Mathematics of Network Security

Dr. Risala Tasin Khan

Why Mathematics is Needed in Security?

- Modern security is heavily based on some areas of mathematics, including number theory, linear algebra, and algebraic structures.
- Security algorithms are designed around computational hardness assumptions using mathematical functions and formula, making such algorithms hard to break in practice by any adversary.
- A list of mathematical fields used in network security is given below.

❑ **Number Theory:**

- It is used to understand why and how RSA works. Some algorithms use number theory for the difficulty of factoring large numbers as their basis.

❑ **Group theory:**

- Group theory is used to understand why and how El Gamal works.

Cont…

Probability theory:

- It is used in analyzing many kinds of ciphers to better understand what "statistical security" means.

❑ **Algebraic structure:**

- The theory of finite fields is used in multiparty computation.

❑ **Linear Algebra:**

- Lagrange interpolation is used in Shamir's Secret Sharing Scheme. Some linear operations are also used in AES.

What does it mean-?

1. Z The set of integers, denoted by Z, contains all integral numbers (with no fraction) from negative infinity to positive infinity

2. Z^+ Z+ = {1, 2, 3, ……., +} Set of all positive integers ranging from 1 to +infinity

3. Z^- -z = {-, ……., -3, -2, -1} Set of all negative integers ranging from -1 to -infinity

4. $Z^{Non\text{-}Neg}$ ZNon-Neg = {0, 1, 2, 3, ……., +}

   set of non neg integers ranging from 0 to +infinity

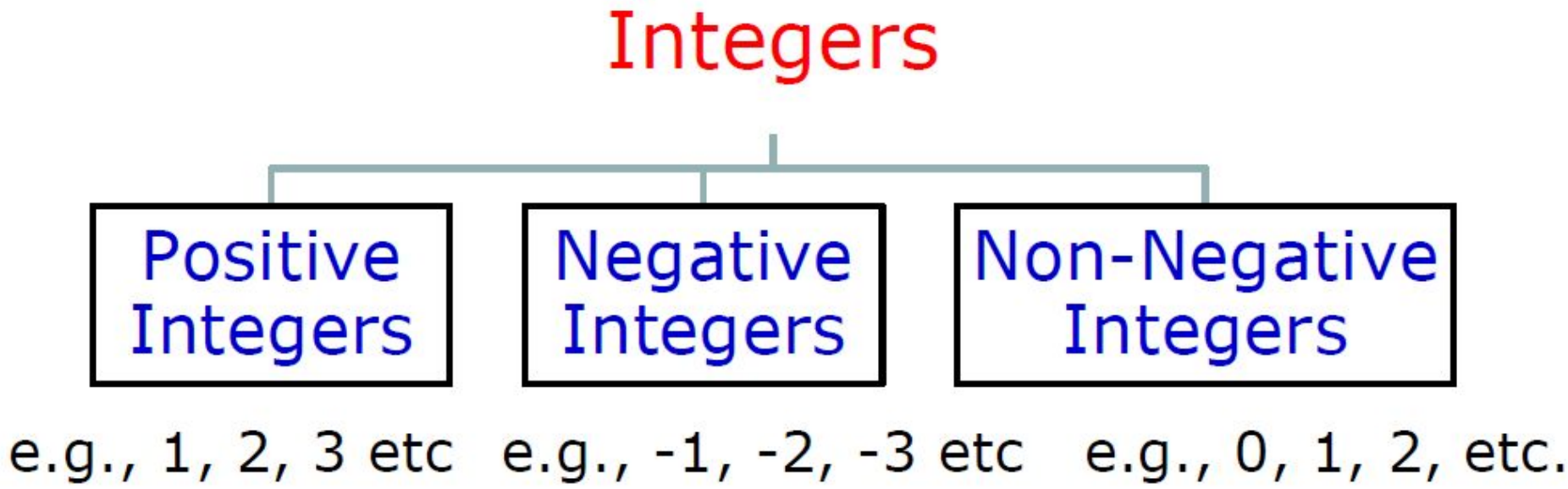
   ZNon-Pos = {-, ……, -3, -2, -1, 0}

5. Z_n Zn = {0, 1, 2, 3, ……., n-1} Set of non-negative integers ranging from 0 to (n-1)
   (Set of additive inverse in n modulus)

6. Z^*_n Z*10 = {1, 3, 7, 9}
   (Set of Multiplicative inverse in n modulus

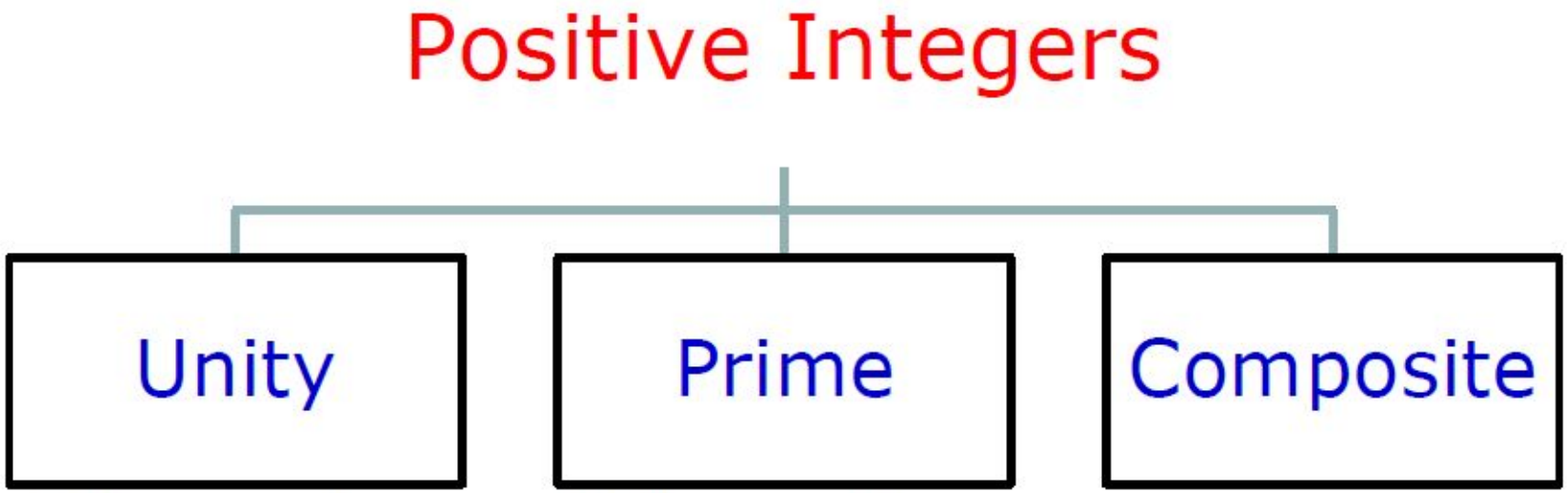
Types of Integers

- Any integer can fall into three categories.



Types of Positive Integers

- Any positive integer can fall into three categories.



It has only one divisor, i.e., itself, e.g., 1

It has only two divisors, 1 and itself, e.g., 2, 3, 13 etc.

It has more than two divisors, e.g., **6** (four divisors (1, 2, 3, 6), **9** (three divisors 1, 3, 9) etc.

Set of Integers

In integer arithmetic, we use a **set** and a few **operations**.

- Though you are familiar with this set and the corresponding operations, but they are reviewed here to create a background for modular arithmetic.
- The set of integers, denoted by **Z**, contains all integral numbers (with no fraction) from negative infinity to positive infinity.

$$\mathbf{Z} = \{\ .\ .\ .\ , -2, -1, 0, 1, 2, \ .\ .\ .\ \}$$

Figure: The set of integers

Some subset of integers are listed below:

$Z^+ = \{1, 2, 3, \ldots\ldots\ldots\ldots, +\infty\}$ Or, $Z^+ = \{x \in Z \mid x > 0\}$

Set of all positive integers, ranging from 1 to +∞

(Set of all natural numbers)

$Z^- = \{-\infty, \ldots\ldots\ldots\ldots, -3, -2, -1\}$

Set of all negative integers, ranging from -1 to -∞

$Z^{Non\text{-}Neg} = \{0, 1, 2, 3, \ldots\ldots\ldots, +\infty\}$

Set of all non-negative integers ranging from 0 to +∞

(Set of whole numbers)

Some subset of integers are listed below (.....):

$$Z^{Non\text{-}Pos} = \{-\infty, ..., -3, -2, -1, 0\}$$

Set of non-positive integers ranging from 0 to -∞

$$Z_n = \{0, 1, 2, 3, ..., n-1\}$$

Set of non-negative integers ranging from 0 to (n-1)

(Set of additive inverse in n modulus)

$$Z^*_{10} = \{1, 3, 7, 9\}$$

Set of multiplicative inverse in 10 modulus

Binary Operations

A **binary operation** takes two inputs (e.g. ***a*** and ***b***) and creates one output (e.g. ***c***).

- In cryptography, we are interested in three binary operations applied to the set of integers: addition, subtraction and multiplication.

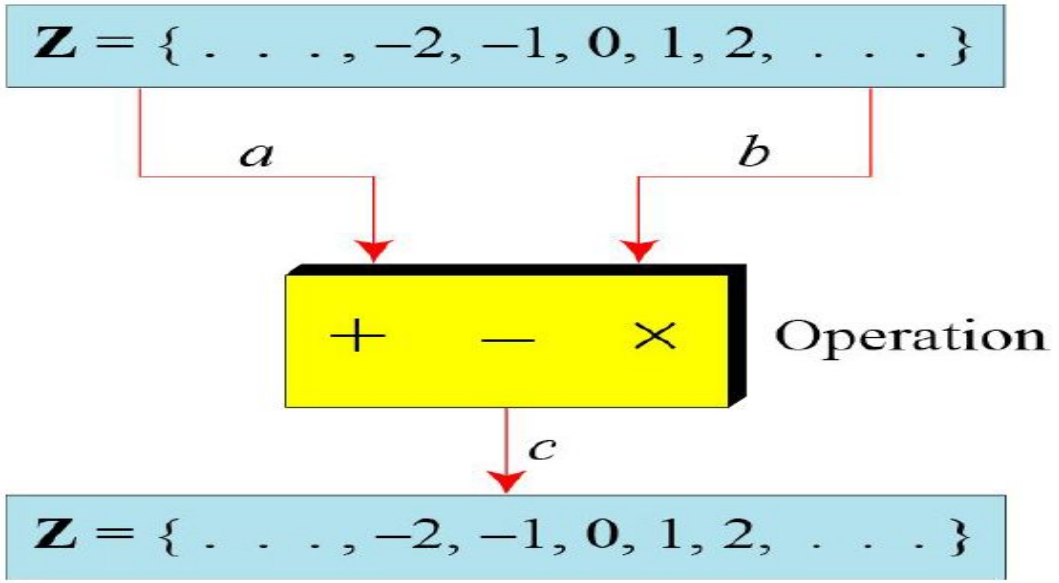


Figure: Three binary operations for the set of integers

Binary Operations

- The following examples shows the results of the three binary operations on two integers.
  - Because each input can be either positive or negative, we can have four cases for each operation.

Add:	$5 + 9 = 14$	$(-5) + 9 = 4$	$5 + (-9) = -4$	$(-5) + (-9) = -14$
Subtract:	$5 - 9 = -4$	$(-5) - 9 = -14$	$5 - (-9) = 14$	$(-5) - (-9) = +4$
Multiply:	$5 \times 9 = 45$	$(-5) \times 9 = -45$	$5 \times (-9) = -45$	$(-5) \times (-9) = 45$

Integer Division

In integer arithmetic, if we divide an integer a by a positive integer d, we can get two integers- one is called quotient q and another is called remainder r where $0 \leq r < d$.

- The relationship between these four integers is given below:

$$a = d \times q + r$$

  - d is called the divisor
  - a is called the dividend
  - q is called the quotient [it is expressed by the notation $q = a \text{ div } d$]
  - r is called the remainder [it is expressed by the notation $r = a \bmod d$]

Note:

- **Division is not a binary operation**, because it produces two output instead of one (*q* and *r*). We can call it division

When we use the above division relationship in cryptography, we impose two restrictions:

1. **The divisor be a positive integer (i.e. d>0)**
2. **The remainder be a non-negative integer (i.e. r>=0)**

- Figure below illustrate this fact.

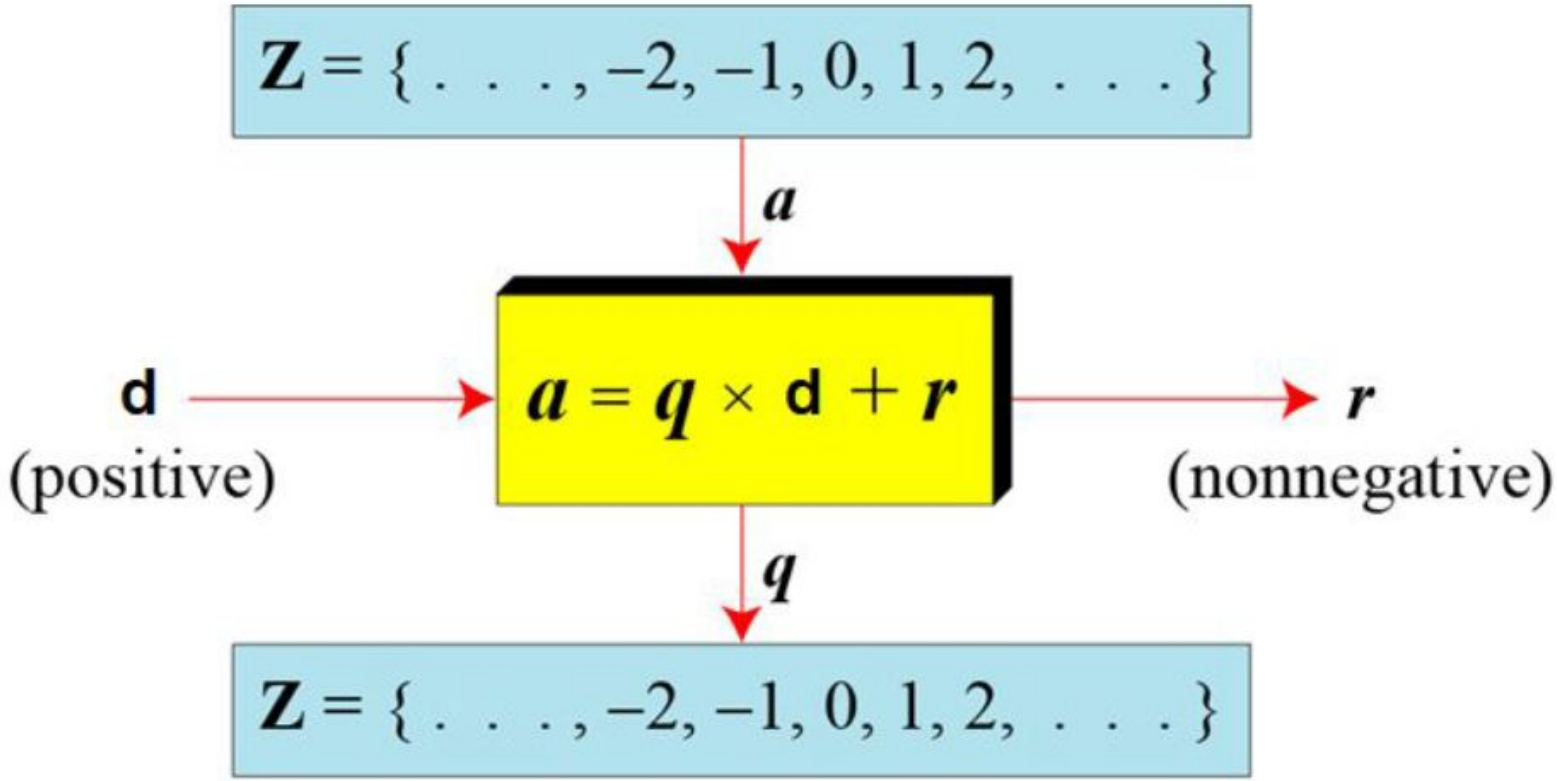


Figure: Division algorithm for integers

Example-1:

- Assume that a = 255 and d = 11. We can find q = 23 and r = 2 using the division algorithm a = d × q + r.

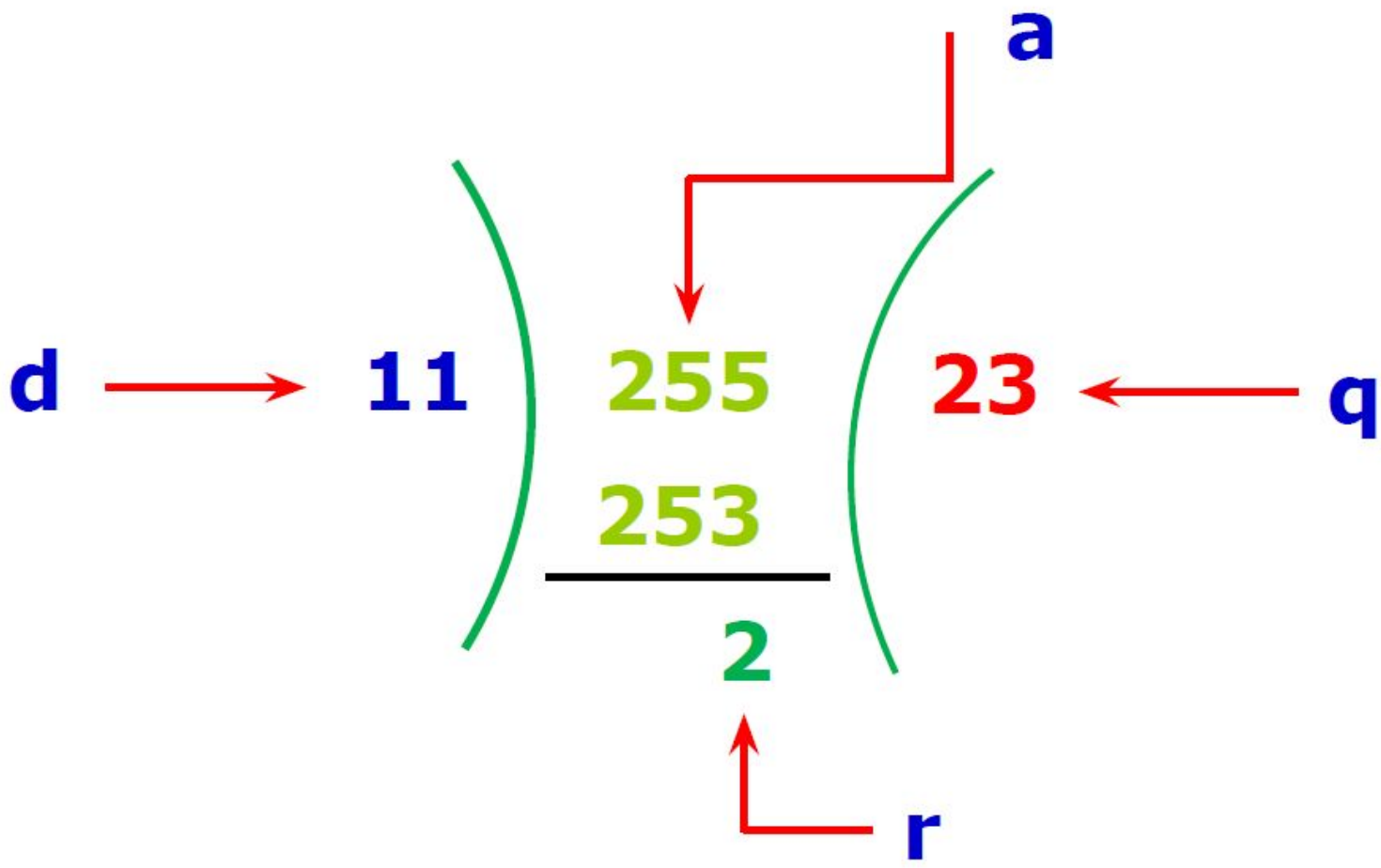


Figure: Finding the quotient and the remainder

Divisibility

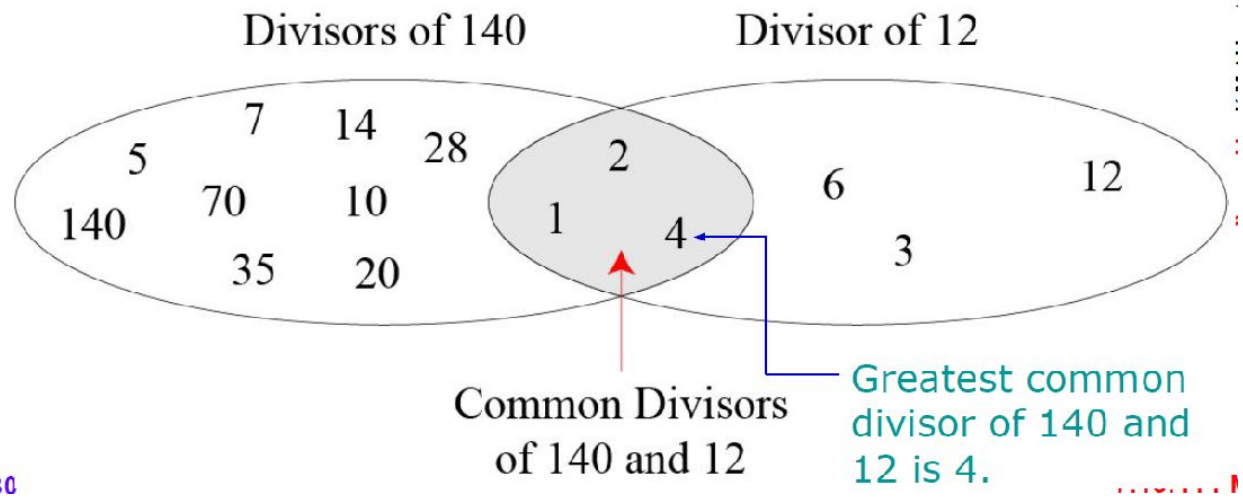
- If ***a*** and ***b*** are two integers where ***a≠0***, we can say that ***a*** **divides** ***b*** (or, ***b*** is divisible by ***a***) if there exists an integer ***k*** such that ***b*** **=** ***ak***.
  - The statement ***a*** divides ***b*** is written as ***a*** | ***b***.
  - The statement ***a*** does not divides b is written as ***a*** ∤ ***b***.
  - When ***a*** divides ***b*** we say that ***a*** is a factor or divisor of ***b*** and that ***b*** is a multiple of ***a***.
  - If ***a*** | ***b***, then ***b*** / ***a*** is an integer called quotient.
  - If ***a*** ∤ ***b***, ***b*** % ***a*** is an integer called remainder.

- **Example:**
  - 4 | 28 because 28 is a multiple of 4, i.e., 28 = 4 × 7
  - 4 ∤ 30, because 30 is not a multiple of 4.

Greatest Common Divisor (GCD)

The greatest common divisor (GCD) of two positive integers is the largest integer that can divide both integers.

- GCD is often needed in cryptography.
- Two positive integers may have many common divisors, but only one is the greatest of them.
  - For example, the common divisors of 12 and 140 are: 1, 2, and 4. However, the greatest common divisor is 4.



Least Common Multiple (LCM)

The least common multiple (LCM) of two positive integers ***a*** and ***b*** is the smallest positive integer that is divisible by both ***a*** and ***b***.

- The least common multiple of ***a*** and ***b*** is denoted by ***lcm(a,b).***
  - For example, lcm(4,6)=12; lcm(3,7)=21
- We can determine lcm using a variety of methods, e.g., by prime factorizations.
- Given ***a*** and ***b*** be two positive integers. Determine the prime factors of both ***a*** and ***b***.

$$a = p_1^{a_1} \times p_2^{a_2} \times p_3^{a_3} \ldots\ldots \times p_n^{a_n}$$

$$b = p_1^{b_1} \times p_2^{b_2} \times p_3^{b_3} \ldots\ldots \times p_n^{b_n}$$

$$lcm(a,b) = p_1^{\max(a_1,b_1)} \times p_2^{\max(a_2,b_2)} \times p_3^{\max(a_3,b_3)} \times \ldots\ldots\ldots \times p_n^{\max(a_n,b_n)}$$

Example:

$a = 60 = 2^2 \times 3^1 \times 5^1$

$b = 54 = 2^1 \times 3^3 \times 5^0$

$lcm(a,b) = 2^2 \times 3^3 \times 5^1 = 540$

GCD Using Euclidean Algorithm

Finding the GCD of two positive integers by listing all common divisors is not practical when the two integers are large.

- More than 2000 years ago, a great mathematician named **Euclid** developed an algorithm that can find the GCD of two large positive integers.
- The Euclidian algorithm is based on the two facts:

Fact 1:

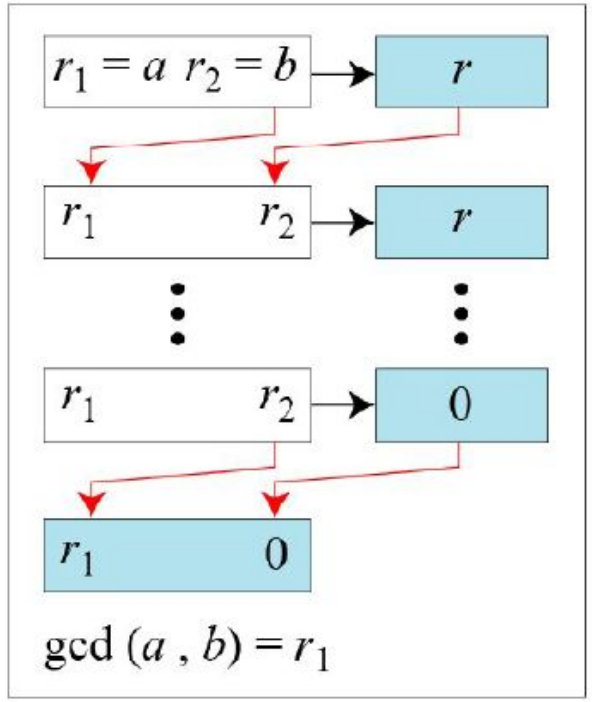
Between the two given integer ***a*** and ***b***, if the 2nd integer is zero, then **gcd (a, 0) = a**. **Example**: gcd (5, 0) = 5

Fact 2:

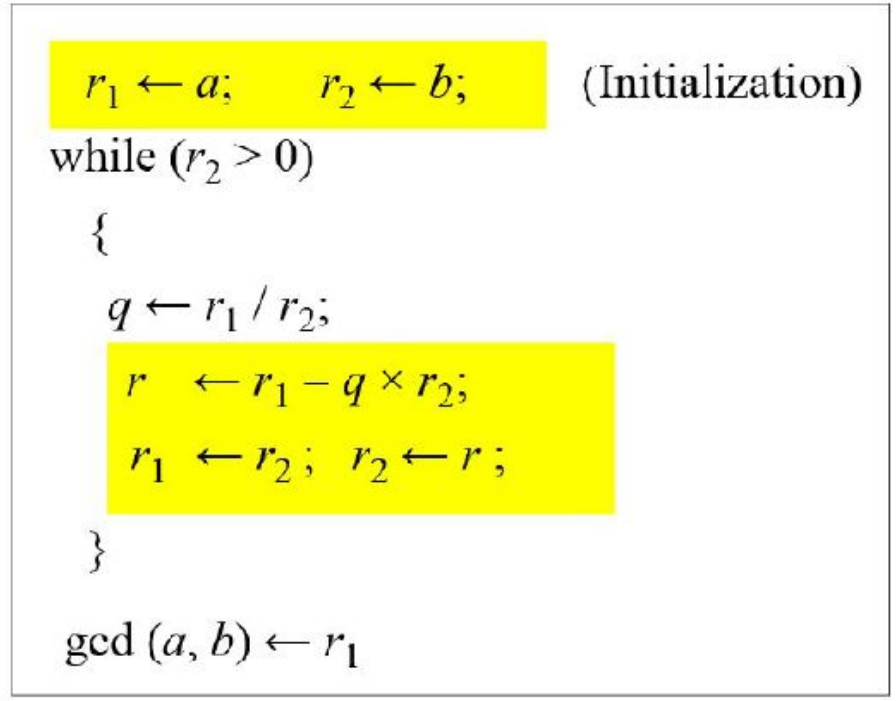
When both integer is positive, then **gcd (a, b) = gcd (b, r)**, where **r** is the remainder of dividing ***a*** by ***b*** (here the value of first and second integer is changed until the second integer becomes zero.

GCD Using Euclidean Algorithm

Figure below shows how we use Fatct-1 and Fact-2 to calculate **gcd (a, b)** using Euclidean algorithm.



a. Process



b. Algorithm

Figure: Euclidean Algorithm

Note:

When gcd (a, b) = 1, we say that ***a*** and ***b*** are **relatively prime** or they are **coprime.**

Example-1

Find the greatest common divisor of 2740 and 1760.

Solution:

We have gcd (2740, 1760) = 20.

q	r_1	r_2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

Example-2

Find the greatest common divisor of 25 and 60.

Solution:

We have gcd (25, 60) = 5.

q	r_1	r_2	r
0	25	60	25
2	60	25	10
2	25	10	5
2	10	5	0
	5	0	

Note:

The above example shows that it does not matter if the first number is smaller than the second number. We immediately get our correct ordering gcd(60, 25).

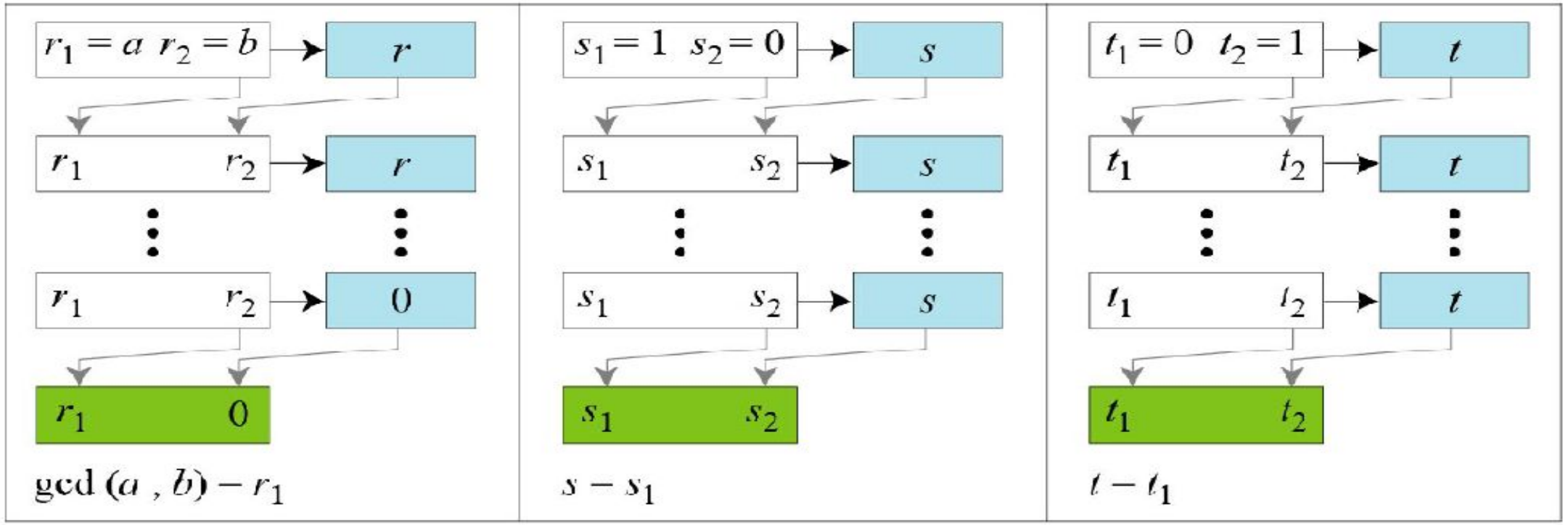
Extended Euclidean Algorithm

Given two integers ***a*** and ***b***, we often need to find other two integers, ***s*** and ***t***, such that

$$s \times a + t \times b = \gcd(a, b)$$

- The extended Euclidean algorithm can calculate the **gcd (a, b)** and at the same time calculate the value of ***s*** and ***t***.
- Using extended Euclidean algorithm, we also can find the solutions to the linear Diophantine equations of two variables, an equation of type **ax + by = c**.

Extended Euclidean Algorithm



a. Process

Figure: Extended Euclidean algorithm, part a: Process

Note:

- Figure shows that the extended Euclidean algorithm uses the same number of steps as the Euclidean algorithm, however, in each step, we use three sets of calculations and exchanges instead of one. Here, three sets of variables are used: ***r***'s, ***s***'s and ***t***'s

Extended Euclidean Algorithm

$r_1 \leftarrow a;$ $r_2 \leftarrow b;$
$s_1 \leftarrow 1;$ $s_2 \leftarrow 0;$
$t_1 \leftarrow 0;$ $t_2 \leftarrow 1;$ (Initialization)

while $(r_2 > 0)$

{

$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$
$r_1 \leftarrow r_2;$ $r_2 \leftarrow r;$ (Updating r's)

$s \leftarrow s_1 - q \times s_2;$
$s_1 \leftarrow s_2;$ $s_2 \leftarrow s;$ (Updating s's)

$t \leftarrow t_1 - q \times t_2;$
$t_1 \leftarrow t_2;$ $t_2 \leftarrow t;$ (Updating t's)

}

gcd $(a, b) \leftarrow r_1;$ $s \leftarrow s_1;$ $t \leftarrow t_1$

b. Algorithm

Figure: Extended Euclidean algorithm, part b: Algorithm

Example-1

Given a = 161 and b = 28, find gcd (a, b) and the values of s and t such that gcd(a, b) = s × a + t × b.

$$\mathbf{r = r_1 - q \times r_2 \quad s = s_1 - q \times s_2 \quad t = t_1 - q \times t_2}$$

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	−5
1	28	21	7	0	1	−1	1	−5	6
3	21	7	0	1	−1	4	−5	6	−23
	7	0		**−1**	4		**6**	−23	

We get gcd (161, 28) = 7, $s = -1$ and $t = 6$.

The result can be tested, because (-1) × 161 + 6 × 28 = 7

Linear Diophantine Equation by EEA

- A linear Diophantine equation of two variables is like: ax + by = c.
  - Using extended Euclidean algorithm, we can find the solutions to the linear Diophantine equations, that is, we can find the integer values for x and y that satisfy the equation.
- Linear Diophantine equation has **either** no solution **or** has an infinite number of solutions:

Let d = gcd(a, b)

- If ***d*łc**, then the equation **ax + by = c** has no solution.
- If ***d*|*c***, then the equation has an infinite number of solutions.
  - Among the solutions, one is called **particular solution**, and the rest is called **general solutions**.

Linear Diophantine Equation by EEA

Particular solution:

Let $ax + by = c$ is a linear Diophantine equation. Also let, $d = \gcd(a,b)$.

- If $d|c$, then a particular solution of the equation can be found using the following steps:

1. Reduce the equation to $\mathbf{a_1x + b_1y = c_1}$ by dividing both sides of the equation by ***d***.
2. Solve for ***s*** and ***t*** in the relation $\boldsymbol{a_1s + b_1t = 1}$ using extended Euclidean algorithm.
3. The particular solution can be found by the following relations:

$$\mathbf{x_0 = (c/d)s \text{ and } y_0 = (c/d)t}$$

Linear Diophantine Equation by EEA

General Solution:

After finding the particular solution, the general solutions of linear Diophantine equation can be found by the following relations:

$$x = x_0 + k\,(b/d) \text{ and } y = y_0 - k(a/d)$$
where k is an integer

Linear Diophantine Equation by EEA

Example:

Using extended Euclidean algorithm, find the particular and general solutions to the following linear Diophantine equation:

$$21x + 14y = 35.$$

Solution:

Here, a = 21, b = 14 and c = 35.

d = gcd(a, b) = gcd(21,14) = 7. Since d | c or 7 | 35, the equation has an infinite number of solutions.

Reduce the equation by dividing both sides by 7 to find the equation: $3x + 2y = 5$. Here $a_1=3$, $b_1=2$ and $c_1=5$.

Using extended Euclidean algorithm, we find the value of s and t such that $3s + 2t = 1$. We have s = 1 and t = -1.

Particular solution: $x_0 = (c/d)s$ and $y_0 = (c/d)t$

$X_0 = (35/7) \times 1 = 5 \times 1 = 5$ and $y_0 = (35/7) \times (-1) = -5$

Example: (Cont…)

Solution:

General solution:

$x = x_0 + k\,(b/d)$ and $y = y_0 - k(a/d)$ where k is an integer

$X = 5 + k\,(14/7) = 5 + k \times 2$

$y = -5 - k\,(21/7) = -5 - k \times 3$

Therefore, considering the value of k as 0, 1, 2, 3, ….., the solutions to the above equation are (5, -5), (7, -8), (9, -11) …..

Congruence Relation

Consider the result of 2 mod 10 = 2, 12 mod 10 = 2, 22 mod 10 = 2, 32 mod 10 = 2 and so on.

- In modular arithmetic, integers like 2, 12, 22 and 32 are called congruent mod 10.

Two integers a and b are congruent modulo a positive integer m if and only if they have the same remainder r when divided by m, i.e., when a mod m = r and b mod m = r.

- More technically, if a and b are integers and m is a positive integer, then a is congruent to b modulo m if m divides a – b.
- The symbol ≡ is called the congruence operator.
- The notation a ≡ b (mod m) says that a is congruent to b modulo m.
- We say that a ≡ b (mod m) is a congruence and that m is its modulus.
- If a is not congruent to b modulo m, we write a≠b (mod m)

Example:

7 and 13 are congruent modulo 3, i.e., $7 \equiv 13 \pmod 3$.

24 and 14 are not congruent modulo 6, i.e., $24 \not\equiv 14 \pmod 6$.

$$2 \equiv 12 \pmod{10} \qquad 13 \equiv 23 \pmod{10}$$

$$3 \equiv 8 \pmod 5 \qquad 8 \equiv 13 \pmod 5$$

Application of Congruence

- Congruences have many applications in cryptography, e.g., shift ciphers
  - Shift cipher with key k encrypts message by shifting each letter by k letters in alphabet (if past Z, then wrap around)
  - What is encryption of ”KILL HIM” with shift cipher of key 3?

Modular Arithmetic

- Given any positive integer ***n*** and any nonnegative integer ***a***, if we divide ***a*** by ***n***, we get an integer quotient ***q*** and an integer remainder ***r*** such that $a = q \times n + r$.
- This division relation has two inputs (*a* and *n*) and two outputs (*q* and *r*).
- **In modular arithmetic, we are interested in only one of the outputs, the remainder *r*. In other words, we want to know what is the value of *r* when we divide *a* by *n*.**
- This implies that, using modular arithmetic, we can change the division relation into a binary operaton (called modulo operator) with two inputs *a* and *n* and one output r.
- The modulo operator is shown as mod. The second input (n) is called the **modulus**. The output r is called the **residue**.

Modular Arithmetic

Figure below shows the division relation compared with the modulo operator.

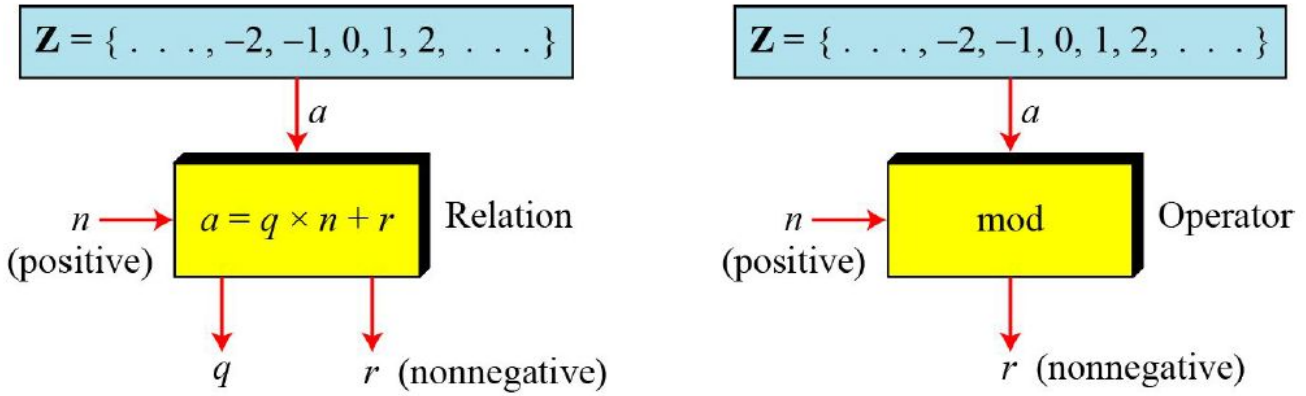


Figure: Division relation Vs. modulo operator

In the figure we see that the modulo operator (mod) takes an integer (a) from the set of integers (Z) and a positive modulus (n). The operator creates a non-negative residue (r) where 0 <= r <= n-1. We can say that:

a mod n = r

Modular Arithmetic

There are three cases for calculating modulus:

Case-1:

When both of a and n is positive integer where a<n:

- In this case, we add as many multiples of n with a as necessary to get a greater than n. Then divide a by n to get the remainder r. The result will be in the range 0 to n-1. For example, 2 mod 7 = 9 mod 7 = 2.

Case-2:

When both of a and n is positive integer with a>=n:

- In this case, just divide a by n to get the remainder r. The result will be in the range 0 to n-1. For example, 9 mod 7 = 2.

Case-3:

When a is negative and n is positive integer:

- In this case, we add as many multiples of n with a as necessary to get a positive and greater than n. Then divide a by n to get the remainder r. The result will be in the range 0 to n-1. The process is known as modulo reduction. For example, -12 mod 7 = -5 mod 7 = 2 mod 7 = 9 mod 7

Modular Arithmetic

Example:

Find the result of the following operations:

a. 27 mod 5
b. 36 mod 12
c. −18 mod 14
d. −7 mod 10

Solution

a. Dividing 27 by 5 results in r = 2. Therefore 27 mod 5 = 2

b. Dividing 36 by 12 results in r = 0. Therefore 36 mod 12 = 0

c. Dividing −18 by 14 results in r = −4. After adding the modulus (14) with the result to make it non-negative, we have r = -4 + 14 = 10. Therefore -18 mod 14 = 10 -4mod14=10mod14 =24mod14=10

d. Dividing −7 by 10 results in r = −7. After adding the modulus (10) with the result to make it non-negative, we have r = -7 + 10 = 3. Therefore −7 mod 10 = 3

Modular Arithmetic- Set of Residues

The result of ***a mod n*** is always an integer between *0* and *n-1*.

- Therefore, the modulo operation creates a set, which in modular arithmetic is referred to as the set of least residues modulo n, or Z_n.
- Figure below shows the set of residues Z_n and three instances of the set of residues Z_2, Z_6, Z_{11}.

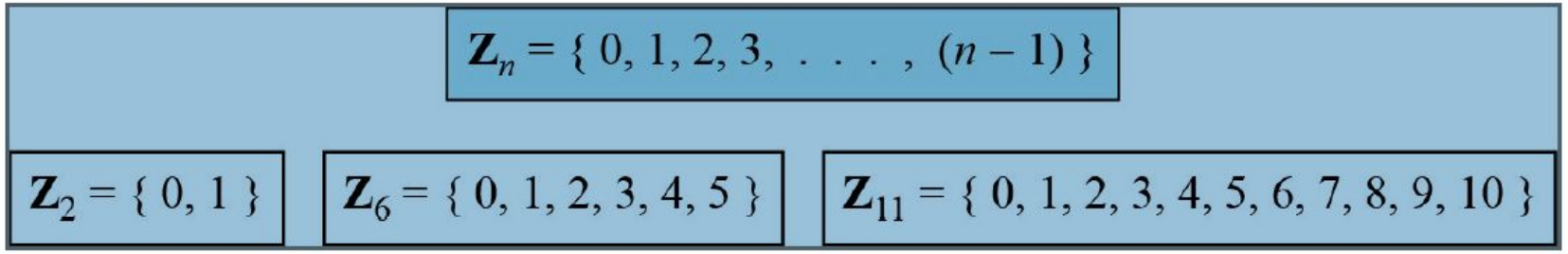


Figure: Some Z_n sets

Modular Arithmetic- Inverse Operation

In cryptography, we often need to find the inverse of a number relative to an operation. For example, if the sender uses an integer as the encryption key, the receiver uses the inverse of that integer as the decryption key.

- We are normally looking for two kinds of inverse:

1. **Additive Inverse**
   - If the operation is addition, then **additive inverse** is used.
   - The set of additive inverse is expressed as Z_n
2. **Multiplicative Inverse**
   - If the operation is multiplication, we are normally looking for multiplicative inverse.
   - The set of multiplicative inverse is expressed as Z_n^*.

Prepared by: K M Akkas Ali, Professor, IIT, JU

Modular Arithmetic- Additive Inverse

In Z_n, two numbers ***a*** and ***b*** are additive inverses of each other if

$$a + b \equiv 0 \pmod{n}$$

In modular arithmetic, each integer has an additive inverse.

- **The sum of an integer and its additive inverse is congruent to 0 modulo n.**

Example:

Find all additive inverse pairs in Z_{10}.

Solution:

The six pairs of additive inverses are

(0, 0), (1, 9), (2, 8), (3, 7), (4, 6), and (5, 5).

Modular Arithmetic- Multiplicative Inverse

In Z_n, two numbers **a** and **b** are the multiplicative inverse of each other if

$$a \times b \equiv 1 \pmod{n}$$

In modular arithmetic, an integer may or may not have a multiplicative inverse.

- **If an integer and the given modulus are co-prime (or if the GCD between the integer and modulus is 1), then multiplicative inverse of the integer exists in the given modulus.**

Example-1: Find the multiplicative inverse of 8 in Z_{10}.

Solution

There is no multiplicative inverse because gcd(10,8) = 2 ≠ 1.

- In other words, we cannot find any number between 0 and 9 such that when multiplied by 8, the result is congruent to 1.

Modular Arithmetic- Multiplicative Inverse

Example-2: Find all multiplicative inverses in Z_{10}.

Solution

There are only three pairs: (1, 1), (3, 7) and (9, 9).

- The integers 0, 2, 4, 5, 6, and 8 do not have a multiplicative inverse.

Example: Find all multiplicative inverse pairs in Z_{11}.

Solution

We have seven pairs: (1, 1), (2, 6), (3, 4), (5, 9), (7, 8), (9, 5), and (10, 10).

Prepared by: K M Akkas Ali, Professor, IIT, JU

Multiplicative Inverse Using EEA

The extended Euclidean algorithm finds the multiplicative inverses of b in Z_n when n and b are given and gcd (n, b) = 1.

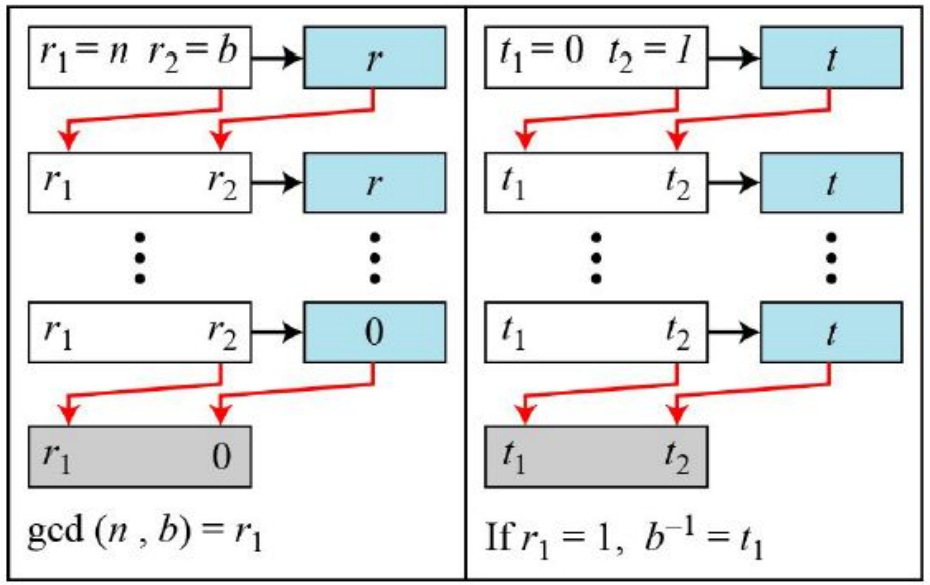
- **The multiplicative inverse of b is the value of t after being mapped to Zn.**

Prepared by: K M Akkas Ali, Professor, IIT, JU

Multiplicative Inverse Using EEA

Multiplicative inverse of an integer in a particular modulus can be determined using extended Euclidean algorithm.

- The process and algorithm are given below:



a. Process

$r_1 \leftarrow n; \quad r_2 \leftarrow b;$
$t_1 \leftarrow 0; \quad t_2 \leftarrow 1;$

while $(r_2 > 0)$
{
$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$
$r_1 \leftarrow r_2; \quad r_2 \leftarrow r;$

$t \leftarrow t_1 - q \times t_2;$
$t_1 \leftarrow t_2; \quad t_2 \leftarrow t;$

}
if $(r_1 = 1)$ then $b^{-1} \leftarrow t_1$

b. Algorithm

Prepared by: K M Akkas Ali, Professor, IIT, JU

Multiplicative Inverse Using EEA

Example:

Find the multiplicative inverse of 11 in Z_{26}.

Solution:

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	−2
2	11	4	3	1	−2	5
1	4	3	1	−2	5	−7
3	3	1	0	5	−7	26
	1	0		−7	26	

Since, gcd (26, 11) = 1; the multiplicative inverse of 11 in 26 modulus is -7 or 19 (Since, multiplicative inverse can not be negative).

Multiplicative Inverse Using EEA

Example:

Find the multiplicative inverse of 23 in Z_{100}.

Solution:

q	r_1	r_2	r	t_1	t_2	t
4	100	23	8	0	1	−4
2	23	8	7	1	−4	19
1	8	7	1	−4	9	−13
7	7	1	0	9	−13	100
	1	0		−13	100	

The gcd (100, 23) is 1; the multiplicative inverse of 23 in 100 modulus is -13 or 87.

Multiplicative Inverse Using EEA

Example:

Find the inverse of 12 in Z_{26}.

Solution

q	r_1	r_2	r	t_1	t_2	t
2	26	12	2	0	1	−2
6	12	2	0	1	−2	13
	2	0		−2	13	

The gcd (26, 12) is 2; the inverse does not exist.

Modular Inverse- Addition & Multiplication Tables

We can easily find out the additive and multiplicative inverse of an integer in $\mathbf{Z_n}$ using addition and multiplication table respectively.

- Figure below shows the addition and multiplication table for Z_{10}.

z = (x + y) mod 10

y \ x	0	1	2	3	4	5	6	7	8	9
0	**0**	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	**0**
2	2	3	4	5	6	7	8	9	**0**	1
3	3	4	5	6	7	8	9	**0**	1	2
4	4	5	6	7	8	9	**0**	1	2	3
5	5	6	7	8	9	**0**	1	2	3	4
6	6	7	8	9	**0**	1	2	3	4	5
7	7	8	9	**0**	1	2	3	4	5	6
8	8	9	**0**	1	2	3	4	5	6	7
9	9	**0**	1	2	3	4	5	6	7	8

Addition Table in $\mathbf{Z_{10}}$

z = (x * y) mod 10

y \ x	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	**1**	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	**1**	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	**1**	8	0	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	**1**

Multiplication Table in $\mathbf{Z_{10}}$

Figure: Addition and multiplication table for Z_{10}

Prepared by: K M Akkas Ali, Professor, IIT, JU

Modular Inverse- Addition & Multiplication Tables

- Figure below shows the multiplication table for Z_{13}.

x

z = (x * y) mod 13

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12
2	0	2	4	6	8	10	12	1	3	5	7	9	11
3	0	3	6	9	12	2	5	8	11	1	4	7	10
4	0	4	8	12	3	7	11	2	6	10	1	5	9
5	0	5	10	2	7	12	4	9	1	6	11	3	8
6	0	6	12	5	11	4	10	3	9	2	8	1	7
7	0	7	1	8	2	9	3	10	4	11	5	12	6
8	0	8	3	11	6	1	9	4	12	7	2	10	5
9	0	9	5	1	10	6	2	11	7	3	12	8	4
10	0	10	7	4	1	11	8	5	2	12	9	6	3
11	0	11	9	7	5	3	1	12	10	8	6	4	2
12	0	12	11	10	9	8	7	6	5	4	3	2	1

Figure: Addition and multiplication table for Z_{13}

Set of Additive and Multiplicative Inverse

Set of Additive Inverse Z_n

Z_n is a set that contains all integers from 0 to n-1.

- In Z_n , each integer has an additive inverse.
- Therefore Z_n can also be used as the set of additive inverse. Each member of Z_n has an additive inverse.

Set of Multiplicative Inverse Z_{n*}

In Z_n, an integer may or may not have a multiplicative inverse. Only some members of Z_n have a multiplicative inverse.

- Therefore, for multiplication operation, we need another set Z_{n*} which is a subset of Z_n that includes only those integers from Z_n that have a unique multiplicative inverse.

Prepared by: K M Akkas Ali, Professor, IIT, JU