1. Euler function

$$\varphi(n) = \mathrm{len}(\{1, 2, 3..n\}, \gcd(k, n) = 1)$$

Phi is multiplicative function

$$\varphi(ab) = \varphi(a)\varphi(b)$$

$\varphi(n) = p_1^{k_1-1}(p_1 - 1)p_2^{k_2-1}(p_2 - 1)...$ Where p is prime number from factorization n.

Example

$$\varphi(54) = \varphi(2 * 3^3) = \varphi(2) * \varphi(3^3)$$

Euler's theorem If a and p is coprime, than $a^{\varphi(n)} \equiv 1 \bmod n$.

2. Modular arithmetics

2.1. Congruence

We say that 3 is congruent to 15 by modulo 12, written $15 \equiv 3 (\bmod 12)$

Coprime two integers GCD is 1.

2.2. Fermat's little theorem

Special case of euler theorem.

Theorem If p is a prime number, then for any integer a, the number $a^p - a$ is an integer multiple of p

$$a^p \equiv a \bmod p$$

If a is coprime to p.

$$a^{p-1} \equiv 1 \bmod p$$

2.3. Primitive root modulo

$\boldsymbol{a|b}$ a divides b=> b/a = 0

Primitive root modulo n g is called primitive root modulo p if every a coprime number to n is congruent to a power of g modulo p.

$$\forall a \in Z : \gcd(a, p) = 1, \exists n : g^n = a \longrightarrow g \text{ is primitive root modulo}$$

N is not required to be prime. G is a *primitive root modulo* n if and only if g is a generator of the multiplicative group of integers modulo n.

2.4. P Group

2.4.1. How to check that group is cyclic

2.4.2. Theorem to check generator in p group

$\alpha \in Z_p^*$ is a generator of Z_p^* if and only if

$$\alpha^{\frac{p-1}{q}} \neg \equiv 1 \bmod p$$

For all primes q such that $q|(p-1)$

Task

Task find the all generators of Z_{11}^*

Let's begin with $p - 1 = 10$, 10 = 2*5.

Generator check condition for each divider of $(p - 1)$:

- $\alpha^5 \neg \equiv 1 \bmod 11$
- $\alpha^2 \neg \equiv 1 \bmod 11$

Solution is to check each element in group to match conditions.

2.4.3. How to count generators in group

Theorem let p be prime, that Z_p^* contains exactly $\varphi(p - 1)$ generators.

2.4.4. How to find generator

2.4.5. Discrete logarithm in p

If B $\in Z_p^*$, then B $= g^x$ for some unique $0 \le x \le p - 2$. X is called the discrete logarithm of B to base g.

$g^x = \mathrm{B} \Rightarrow \log_g \mathrm{B} = x$ in Z

Problem find the integer x, such that

$$\log_g \mathrm{B} \text{ in } Z_p^*$$

The naive approach is exhaustive search: compute $g^x, g^2x, ...$ until B is obtained.

2.5. N Group

2.5.1. Theorem to check generator in n group

For $n \ge 1$, we consider Z_n^*

$$Z_n^* = \{k \in \{1, ..., n\} / \gcd(k, n) = 1\}$$

$$\mathrm{len}(Z_n^*) = \varphi(n)$$

Z_n^* **is cyclic**(has at least one generator) when:

1. n=2 or 4
2. $n = p^x, x \in \{1, 2...\}$
3. $n = 2p^x, x \in \{1, 2...\}$

Theorem to check generator

Assume Z_n^* is cyclic. $\alpha \in Z_n^*$ is a generator if and only if

$$\alpha^{\frac{\varphi(n)}{p}} \neg \equiv 1 \bmod n$$

For each prime p divisor of $\varphi(n)$

2.5.2. How to count generators in group

Theorem if Z_n^* is cyclic, then it has $\pi(\pi(n))$ generators.

2.5.3. Discrete logarithm in n

If B $\in Z_n^*$, then B $= g^x$ for some unique $0 \le x \le p - 2$. X is called the discrete logarithm of B to base g.

Example

Find $\log_{13} 47$ in Z_{50}^*

1. Check Z_{50}^* is cyclic(e.g. has generators)
2. Check g 13 is generator.(requires find $\varphi(n)$)
3. Start to calculate elements.(exhaustive search)

3. Algorithms for computing discrete algorithms

1. Brute force
2. Shank's baby-step giant-step method

3.1. Some equations

$$3\mathrm{B} \bmod 13 = 1 \Rightarrow 3B \equiv 1 \bmod 13$$

$$(ab) \bmod m = [(a \bmod m)(b \bmod m)] \bmod m$$

3.2. Core Equation for DH

$$y_1 = (a^{x_1} \bmod p)^{x_2} \bmod p = a^{x_1 x_2} \bmod p$$

Some modular arithmetics to proof:

$$y_1 = (a^{x_1} \bmod p)^{x_2} \bmod p = ((a \bmod p)^{x_1} \bmod p)^{x_2} \bmod p =$$

Reduce extra modulo due to modulo properties:

$$(a \bmod p)^{x_1} \bmod p = a^{x_1} \bmod p$$

Continue:

$$= (a \bmod p)^{x_1 x_2} \bmod p$$

$$= a^{x_1 x_2} \bmod p$$

3.3. Factorization problem

For example RSA relies on difficulty of factoring the product of two large prime numbers.

But for this need to determine or find large prime number.

Determine if number is prime:

1. Simple methods(advanced brute force, without 2,3 and maybe some memoization,etc)
2. Probabilistic tests(all primes + some non primes - never FN, but sometimes FP)