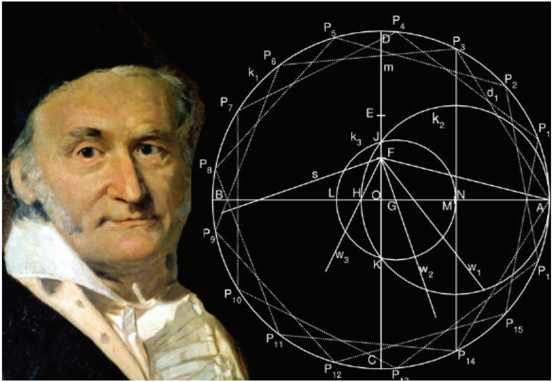
Elementary Number Theory



Contents

- Number Sets
- GCD and Euclidean Algorithms
- Binary GCD Algorithm
- Modular Addition and Multiplication
- Multiplicative Inverse
- Modular Exponentiation

Number Sets

- We represent the set of integers as $\mathcal{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
- We denote the set of positive integers modulo n as $\mathcal{Z}_n = \{0, 1, \ldots, n-1\}$
- Elements of $\mathcal{Z}_n$ can be thought of as equivalency classes
- For $n \geq 2$, every integer in $a \in \mathcal{Z}$ maps into one of the elements $r \in \mathcal{Z}_n$ using the division law $a = q \cdot n + r$ which is represented as $a \equiv r \pmod{n}$

Number Sets

- Let $\mathcal{Z}_5 = \{0, 1, 2, 3, 4\}$
- Therefore, 0 represents the infinite set of negative and positive integers: $0 \equiv \{\ldots, -15, -10, -5, 0, 5, 10, 15 \ldots\}$
- Similarly, 1 represents the infinite set of negative and positive integers: $1 \equiv \{\ldots, -14, -9, -4, 1, 6, 11, 16, \ldots\}$

Number Sets

- The symbol $\mathcal{Z}_n^*$ represents the set of positive integers that are less than n and relatively prime to n
- If $a \in \mathcal{Z}_n^*$, then $\gcd(a, n) = 1$
- When $n = p$ is prime, the set would be $\mathcal{Z}_p^* = \{1, 2, \ldots, p-1\}$
- When n is not a prime, the number of elements that are less than n and relatively prime to n is given as $\phi(n) = |\mathcal{Z}_n^*|$
- Euler's Phi (totient) Function $\phi(n)$ is defined as the number of numbers in the range $[1, n-1]$ that are relatively prime to n

Greatest Common Divisor

- Given two positive integers a and b, their greatest common divisor (GCD) is denoted as $g = \gcd(a, b)$
- We can compute $\gcd(a, b)$ from the prime factorizations of a and b

$$\begin{aligned} a &= p_1^{e_1} \cdot p_2^{e_2} \cdots p_r^{e_r} \\ b &= p_1^{f_1} \cdot p_2^{f_2} \cdots p_r^{f_r} \end{aligned}$$

- Zero exponents are used to make the set of primes $p_1, p_2, \ldots, p_r$ the same for both a and b
- The GCD is computed as

$$\gcd(a, b) = p_1^{\min(e_1, f_1)} \cdot p_2^{\min(e_2, f_2)} \cdots p_r^{\min(e_r, f_r)}$$

- However, integer factorization algorithms require exponential time

GCD and Euclidean Algorithm

- The most commonly used algorithm for computing the greatest common divisor of two integers is the Euclidean algorithm
- The Euclidean algorithm is based the property

$$\gcd(a, b) = \gcd(b, a - Q \cdot b)$$

  where Q is the integer division $Q = \lfloor a/b \rfloor$

- By applying this reduction rule repeatedly, the Euclidean algorithm obtains $\gcd(a, b) = \gcd(g, 0) = g$
- For example, to compute $\gcd(56, 21)$, we perform the iterations

$$\begin{array}{rclcl} \gcd(56,21) & \rightarrow & \lfloor 56/21 \rfloor = 2 & \rightarrow & \gcd(21, 56 - 2 \cdot 21) \\ \gcd(21,14) & \rightarrow & \lfloor 21/14 \rfloor = 1 & \rightarrow & \gcd(14, 21 - 1 \cdot 14) \\ \gcd(14,7) & \rightarrow & \lfloor 14/7 \rfloor = 2 & \rightarrow & \gcd(7, 14 - 2 \cdot 7) \\ \gcd(7,0) & = & 7 & & \end{array}$$

GCD and Euclidean Algorithm

- Given the positive integers a and b with $a > b$, the Euclidean algorithm computes the greatest common divisor g in $O(k)$ steps where k is the number of bits in a

function EA(a, b)
Input: a, b with $a > b$
Output: $g = \gcd(a, b)$

while $b \neq 0$
    $Q \leftarrow a/b$
    $r \leftarrow a - Q \cdot b$
    $a \leftarrow b$
    $b \leftarrow r$
return a

GCD and Euclidean Algorithm Example

- Given $a = 117$ and $b = 45$, the Euclidean Algorithm computes

a	b	Q	r	new a	new b
117	45	2	27	45	27
45	27	1	18	27	18
27	18	1	9	18	9
18	9	2	0	9	0
9	0				

- The EA function returns 9 since $\gcd(117, 45) = 9$

Extended Euclidean Algorithm

- Another important property of the GCD is that, if $\gcd(a, b) = g$, then there exists integers s and t such that

$$s \cdot a + t \cdot b = g$$

- We can compute s and t using the extended Euclidean algorithm by working back through the remainders in the Euclidean algorithm
- For example, to find $\gcd(833, 301) = 7$, we write

$$\begin{aligned} 833 - 2 \cdot 301 &= 231 \\ 301 - 1 \cdot 231 &= 70 \\ 231 - 3 \cdot 70 &= 21 \\ 70 - 3 \cdot 21 &= 7 \\ 21 - 3 \cdot 7 &= 0 \end{aligned}$$

Extended Euclidean Algorithm

- Since $g = 7$, we start with the 4th equation and plug in the remainder value from the previous equation to this equation, and then move up

$$\begin{aligned} 70 - 3 \cdot (231 - 3 \cdot 70) &= 7 \\ 10 \cdot 70 - 3 \cdot 231 &= 7 \\[1em] 10 \cdot (301 - 1 \cdot 231) - 3 \cdot 231 &= 7 \\ 10 \cdot 301 - 13 \cdot 231 &= 7 \\[1em] 10 \cdot 301 - 13 \cdot (833 - 2 \cdot 301) &= 7 \\ -13 \cdot 833 + 36 \cdot 301 &= 7 \end{aligned}$$

- Therefore, we find $s = -13$ and $t = 36$
- This implies $g = s \cdot a + t \cdot b \Rightarrow 7 = (-13) \cdot 833 + 36 \cdot 301$

Computation of Multiplicative Inverse

- The EEA allows us to compute the multiplicative inverse of an integer a modulo another integer n, if $\gcd(a, n) = 1$
- The EEA obtains the identity $g = s \cdot a + t \cdot b$ which implies

$$\begin{aligned} s \cdot a + t \cdot n &= 1 \\ s \cdot a &= 1 \pmod{n} \\ a^{-1} &= s \pmod{n} \end{aligned}$$

For example, $\gcd(23, 25) = 1$, and the extended Euclidean algorithm returns $s = 12$ and $t = 11$, such that

$$1 = 12 \cdot 23 - 11 \cdot 25$$

therefore $23^{-1} = 12 \pmod{25}$

Fermat's Little Theorem

- Theorem: If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} = 1 \pmod p$
- For example, $p = 7$ and $a = 2$, we have $a^{p-1} = 2^6 = 64 = 1 \pmod 7$
- FLT can be used to compute the multiplicative inverse if the modulus is a prime number
$$a^{-1} = a^{p-2} \pmod p$$
since $a^{-1} \cdot a = a^{p-2} \cdot a = a^{p-1} = 1 \bmod p$
- The converse of the FLT is not true: If $a^{n-1} = 1 \pmod n$ and $\gcd(a, n) = 1$, then n may or may not be a prime.
- Example: $\gcd(2, 341) = 1$ and $2^{340} = 1 \pmod{341}$, but 341 is not prime: $341 = 11 \cdot 31$

Euler's Phi Function

- Euler's Phi (totient) Function $\phi(n)$ is defined as the number of numbers in the range $[1, n-1]$ that are relatively prime to n
- Let $n = 7$, then $\phi(7) = 6$ since for all $a \in [1, 6]$, we have $\gcd(a, 7) = 1$
- If p is a prime, $\phi(p) = p - 1$
- For a positive power of prime, we have $\phi(p^k) = p^k - p^{k-1}$
- If n and m are relatively prime, then $\phi(n \cdot m) = \phi(n) \cdot \phi(m)$
- If all prime factors of n is known, then $\phi(n)$ is easily computed:

$$\phi(n) = n \cdot \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

Euler's Theorem

- Theorem: If $\gcd(a, n) = 1$, then $a^{\phi(n)} = 1 \pmod{n}$
- Example: $n = 15$ and $a = 2$, we have $2^{\phi(15)} = 2^8 = 256 = 1 \bmod 15$
- Euler's theorem can be used to compute the multiplicative inverse for any modulus:

$$a^{-1} = a^{\phi(n)-1} \pmod{n}$$

  however, this requires the computation of the $\phi(n)$ and therefore the factorization of n
- To compute $23^{-1} \bmod 25$, we need $\phi(25) = \phi(5^2) = 5^2 - 5^1 = 20$, and therefore,

$$23^{-1} = 23^{20-1} = 23^{19} = 12 \pmod{25}$$

Representing Numbers mod n

- The elements of $\mathcal{Z}_n$ can be represented in two distinct ways: the Least Positive (LP) representation
  the Least Magnitude (LM) representation
- The **Least Positive** representation uses
  $\mathcal{Z}_n = \{0, 1, 2, \ldots, n-1\}$
- Example: the least positive representation mod 10
  $\mathcal{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Example: the least positive representation mod 11
  $\mathcal{Z}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

Representing Numbers mod n

- The **Least Magnitude** representation for n is odd
  $\mathcal{Z}_n = \{-(n-1)/2, \ldots, -2, -1, 0, 1, 2, \ldots, (n-1)/2\}$
- Example: the least magnitude representation mod 11
  $\mathcal{Z}_{11} = \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$
- The **Least Magnitude** representation for n is even
  Either: $\mathcal{Z}_n = \{-n/2+1, \ldots, -2, -1, 0, 1, 2, \ldots, n/2\}$
  Or: $\mathcal{Z}_n = \{-n/2, \ldots, -2, -1, 0, 1, 2, \ldots, n/2-1\}$
- Example: the least magnitude representation mod 10
  Either: $\mathcal{Z}_{10} = \{-4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$
  Or: $\mathcal{Z}_{10} = \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4\}$
- The LM property: a is LM mod n if $|a| \leq |n-a|$

Modular Arithmetic Operations

- Given a positive odd n, how does one compute modular additions, subtractions, multiplications, and exponentiations?
- $s = a + b \pmod{n}$ is computed in two steps: 1) add, 2) reduce
- If $a, b < n$ to start with, then the reduction step requires a subtraction

$$\text{if} \quad s > n \ , \quad \text{then} \quad s = s - n$$

- $s = a - b \pmod{n}$ is computed similarly: 1) subtract, 2) reduce
- The **least positive** representation is often preferred
- The least positive representation uses unsigned arithmetic
- Negative numbers are brought to the range $[0, n-1]$

Modular Multiplication

- Modular Multiplication $a \cdot b \pmod{n}$ can be computed in two steps:
  - Multiplication step: $c \leftarrow a \cdot b$
  - Reduction step: $r \leftarrow c \bmod n$
- The reduction step may require division by n to obtain the remainder

$$a \cdot b = c = Q \cdot n + r$$

- However, we do not need the quotient!
- The division by n is an expensive operation
- The Montgomery Multiplication: A new algorithm for performing modular multiplication that does not require division by n

Modular Exponentiation

- The computation of $b = a^e \pmod{n}$: Perform the steps of the exponentiation a^e, reducing numbers at each step mod n
- Reduction is required, otherwise a^e doubles in size at each size
- Exponentiation algorithms: binary method, m-ary methods, sliding windows, power tree method, factor method
- The binary method is the most commonly used algorithm
- The binary method uses the binary expansion of the exponent $e = (e_{k-1}e_{k-2} \cdots e_1 e_0)$, and performs squaring and multiplication operations at each step

Modular Exponentiation with Binary Method

- Given the inputs a, n, and $e = (e_{k-1}e_{k-2}\cdots e_1e_0)_2$, the binary method computes $b = a^e \pmod{n}$ as follows

1: **if** $e_{k-1} = 1$ **then** $b \leftarrow a$ **else** $b \leftarrow 1$
2: **for** $i = k-2$ **downto** 0
2a: $b \leftarrow b \cdot b \pmod{n}$
2b: **if** $e_i = 1$ **then** $b \leftarrow b \cdot a \pmod{n}$
3: **return** b

- $e = (1\ 10111) = 55$
- $k = 6$
- $e_5 = 1 \Rightarrow b \leftarrow a$

$i \rightarrow$	4	3	2	1	0
$e_i \rightarrow$	1	0	1	1	1
Step 2a	a^2	a^6	a^{12}	a^{26}	a^{54}
Step 2b	a^3	a^6	a^{13}	a^{27}	a^{55}