NUMBER THEORY -3

Property of Modulo:

gcd(a,b) = gcd(a,b-a) if b>a

proof:

gcd(a,b) = g

a/g = a1, b/g = a2, where a1,a2 are coprime.

g|a, g|b → g divies a and g divides b

g|a,

g|b-a (a2-a1 is an integer, so (b-a)/g is also integer)

so, g is a factor of gcd(a,b-a).

a/g = a1,

(b-a)/g = b1-a1.

are a1 and b1-a1 coprime?

let they have a divisor d, d is not 1.

d|a1, d|b1-a1→ d|b1-a1+a1→ d|b1

d|a1, d|b1

so d has to be 1, as a1 and a2 are coprime. but we assumed d is not 1. So contradiction.

gcd(a,b) = gcd(a,|b-a|) = gcd(a,a+b)***

Segmented Sieve

u are given range from [l,r]
u have to find prime numbers in this range
but   r<=1e12 and r-l+1<=1e5

any composite number n can be marked not prime by any prime number less than sqrt(n)

find all prime numbers till sqrt(r)

 ((l+i-1)/i)*i   number just greater than l and divisible by i

```
vector<int>  segment(int l,int r){

    // sqrt(r)<=1e6
    // we can use sieve to store all prime numbers till
here
    int lim=sqrt(r)+1;
    vector<bool> isPrime(lim+1,true);
    isPrime[1]=false;
    for(int i=2;i*i<=lim;i++){
        for(int j=i*i;j<=lim;j+=i){
            isPrime[j]=false;
        }
    }

    vector<int> primes; // all prime numbers til sqrt(r)
    for(int i=1;i<=lim;i++) if(isPrime[i])
primes.push_back(i);

    // [l,r]
    // [1e11-1e11+1e5]   r-l+1<=1e5

    // l---> 1
```

```
    // l+1 ---> 2                    l+i--> i+1
    // l+2 ---> 3
    // l+3---> 4
    // ...


    vector<bool> nPrime(r-l+2,true);

    for(int i:primes){
        // using this mark all not prime numbers which
are multiple
        // of i in range [l,r]

        // for(int j=i*i;j<=N;j+=i)  odd

        // 2  --> 4 6 8 10 12
        // [1e9,1e9+1e5]
        // we have to start internal loop from multiple
of i which is
        // just greater than l

        int start=max(i*i,((l+i-1)/i)*i);
        // ((l+i-1)/i)*i    number just greater than l
and divisible
        // by i

        for(int j=start;j<=r;j+=i){
            nPrime[j-l+1]=false;
         }
     }
```

```
    // tc:
    //
    // i--> inner loop (r-l+1)/i

    // total tc :
(r-l+1)/2+(r-l+1)/5+(r-l+1)/7+....
    //    (r-l+1)(1/2+1/5+1/7....)
    // O((r-l+1)loglog(r))

    vector<int> ans;

    for(int i=1;i<=r-l+1;i++){
        if(nPrime[i]){
            int num=l+i-1;
            ans.push_back(num);
        }
    }

    return ans; // all prime numbers in range [l,r]

}
```

Congruence Modulo :

Notation:

$a \equiv b \,(mod\, c)$

read as: "a is congruent to b modulo c"

meaning: a-b is divisible by c.

Note: "$a \equiv b\,(mod\, c)$" implies "c divides (a-b)" and "c divides (a-b)" implies "$a \equiv b\,(mod\, c)$".

properties:

a-b divisible by c
A+D   B+D
A*D-B*D
(a^d-b^d)=(a-b)(something)

1. if $a \equiv b\,(mod\, c)$, then $a \pm d \equiv b \pm d\,(mod\, c)$
2. if $a \equiv b\,(mod\, c)$, then $a * d \equiv b * d\,(mod\, c)$
3. if $a \equiv b\,(mod\, c)$, then $a^d \equiv b^d\,(mod\, c)$
4. if $a \equiv b\,(mod\, c)$,then $a/d \equiv b/d \;(mod\, c/gcd(c,d))$

   impt

   obviously a and b must be divisible by d

   if gcd(c,d)==1

   $a/d \equiv b/d \;(mod\, c)$

Fermat's Little Theorem :

general form for any a and prime p

$a^p \equiv$ **amodp**   p must be prime

if a%p!=0   p is prime and a is not divisible by p

a^(p-1)≡ 1 mod p   if a is not divisible by p
and p is prime

Calculate it
a^b mod p   P → prime

3^100000 mod(53)

53→ prime 3 not divisible by 53
using flt
3^52≡1mod53

3^((52*x)+rem) == 3^(52*x) *(3^rem)

3^100000 mod 53 =(3^(52*q)) * 3^4 mod 53

= 1*3^4 mod 53

a^b mod p   p→ prime a not divisible by p
a^(bmod(p-1)) mod p

Inverse using flt

(a*b)mod m = (amodm*bmodm)modm
(a/b) mod m = (amodm * a^-1 mod m) mod m

ab=1 mod m
then b is inverse of a

inverse exist only gcd(a,m)=1

a^p-1 = 1 mod p
inverse exists here
a^p-1 * a^-1 = a^-1 mod p
a^p-2 = a^-1 mod p   when p is prime and a is not divisible by p

biexm(a,p-2) = a^-1 mod p

Euler Totient Function (Phi Function):

phi (n) :- no of integers in [1,n] which are coprime to n; phi(1)=1

Properties of Phi

1 . phi(p) = p-1   p is prime

2. phi(p^k) =p^k-p^(k-1)

phi(3^2)   1 2 3 4 5 6 7 8 9

phi=(3^2) = 6

phi(p^k) = p^k-(p^k)/p=p^k-p^(k-1)

3. phi(a*b) =phi(a)*phi(b)   if and only if gcd(a,b)=1 a and b coprime

N=$p1^{a1} * p2^{a2} * p3^{a3} \quad * \quad pm^{am}$ **pi → prime**

p1^a1 p2^a2 … these are pairwise co primes

phi(N)=phi(p1^a1)*phi(p2^a2) …..

=

$(p1^{a1} - p1^{a1-1}) \quad * (p2^{a2} - p2^{a2-1}) \qquad * \quad (pm^{am} - pm^{am-1})$

p1^a1(1-1/p1)   *   p2^a2 (1-1/p2)   *   pm^am(1-1/pm)

phi(N)=N(1-1/p1)*(1-1/p2)   .. (1-1/pm)   N>=2

phi(1)=1

4 . phi(d1)+phi(d2)+phi(d3) +   +phi(dk) = N

di→ divisor of N

method 1
for single query

```
int CalcPhi(int n){
    if(n==1) return 1;
    int phi=n;
    for(int i=2;i*i<=n;i++){
        if(n%i==0){
            // i prime factor of n
            phi=phi-phi/i;

            while(n%i==0) n/=i;
        }
    }




    // 28   1 2 3 4 5
    // 7
    if(n>1) {
        phi=phi-phi/n;
    }

    return phi;
}
```

tc: sqrt(n)

Using precomputation

```
#include<bits/stdc++.h>
using namespace std;

#define int long long
const int N=1e6;
int phi[N+1];
int32_t main(){

    for(int i=1;i<=N;i++) phi[i]=i;

    for(int i=2;i<=N;i++){
        if(phi[i]==i){  // i is prime
            for(int j=i;j<=N;j+=i){
                phi[j]=phi[j]-phi[j]/i;
            }
        }
    }
    // n/2+n/3+n/5+n/7
    // tc : O(nloglogn)

    //  7     --> 7
    // phi[7]=phi[7]-phi[7]/7
    // =   7-7/7=6

    // 28       2 7
```

```
    for(int i=1;i<=20;i++) cout<<phi[i]<<" ";
    cout<<endl;


    return 0;
}
```

// when q queries are given
find phi(x) x<=1e6
q→ 1e6

i prime i 2*i 3*i 4*i 5*i

https://www.spoj.com/problems/ETF/

<u>Application of ETF to find modular inverse when m is not prime</u>

Find $a^{-1} mod\, m$

1. if a and m are not coprime inverse do not exist

2. if a and m are coprime then by euler theorem

<u>Euler Theorem</u>

$a^{\phi(m)} \equiv 1 (mod\, m)$ **if a and m are coprime.**

Note that if m is prime, this becomes Fermat's

Little Theorem! 🤓

inverse exist hence

$a^{\phi(m)-1} \equiv a^{-1}(mod\, m)$

using phi(m) we can calculate $a^{-1}(mod\, m)$

a^-1 mod m = binexm(a,phi(m)-1,mod)
for flt
a^-1 mod m = binexm(a,m-2,mod)

Lengendre's Formula:

Find the power of a prime number p, in the prime factorization of n!.

10! find the powers of 2 in 10!
5+2+1=8

$$Mp(n!) = \sum_{i=1}^{\infty} floor(n/p^{i})$$

implement by urself

Goldman Conjecture:

any even number greater than 2 can be expressed as a sum of 2 primes.

4=2+2
6=3+3
14=11+3

Prime Gap:

Wikipedia link for prime Gap

The difference between consecutive prime numbers is at most 300 for N<=1e9

https://codeforces.com/problemset/problem/584/D

3 3

n n-1 n-2 n-3 n-4 n-300
n → odd n-prime → even
prime >=n-300
prime
n-prime <=300
x=n-prime <=300 x-> even

use brute force for x to represent it as sum of two primes

Linear Congruence Equation:

ax≡b(mod c).. a,b,c given and find atleast 1 x satisfying this.

case 1:

if c is prime,

a^-1 mod c = binpow(a,c-2,c)

x≡$a^{-1}b(mod\ c)$

case 2:
c is not prime.
ax≡b (mod c)
ax-b=cy
ax+cy = b if we find x and y satisfying this, then we have found a solution.
gcd(a,c) = g.
ax≡b (mod c)
(a/g).x≡(b/g)(mod c/g).. assuming b is div by g.

Ax≡B(modC).
now, modular inverse of A and C exist… so we can it.
AX+CY = 1, if we find numbers X, Y satisfying this, we can solve it.

if gcd(a,b) = g,
then we can find numbers x and y such that
ax+by = g→ we will learn an algo for that,
theorem: bezout’s theorem.. it guarantees that x and y will always exist.

Extended euclidean algorithm:

gcd(32,20)=4→ normal euclidean algo, only gives gcd.

Now we want the gcd expressed as a ax+by

gcd(a,b) = gcd(b,a%b)

suppose g = x1.b+y1.(a%b)

a=bq+r.

b=r.q1+r1

a*y1+b*(x1-floor(a/b)*y1) = g

floor(a/b) = q

32 = 20*1+12

gcd(20,12) = 4

4= 12*2-20*1

32 = 20*1+(4+20)/2

4 = 32*2-20*3

Chinese Remainder Theorem:

x≡a1 (mod p1)

≡a2(mod p2)

≡a3(mod p3)

…

p1,p2,p3,... pn are pairwise coprime.

x≡2(mod3)
≡3(mod4)
≡1(mod5)

x = 3*4*x1+3*5*x2+4*5*x3

20x3≡2(mod 3)
15x2≡3(mod 4)
12x1≡1(mod 5)

these are 3 linear congruence equations.

modinv(20,3),modinv(15,4),modinv(12,5)

https://codeforces.com/problemset/problem/919/E

$$n \cdot a^n \equiv b \pmod{p},$$

we check for i from 0 to p-2
a^i ≡ x (mod p)---eqn(i)
k.a^i ≡ k.x (mod p)..
let us try to find k, such that this corresponds to a solution,...
k.x ≡ b(mod p)
so, k ≡ b.x^(-1) (mod p)--- eqn (ii)
a^k if divided by p,
k = b.(p-1) + k%(p-1)

$$a^k \equiv (a^{(p-1)})^b * a^{k\%(p-1)} (mod\, p)$$

now using FLT, $(a^{(p-1)})$ ≡ 1 (mod p), so raising both sides to power b,

$(a^{(p-1)})^b$ ≡ 1 (mod p)

so , a^k ≡ $a^{k\%(p-1)}$ (mod p)

So, comparing with eqn(i),

k%(p-1) = i.

Conclusion: if $a^i \equiv x(mod\, p)$ these and we find k such that,

1. k ≡ b.x^(-1) (mod p)... where x is the rem when a^i is divided by p.
2. k ≡ i (mod p-1)

then k.$a^k \equiv b(mod\, p)$.

$k.a^k \equiv k.x(mod\, p) \equiv b\,(mod\, p)$

[as k = (p-1)*q+i → i = k+(-q)*(p-1)]

Now this is same as solving 2 congruence equations. So use CRT!.

let k ≡ M (mod p).. writing b.x^-1 as M

and k ≡ i (mod p-1).

we know,

(p-1).(p-1)≡ 1 (mod p).(p^2-2p+1)

so modinv(p-1,p) is p-1.

and modinv(p,p-1) is 1 (as p.1 ≡ 1 mod(p-1))

so, using CRT

k = M.(p-1).modinv(p-1,p)+i.p.modinv(p,p-1)

= M.(p-1)*(p-1) + i.p

now,

general value of k =

M.(p-1)*(p-1) + i.p + Q.(p*(p-1)),.. where Q is an integer,.. So we have to find how many such Q are there so that k lies in the range 1<=k<=x (x is given in question).

let y = 2+q.3… find number of y less than 23
23%3→ 2
so, 2,5,8,11,14,17,20,23

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double ld;
#define fastio                          \
    ios_base::sync_with_stdio(false); \
    cin.tie(NULL);                      \
    cout.tie(NULL)
#define max3(a, b, c) max(max(a, b), c)
#define max4(a, b, c, d) max(max(a, b), max(c, d))
#define fr(i, n) for (ll i = 0; i < n; i++)
ll gcd(ll a, ll b)
{
    return b == 0 ? a : gcd(b, a % b);
}
long long binpow(long long a, long long b, long long m)
{
    a %= m;
    long long res = 1;
    while (b > 0)
    {
        if (b & 1)
            res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
```

```
    }
    ll modinv(ll a, ll p)
    {
        return binpow(a, p - 2, p);
    }
    int main()
    {
        fastio;
        ll a, b, p, x;
        cin >> a >> b >> p >> x;
        ll res = 0;
        for (ll i = 0; i < p - 1; i++)
        {
            ll modd = p * (p - 1);
            ll z = binpow(a, i, p);
            ll M = (b * modinv(z, p)) % p;
            ll k = (((p * i) % modd) + (((p - 1) * (p - 1)) % modd * M) %
modd) % modd;
            ll complete_cycles = x / modd;
            res += complete_cycles;
            ll remain = x % modd;
            if (remain >= k)
                res++;
        }
        cout << res << "\n";
    }
```