

# Introduction to Number Theory

---

Assylbek Issakhov,  
Ph.D., professor

# Number Theory

---

- The part of mathematics devoted to the study of the set of integers and their properties is known as *Number Theory*. In this lecture we will develop some of the important concepts of Number Theory including many of those used in computer science.

# Division

---

- When one integer is divided by a second nonzero integer, the quotient may or may not be an integer. For example,

$$\frac{12}{3} = 4$$

- is an integer, whereas

$$\frac{11}{4} = 2.75$$

- is not. This leads to Definition 1.

# Division

---

- **DEFINITION 1.** If  $a$  and  $b$  are integers with  $a \neq 0$ , we say that  $a$  *divides*  $b$  if there is an integer  $c$  such that  $b = ac$ , or equivalently, if  $b/a$  is an integer. When  $a$  divides  $b$  we say that  $a$  is a *factor* or *divisor* of  $b$ , and that  $b$  is a *multiple* of  $a$ . The notation  $a|b$  denotes that  $a$  divides  $b$ .
- We write  $a \nmid b$  when  $a$  does not divide  $b$ .

## Examples

---

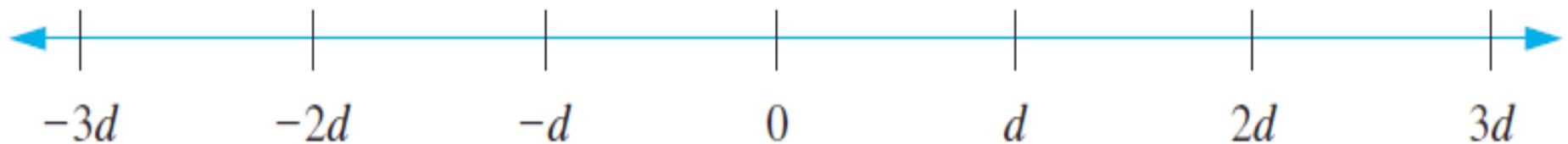
- Determine whether  $3|7$  and whether  $3|12$ .
- Let  $n$  and  $d$  be positive integers. How many positive integers not exceeding  $n$  are divisible by  $d$ ?

## Examples

---

- In Figure, a number line indicates which integers are divisible by the positive integer  $d$ .
- Answer:

$$\left[ \frac{n}{d} \right]$$



# Division

---

- **THEOREM 1.** Let  $a, b$ , and  $c$  be integers, where  $a \neq 0$ . Then
  - (i) if  $a|b$  and  $a|c$ , then  $a|(b + c)$ ;
  - (ii) if  $a|b$ , then  $a|bc$  for all integers  $c$ ;
  - (iii) if  $a|b$  and  $b|c$ , then  $a|c$ .

# Division

---

- **COROLLARY 1.** If  $a, b$ , and  $c$  are integers, where  $a \neq 0$ , such that

$$a|b \text{ and } a|c,$$

- then

$$a|(mb + nc)$$

- whenever  $m$  and  $n$  are integers.

# The Division Algorithm

---

- When an integer is divided by a positive integer, there is a quotient and a remainder, as the division algorithm shows.
- **THEOREM 2. (*THE DIVISION ALGORITHM*)** Let  $a$  be an integer and  $d$  a positive integer. Then there are unique integers  $q$  and  $r$ , with  $0 \leq r < d$ , such that

$$a = dq + r$$

# The Division Algorithm

---

- **DEFINITION 2.** In the equality given in the division algorithm,  $d$  is called the *divisor*,  $a$  is called the *dividend*,  $q$  is called the *quotient*, and  $r$  is called the *remainder*. This notation is used to express the quotient and remainder:

$$q = a \text{ } \mathbf{div} \text{ } d$$

$$r = a \text{ } \mathbf{mod} \text{ } d.$$

# The Division Algorithm

---

- **Remark:** Note that both  $a \text{ div } d$  and  $a \text{ mod } d$  for a fixed  $d$  are functions on the set of integers.
- Furthermore, when  $a$  is an integer and  $d$  is a positive integer, we have

$$q = a \text{ div } d = [a/d]$$

- and

$$r = a \text{ mod } d = a - dq$$

# Examples

---

- What are the quotient and remainder when 101 is divided by 11?
- What are the quotient and remainder when  $-11$  is divided by 3?

# The Division Algorithm

---

- **Remark:** A programming language may have one, or possibly two, operators for modular arithmetic, denoted by mod (in BASIC, Maple, Mathematica, EXCEL, and SQL), % (in C, C++, Java, and Python), rem (in Ada and Lisp), or something else. Be careful when using them, because for  $a < 0$ , some of these operators return  $a - m([a/m] + 1)$  instead of  $a \bmod m = a - m[a/m]$ . Also, unlike  $a \bmod m$ , some of these operators are defined when  $m < 0$ , and even when  $m = 0$ .

# Modular Arithmetic

---

- In some situations we care only about the remainder of an integer when it is divided by some specified positive integer.
- For instance, when we ask what time it will be (on a 24-hour clock) 50 hours from now, we care only about the remainder when 50 plus the current hour is divided by 24.

# Modular Arithmetic

---

- Because we are often interested only in remainders, we have special notations for them. We have already introduced the notation  $a \text{ mod } m$  to represent the remainder when an integer  $a$  is divided by the positive integer  $m$ . We now introduce a different, but related, notation that indicates that two integers have the same remainder when they are divided by the positive integer  $m$ .

# Modular Arithmetic

---

- **DEFINITION 3.** If  $a$  and  $b$  are integers and  $m$  is a positive integer, then  $a$  is *congruent to  $b$  modulo  $m$*  if  $m$  divides  $a - b$ . We use the notation

$$a \equiv b \pmod{m}$$

- to indicate that  $a$  is congruent to  $b$  modulo  $m$ . We say that  $a \equiv b \pmod{m}$  is a **congruence** and that  $m$  is its **modulus** (plural **moduli**). If  $a$  and  $b$  are not congruent modulo  $m$ , we write

$$a \not\equiv b \pmod{m}.$$

# Modular Arithmetic

---

- Although both notations  $a \equiv b \pmod{m}$  and  $a \text{ mod } m = b$  include “mod,” they represent fundamentally different concepts. The first represents a relation on the set of integers, whereas the second represents a function.
- However, the relation  $a \equiv b \pmod{m}$  and the **mod**  $m$  function are closely related, as described in the next Theorem 3.

# Modular Arithmetic

---

- **THEOREM 3.** Let  $a$  and  $b$  be integers, and let  $m$  be a positive integer. Then

$$a \equiv b \pmod{m}$$

- if and only if

$$a \bmod m = b \bmod m.$$

## Example

---

- Determine whether 17 is congruent to 5 modulo 6.
- Determine whether 24 and 14 are congruent modulo 6.

# Modular Arithmetic

---

- The great German mathematician Karl Friedrich Gauss developed the concept of congruences at the end of the eighteenth century.
- The notion of congruences has played an important role in the development of number theory. The next Theorem 4 provides a useful way to work with congruences.

# Modular Arithmetic

---

- **THEOREM 4.** Let  $m$  be a positive integer. The integers  $a$  and  $b$  are congruent modulo  $m$  if and only if there is an integer  $k$  such that

$$a = b + km.$$

# Modular Arithmetic

---

- The set of all integers congruent to an integer  $a$  modulo  $m$  is called the **congruence class** of  $a$  modulo  $m$ .
- **THEOREM 5.** Let  $m$  be a positive integer. If  $a \equiv b \pmod{m}$  and  $c \equiv d \pmod{m}$ , then

$$a + c \equiv b + d \pmod{m}$$

- and

$$ac \equiv bd \pmod{m}$$

## Example

---

- Because  $7 \equiv 2 \pmod{5}$  and  $11 \equiv 1 \pmod{5}$ , it follows from Theorem 5 that

$$18 = 7 + 11 \equiv 2 + 1 = 3 \pmod{5},$$

and that

$$77 = 7 \cdot 11 \equiv 2 \cdot 1 = 2 \pmod{5}.$$

# Arithmetic Modulo $m$

---

- We can define arithmetic operations on  $\mathbb{Z}_m$ , the set of nonnegative integers less than  $m$ , that is, the set  $\{0, 1, \dots, m - 1\}$ . In particular, we define addition of these integers, denoted by  $+_m$  by

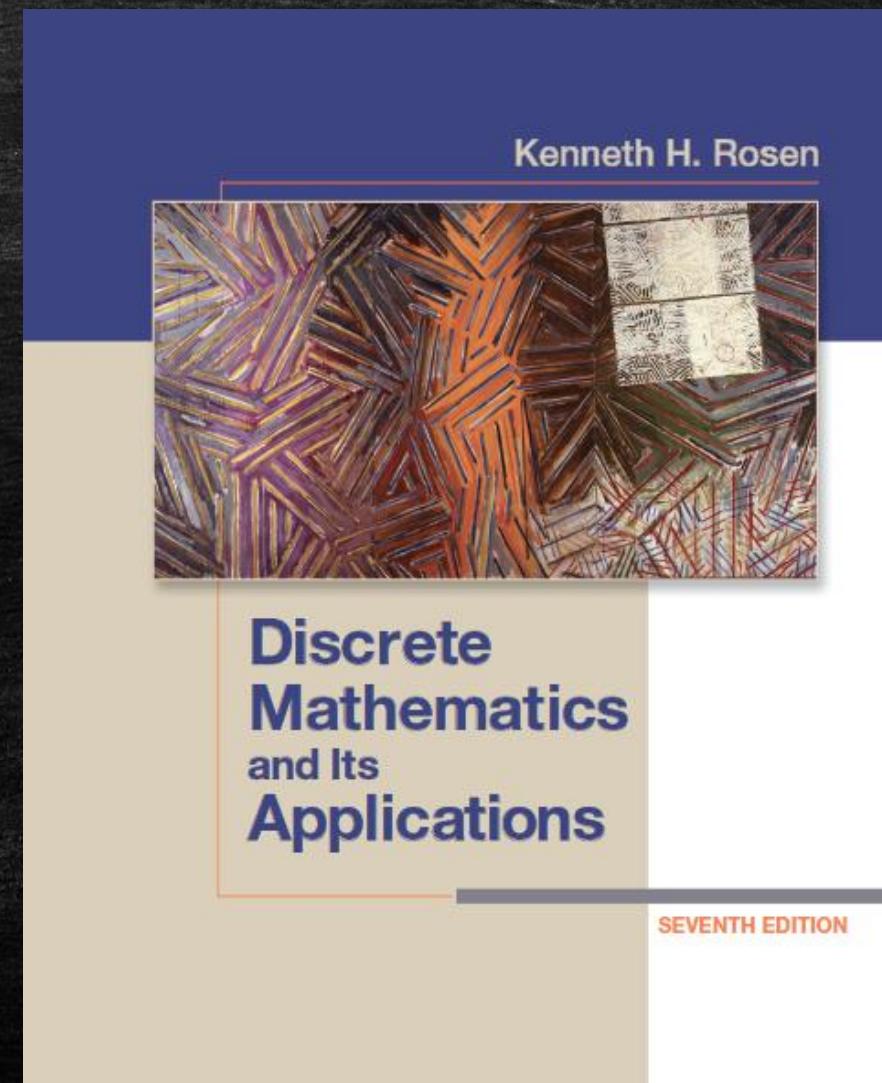
$$a +_m b = (a + b) \bmod m,$$

where the addition on the right-hand side of this equation is the ordinary addition of integers, and we define multiplication of these integers, denoted by  $\cdot_m$  by

$$a \cdot_m b = (a \cdot b) \bmod m$$

**HOMEWORK: Exercises 6, 10, 12, 14, 22, 24  
on pp. 244-245;**

---



# Representations of Integers

---

- In everyday life we use decimal notation to express integers. For example, 965 is used to denote

$$9 \cdot 10^2 + 6 \cdot 10 + 5.$$

However, it is often convenient to use bases other than 10.

# Representations of Integers

---

- In particular, computers usually use binary notation (with 2 as the base) when carrying out arithmetic, and octal (base 8) or hexadecimal (base 16) notation when expressing characters, such as letters or digits.
- In fact, we can use any integer greater than 1 as the base when expressing integers. This is stated in Theorem 1.

# Representations of Integers

---

- **THEOREM 1.** Let  $b$  be an integer greater than 1. Then if  $n$  is a positive integer, it can be expressed uniquely in the form

$$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0,$$

where  $k$  is a nonnegative integer,

$a_0, a_1, \dots, a_k$  are nonnegative integers less than  $b$ , and  $a_k \neq 0$ .

# Representations of Integers

---

- The representation of  $n$  given in Theorem 1 is called the **base  $b$  expansion of  $n$** . The base  $b$  expansion of  $n$  is denoted by

$$(a_k a_{k-1} \dots a_1 a_0)_b.$$

For instance,  $(245)_8$  represents  $2 \cdot 8^2 + 4 \cdot 8 + 5 = 165$ .

- Typically, the subscript 10 is omitted for base 10 expansions of integers because base 10, or **decimal expansions** are commonly used to represent integers.

# Representations of Integers

---

## BINARY EXPANSIONS

- Choosing 2 as the base gives **binary expansions** of integers. In binary notation each digit is either a 0 or a 1. In other words, the binary expansion of an integer is just a bit string.
- Binary expansions (and related expansions that are variants of binary expansions) are used by computers to represent and do arithmetic with integers.

## Example

---

- What is the decimal expansion of the integer that has

$$(1\ 0101\ 1111)_2$$

as its binary expansion?

## Example

---

- *Solution:*

$$\begin{aligned}(1\ 0101\ 1111)_2 \\&= 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 \\&\quad + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\&= 256 + 64 + 16 + 8 + 4 + 2 + 1 = 351\end{aligned}$$

# Representations of Integers

---

## OCTAL AND HEXADECIMAL EXPANSIONS

- Among the most important bases in computer science are base 2, base 8, and base 16.
- Base 8 expansions are called **octal** expansions and base 16 expansions are **hexadecimal** expansions.

## Examples

---

- What is the decimal expansion of the number with octal expansion  $(7016)_8$  ?
- *Solution:*

$$\begin{aligned}(7016)_8 &= 7 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 6 \cdot 8^0 \\ &= 7 \cdot 512 + 8 + 6 = 3598\end{aligned}$$

## Examples

---

- Sixteen different digits are required for hexadecimal expansions. Usually, the hexadecimal digits used are  
 $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F,$
- where the letters  $A$  through  $F$  represent the digits corresponding to the numbers 10 through 15 (in decimal notation).

## Examples

---

- What is the decimal expansion of the number with hexadecimal expansion  $(2AE0B)_{16}$ ?
- *Solution:*  
$$\begin{aligned}(2AE0B)_{16} &= 2 \cdot 16^4 + 10 \cdot 16^3 + 14 \cdot 16^2 + 0 \cdot 16^1 \\&\quad + 11 \cdot 16^0 \\&= 2 \cdot 65536 + 10 \cdot 4096 + 14 \cdot 256 + 0 \\&\quad + 11 = 131072 + 40960 + 3584 + 11 \\&= 175627\end{aligned}$$

# Representations of Integers

---

## BASE CONVERSION

- We will now describe an algorithm for constructing the base  $b$  expansion of an integer  $n$ . First, divide  $n$  by  $b$  to obtain a quotient and remainder, that is,

$$n = bq_0 + a_0, \quad 0 \leq a_0 < b.$$

- The remainder,  $a_0$ , is the rightmost digit in the base  $b$  expansion of  $n$ . Next, divide  $q_0$  by  $b$  to obtain

$$q_0 = bq_1 + a_1, \quad 0 \leq a_1 < b.$$

# Representations of Integers

---

## BASE CONVERSION

- We see that  $a_1$  is the second digit from the right in the base  $b$  expansion of  $n$ .
- Continue this process, successively dividing the quotients by  $b$ , obtaining additional base  $b$  digits as the remainders. This process terminates when we obtain a quotient equal to zero.
- It produces the base  $b$  digits of  $n$  from the right to the left.

## Example

---

- Find the binary expansion of  $(241)_{10}$ .
- *Solution:*

$$\begin{aligned}(241)_{10} &= 128 + 64 + 32 + 16 + 1 \\&= 2^7 + 2^6 + 2^5 + 2^4 + 2^0 \\&= (11110001)_2\end{aligned}$$

## Example

---

- Find the octal expansion of  $(12345)_{10}$ .
- *Solution:*

$$\begin{aligned}(12345)_{10} &= 8192 + 4096 + 32 + 16 + 8 + 1 \\ &= (11\ 000\ 000\ 111\ 001)_2 \\ &= (011\ 000\ 000\ 111\ 001)_2 \\ &= (30071)_8\end{aligned}$$

# Representations of Integers

TABLE 1 Hexadecimal, Octal, and Binary Representation of the Integers 0 through 15.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Octal	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
Binary	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

## ALGORITHM 1 Constructing Base $b$ Expansions.

**procedure**  $\text{base } b \text{ expansion}(n, b)$ : positive integers with  $b > 1$ )

$q := n$

$k := 0$

**while**  $q \neq 0$

$a_k := q \bmod b$

$q := q \text{ div } b$

$k := k + 1$

**return**  $(a_{k-1}, \dots, a_1, a_0)$   $\{(a_{k-1} \dots a_1 a_0)_b\}$  is the base  $b$  expansion of  $n$

# Representations of Integers

---

## Algorithms for Integer Operations

- The algorithms for performing operations with integers using their binary expansions are extremely important in computer arithmetic.
- We will describe algorithms for the addition and the multiplication of two integers expressed in binary notation

# Representations of Integers

---

- Throughout this discussion, suppose that the binary expansions of  $a$  and  $b$  are
$$a = (a_{n-1} \dots a_1 a_0)_2, b = (b_{n-1} \dots b_1 b_0)_2,$$
so that  $a$  and  $b$  each have  $n$  bits (putting bits equal to 0 at the beginning of one of these expansions if necessary).
- We will measure the complexity of algorithms for integer arithmetic in terms of the number of bits in these numbers.

# Representations of Integers

## ADDITION ALGORITHM

- Consider the problem of adding two integers in binary notation. To add  $a$  and  $b$ , first add their rightmost bits. This gives  $a_0 + b_0 = c_0 \cdot 2 + s_0$ , where  $s_0$  is the rightmost bit in the binary expansion of  $a + b$  and  $c_0$  is the **carry**, which is either 0 or 1.
- Then add the next pair of bits and the carry,  $a_1 + b_1 + c_0 = c_1 \cdot 2 + s_1$ , where  $s_1$  is the next bit (from the right) in the binary expansion of  $a + b$ , and  $c_1$  is the carry.

# Representations of Integers

---

## ADDITION ALGORITHM

- Continue this process, adding the corresponding bits in the two binary expansions and the carry, to determine the next bit from the right in the binary expansion of  $a + b$ .
- At the last stage, add  $a_{n-1}$ ,  $b_{n-1}$ , and  $c_{n-2}$  to obtain  $c_{n-1} \cdot 2 + s_{n-1}$ . The leading bit of the sum is  $s_n = c_{n-1}$ . This procedure produces the binary expansion of the sum, namely,

$$a + b = (s_n s_{n-1} \dots s_1 s_0)_2$$

## Example

---

- Add  $a = (1110)_2$  and  $b = (1011)_2$ .
- *Solution:*  
 $a + b = (1110)_2 + (1011)_2 = (11001)_2$
- *Chekking:*  
 $(1110)_2 = 8 + 4 + 2 = 14$   
 $(1011)_2 = 8 + 2 + 1 = 11$   
 $(11001)_2 = 16 + 8 + 1 = 25$

# Representations of Integers

---

## MULTIPLICATION ALGORITHM

- Next, consider the multiplication of two  $n$ -bit integers  $a$  and  $b$ . The conventional algorithm (used when multiplying with pencil and paper) works as follows. Using the distributive law, we see that

$$ab = a(b_0 2^0 + b_1 2^1 + \cdots + b_{n-1} 2^{n-1}) = a(b_0 2^0) + a(b_1 2^1) + \cdots + a(b_{n-1} 2^{n-1}).$$

- We can compute  $ab$  using this equation.

## Example

---

- Find the product of  $a = (110)_2$  and  $b = (101)_2$ .

- *Solution:*

$$\begin{aligned}a \cdot b &= (110)_2 \cdot (101)_2 \\&= (110)_2 + (0000)_2 + (11000)_2 \\&= (11110)_2\end{aligned}$$

- *Chekking:*

$$\begin{aligned}(110)_2 &= 6, & (101)_2 &= 5 \\(11110)_2 &= 16 + 8 + 4 + 2 = 30\end{aligned}$$

**HOMEWORK: Exercises 2, 4, 6, 8, 22, 32  
on pp. 255-256;**

