1$$

$$(4-\underline{2})\%3$$

$$((\underline{4}\%3) - (2\%3))\%3$$

$$(1-2)\%3$$

$$(-1)\%3$$

$$(-1+3)\%3 = 2$$

$$A \% M = (A + M) \% M$$

$$\downarrow$$

$$((A \% M) + (M \% M)) \% M$$

$$(A \% M + 0) \% M$$

$$(A \% M)$$

$$(A - B)\%M$$

$$\downarrow$$

$$\big(\underbrace{(A\%M) - (B\%M)}\ + M\big)\%M$$

$$(\underline{f(a)} + \underline{f(b)}) \% M$$

$$f(a) > 10^{12}$$

$$(2^{100}) \% M$$

$$((2^{50})\% M \cdot (2^{50})\% M)\% M$$

$(5+8)\%3$

$13\%3$

$5\%3 + 8\%3$

$2+2$

4

$$(3 - 11) \% 5$$

$$((3 \% 5) - (11 \% 5) + 5) \% 5$$

$$(3 - 1 + 5) \% 5$$

$$2$$

$$(12 - 8)\%5$$

$$\downarrow$$

$$(12\%5) - (8\%5)$$

$$(2 - 3) \Rightarrow -1$$

$$(A \% M) \longrightarrow 0 \text{ to } M-1$$

$$A \% M == 0$$

M is a factor of A

$$A \% M = B \% M$$

$$\downarrow$$

$$(A - B) \% M = 0$$

modulo $\quad 10^9 + 7 \rightarrow 0 \rightarrow 10^9 + 6$

$\downarrow$

10^{12}

10^{18}

$$(10^9+7)$$

$$(a \cdot b) \% M$$

$$(10^9+6) \cdot (10^9+6)$$

$$(10^{18}) \% M \longrightarrow$$

$M \rightarrow \underline{10^{18}}$

$(a, b) \% 10^{18}$

$10^{18} \quad 10^{18}$

$(2^{100}) \% M \rightarrow$

✓ $A \% M = 0$

M is a factor of A

✓ $A \% M = B \% M$

$$\left.\begin{aligned} (A-B) \% M &= 0 \\ (B-A) \% M &= 0 \end{aligned}\right\}$$

$$A \,\%\, M = (A + k \cdot M) \,\%\, M$$

$$(n^k) \% M$$

$$\left(k \to 10^6\right)$$

$n \to 10^9$, $k \to 10^6$

$M = 10^9 + 7$

ans = 1

```
for ( int i = 1 ; i <= k ; i++ ) {
    ans = (ans * n) % m
}
```

n^k

$n \rightarrow 10^9$

$k \rightarrow 10^{18}$

$m \rightarrow 10^9 + 7$

$$(n^{17}) \rightarrow (n^8 \cdot n^8) \cdot n$$

$$2 \qquad\qquad 2 \cdot n$$

$y \rightarrow y^2$

$n^4 \cdot n^4$

$x \rightarrow x^2$

$n^2 \cdot n^2$

$n \; n$

a^n if $n \% 2 == 0$

$a^{n/2} \cdot a^{n/2}$

if $n \% 2 != 0$

$a^{n/2} \cdot a^{n/2} \cdot a^1$

$$(a^n)\%m$$

$$\left(a^{n/2} \cdot a^{n/2}\right)$$

$$\left\{\begin{array}{l} n \rightarrow 10^{18} \\ a \rightarrow 10^{9} \\ m = 10^{9}+7 \end{array}\right\}$$

$$a^n \rightarrow O(n)$$

$$a^n \rightarrow O(\log n)$$

Binary Modular Exponentiation: Link

Idea:

$$a^n = \begin{cases} 1 & \text{if } n == 0 \\ \left(a^{\frac{n}{2}}\right)^2 & \text{if } n > 0 \text{ and } n \text{ even} \\ \left(a^{\frac{n-1}{2}}\right)^2 \cdot a & \text{if } n > 0 \text{ and } n \text{ odd} \end{cases}$$

Time Complexity: O(log(n))

Binary Modular Exponentiation: Link

Implementation: (Don’t forget to take MOD when mentioned in problem)

Recursive

a^8

```
long long binpow(long long a, long long b) {
    if (b == 0)
        return 1;
    long long res = binpow(a, b / 2);
    if (b % 2)
        return res * res * a;
    else
        return res * res;
}
```

Iterative

a^8

```
long long binpow(long long a, long long b) {
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a;
        a = a * a;
        b >>= 1;
    }
    return res;
}
```

Binary Modular Exponentiation: Link

$b = 11$ $a = 2$

$res = 2^3$

$b = 5$ $a = 2^2$

Implementation: (Don’t forget to take MOD when mentioned in problem)

$res = 2^{11}$

$b = 2$ $a = 2^4$

$b = 1$ $a = 2^8$

Recursive

```
long long binpow(long long a, long long b) {
    if (b == 0)
        return 1;
    long long res = binpow(a, b / 2);
    if (b % 2)
        return res * res * a;
    else
        return res * res;
}
```

Iterative

```
long long binpow(long long a, long long b) {
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a;
        a = a * a;
        b >>= 1;
    }
    return res;
}
```

a^{17}

$$a^{17}$$

$$\downarrow$$

$$a^8 \cdot a^8 \cdot a$$

$$\downarrow$$

$$a^4 \cdot a^4$$

$$\downarrow$$

$$a^2 \cdot a^2$$

$$\downarrow$$

$$a \cdot a$$

$$a^{\boxed{17}} = \underbrace{10001}_{a^{16} * a^{1}}$$

$$a^{10} \longrightarrow 1010$$

$$a^{8} \cdot a^{2}$$

a^{11}

1011

a^4 a^3 a^2 a^1

✓ ✗ ✓ ✓