

# Discrete Mathematics

Sirasit Lochanachit, PhD

## Course Materials

- [https://github.com/noswolf/DM\\_DSBA\\_2020](https://github.com/noswolf/DM_DSBA_2020)
- Lecture Slides and Homework



## Grading (Final)

- Final 40%
- Homework 10%



## Course Outline

- |  |                  |
|--|------------------|
| 1) Number Theory                               | 02/10/2020       |
| 2) Number Theory and Cryptography              | 09/10/2020       |
| 3) Counting                                    | 16/10/2020       |
| 4) Relations                                   | 23/10/2020 (TBA) |
| 5) Graphs & Trees (covered in Data Structures) | 30/10/2020       |
| 6) Boolean Algebra                             | 06/11/2020       |
| 7) Modeling Computation                        | 13/11/2020       |
| 8) Finite State Automata                       | 20/11/2020 (TBA) |

# Number Theory

## ทฤษฎีจำนวน

### Number Theory (or Arithmetic)

- A branch of pure mathematics (independent of application)
- Dedicated to the study of integers and their properties.
- Key ideas: Division and Prime numbers.
- Number theory has important applications to computer science and cryptography.

### Chapter Summary

1. Divisibility and Modular Arithmetic
2. Integer Representations and Algorithms
3. Primes and Greatest Common Divisors
4. Applications of Congruences
5. Cryptography

### 1. Divisibility and Modular Arithmetic

$b / a$

- Division of an integer ( $b$ ) by a nonzero integer ( $a \neq 0$ )
  - Dividend or numerator ( $b$ ) = ตัวตั้ง
  - Divider, divisor, or denominator ( $a$ ) = ตัวหาร

## 1.1 Division

$b / a$

- Definition:

- If  $a$  and  $b$  are integers with  $a \neq 0$ ,  $a$  divides  $b$  (or  $b$  is divided by  $a$ ) if there is an integer  $c$  such that  $b = ac$ .
- $a | b$  denotes  $a$  divides  $b$ .
- If  $a | b$ , then  $b / a$  is an **integer**.
- $a \nmid b$  means  $a$  **does not** divide  $b$ .
  - There is no integer  $c$  such that  $b = ac$ .

## Division

$b / a \rightarrow a | b \rightarrow b = ac$

- Example:

- $4 | 8$  because  $8 / 4 = 2$ , which is an integer
  - In other words, there is  $c = 2$  such that  $8 = 4(2)$
- $-3 | 9$  because  $9 / (-3) = -3$
- $10 | 0$  because  $0 / 10 = 0$
- $5 \nmid 7$  because  $7 / 5 = 1.4$ , which is **not** an integer

## Some Properties of Division

- **Theorem 1:** Let  $a$ ,  $b$ , and  $c$  be integers, where  $a \neq 0$ . Then
  - if  $a | b$  and  $a | c$ , then  $a | (b + c)$ ;
- Direct Proof
  - Suppose that  $a | b$  and  $a | c$ .
  - From the definition of divisibility, it follows that there are integers  $s$  and  $t$  with  $b = as$  and  $c = at$ .
  - Thus,  $b + c = as + at = a(s + t)$
- Example:  $3 | 6$  and  $3 | 12$ , then  $3 | (6 + 12) = 3 | 18$

## Some Properties of Division

- **Theorem 1:** Let  $a$ ,  $b$ , and  $c$  be integers, where  $a \neq 0$ . Then
  - if  $a | b$  and  $a | c$ , then  $a | (b + c)$ ;
  - if  $a | b$  and  $a | c$ , then  $a | mb + nc$  where  $m$  and  $n$  are integers;
  - if  $a | b$  then  $a | bc$  for all integers  $c$ ;
  - if  $a | b$  and  $b | c$  then  $a | c$ ;
  - if  $a | b$  then  $|a| \leq |b|$

## 1.2 Division Algorithm/Theorem

- Division of an integer ( $a$ ) by a positive integer, there is a quotient and a remainder.

$$a / d = q \text{ remainder } r$$

- Theorem 2:** 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  $a = dq + r$ .

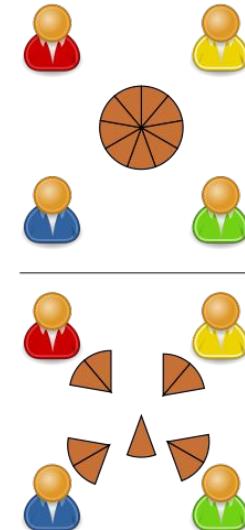
- Also known as **Euclidean Division**.

## Division Algorithm/Theorem

$$a = dq + r$$
$$a / d = q \text{ remainder } r$$

- Definition

- Dividend or numerator ( $a$ ) = ตัวตั้ง
- Divisor, divisor, or denominator ( $d$ ) = ตัวหาร
- Quotient ( $q$ ) = ผลหาร
  - $q = a \text{ div } d$
- Remainder ( $r$ ) = เศษเหลือ
  - $r = a \text{ mod } d$



## Division Algorithm/Theorem

$$a = dq + r$$

$$a / d = q \text{ remainder } r$$

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

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

- Example: Find the quotient and remainder.

- $101 / 11 \rightarrow 101 = (11 * 9) + 2$

- $\quad \text{■ Quotient (q)} \rightarrow 9 = 101 \text{ div } 11$

- $\quad \text{■ Remainder (r)} \rightarrow 2 = 101 \text{ mod } 11$

## Division Algorithm/Theorem

$$a = dq + r$$

$$a / d = q \text{ remainder } r$$

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

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

- Example: Find the quotient and remainder.

- $-13 / 4 \rightarrow -13 = 4(-4) + 3$

- $\quad \text{■ Quotient (q)} \rightarrow -4 = -13 \text{ div } 4$

- $\quad \text{■ Remainder (r)} \rightarrow 3 = -13 \text{ mod } 4$

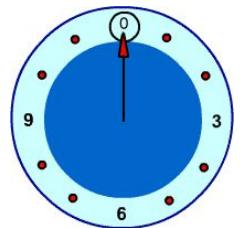
- $-13 / 4 \rightarrow -13 = 4(-3) - 1 ?$

## 1.3 Modular Arithmetic

- In certain circumstances, we are only interested about the remainder of an integer when divided by some positive integer.
  - For these cases, there is an operator called the **modulo** operator.
  - $a \bmod d = r \rightarrow a \bmod d$  is equal to  $r$ 
    - $d$  is known as a modulus.
  - Example:  $20 \bmod 3 = 2$

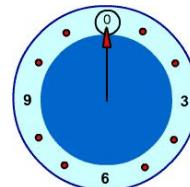
## Modular Arithmetic

- Real-life example:
  - 12-hour clock
  - Circular Queue (Data Structure)

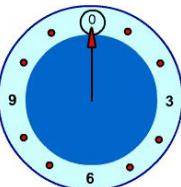
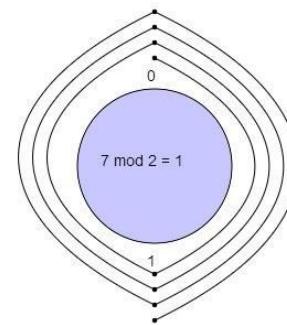
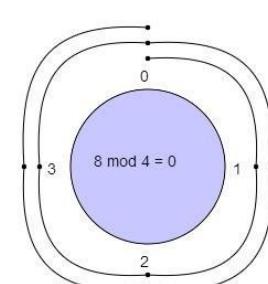


## Modular Arithmetic

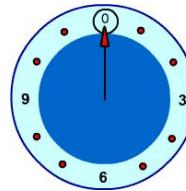
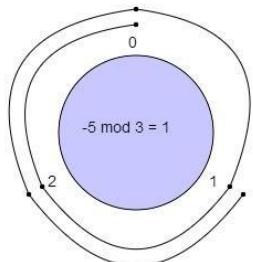
- Visualising modulo operator using circles.
  1. Write 0 at the top of a circle and continuing clockwise writing integers 1, 2, ... up to one less than the modulus ( $d$ ).
  2. Start at 0 and move around the clock  $a$  steps
    - If  $a$  is positive, step clockwise
    - If  $a$  is negative, step counter-clockwise.



## Modular Arithmetic

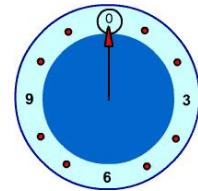


## Modular Arithmetic



## Modular Arithmetic Exercise

1.  $-17 \text{ mod } 7$
2.  $-100 \text{ mod } 8$
3.  $((29 \text{ mod } 17) \text{ mod } 7) \text{ mod } 3$
4.  $(109 \text{ mod } 12) + (-109 \text{ mod } 12)$
5.  $(((-29 \text{ mod } 17) \text{ mod } 7) \text{ mod } 3)$



Alternatively, the formula for modulo operation when  $a$  is negative is:

$$a \text{ mod } d = a - (\text{floor}(a/d) * d)$$

## Congruence Modulo

- Definition: Two integers ( $a$  and  $b$ ) have the same remainder when they are divided by the positive integer  $c$ .
  - $a$  is congruent (สมภาค) to  $b$  modulo  $c$  if  $c$  divides  $a - b$  ( $c \mid a - b$ ).
  - $a \equiv b \pmod{c}$ .
- **Theorem 3:** Let  $a$  and  $b$  be integers, and  $c$  be a positive integer.
  - Then  $a \equiv b \pmod{c}$  if and only if  $a \text{ mod } c = b \text{ mod } c$ .

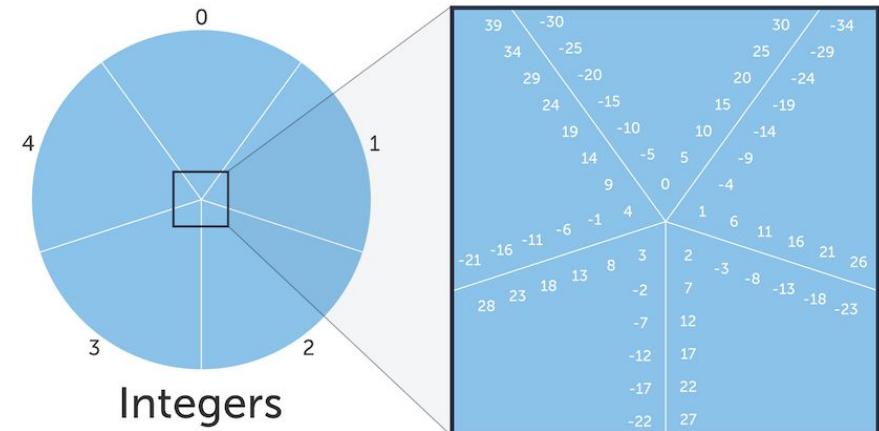
## Congruence Modulo

- Example:  $20 \equiv 8 \pmod{12}$ 
  - $8 \text{ mod } 12 = 8$  and  $20 \text{ mod } 12 = 8$
- The use of “mod” in  $a \equiv b \pmod{c}$  and  $a \text{ mod } c = b$  are different.
  - $a \equiv b \pmod{c}$  is a relation on the set of integers (congruent mod).
  - In  $a \text{ mod } c = b$ , the notation **mod** denotes a function.

## Congruence Modulo

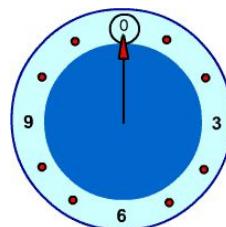
- **Theorem 4:** Let  $c$  be a positive integer. The integers  $a$  and  $b$  are congruent modulo  $c$  if and only if there is an integer  $k$  such that  $a = b + kc$ .
- Proof:
  - If  $a \equiv b \pmod{c}$ , then (by the definition of congruence)  $c \mid a - b$ .
  - Thus, there is an integer  $k$  such that  $a - b = kc$ , so that  $a = b + kc$ .

## Congruence Modulo



## Congruence Modulo Example

- 12-hour clock
  - Currently it is 8:00am
  - 6 Hours later is  $8 + 6 = 14$  (pm)
  - $(8+6) \bmod 12$ , means 14:00 and 2:00pm
  - 14 is **congruent** to 2 modulo 12
    - $14 \equiv 2 \pmod{12}$



## Congruence Properties

- Equivalent statements
  - $a \equiv b \pmod{c}$
  - $a \bmod c = b \bmod c$
  - $c \mid (a - b)$
  - $a = b + kc$
- Example:  $11 \equiv 21 \pmod{5}$ 
  - $11 \equiv 21 \pmod{5}$
  - $11 \bmod 5 = 21 \bmod 5$
  - $5 \mid (11 - 21)$
  - $11 = 21 + 5k$ 
    - $k = -2$

## Congruence Modulo Exercise

- Suppose that  $a$  and  $b$  are integers,
  - $a \equiv 4 \pmod{13}$
  - $b \equiv 9 \pmod{13}$
  - Find the integer  $c$  with  $0 \leq c \leq 12$  such that
    - $c \equiv 9a \pmod{13}$
    - $c \equiv 11b \pmod{13}$
    - $c \equiv a + b \pmod{13}$

## Congruence Modulo is an Equivalence Relation

- Every pair of values in a slice are related to each other.
- No value is in more than one slice.
- A pie contains all slices, which have **all** of the **integer** values.
- A pie has **equivalence relation**, which defines how to partition set of values (**mod**) into **congruence** classes (slices).

## Equivalence Relation Properties

If congruence modulo  $c$  is an equivalence relation for  $(\text{mod } c)$ .

1. **Reflexive:**  $a$  is related to  $a$ .
  - $a \equiv a \pmod{c}$
2. **Symmetric:** if  $a$  is related to  $b$ , then  $b$  is related to  $a$ .
  - if  $a \equiv b \pmod{c}$  then  $b \equiv a \pmod{c}$
3. **Transitive:** if  $a$  is related to  $b$  and  $b$  is related to  $c$ , then  $a$  is related to  $c$ .
  - if  $a \equiv b \pmod{c}$  and  $b \equiv d \pmod{c}$  then  $a \equiv d \pmod{c}$

## Equivalence Relation Properties (Example)

If congruence modulo  $c$  is an equivalence relation for  $(\text{mod } c)$ .

1. **Reflexive:**  $a$  is related to  $a$ .
  - $2 \equiv 2 \pmod{5}$
2. **Symmetric:** if  $a$  is related to  $b$ , then  $b$  is related to  $a$ .
  - if  $2 \equiv 7 \pmod{5}$  then  $7 \equiv 2 \pmod{5}$
3. **Transitive:** if  $a$  is related to  $b$  and  $b$  is related to  $c$ , then  $a$  is related to  $c$ .
  - if  $2 \equiv 7 \pmod{5}$  and  $7 \equiv 17 \pmod{5}$  then  $2 \equiv 17 \pmod{5}$

## Congruence Properties (Sums and Products)

- **Theorem 5:** Let  $c$  be a positive integer.
- If  $a \equiv b \pmod{c}$  and  $d \equiv e \pmod{c}$ , then
  - $a + d \equiv b + e \pmod{c}$  and
  - $ad \equiv be \pmod{c}$
- Example:  $16 \equiv 2 \pmod{7}$  and  $8 \equiv 1 \pmod{7}$ 
  - $16 + 8 \equiv (2 + 1) \pmod{7} \rightarrow 24 \equiv 3 \pmod{7}$
  - $16 * 8 \equiv (2 * 1) \pmod{7} \rightarrow 128 \equiv 2 \pmod{7}$

## Modular Operations

- Addition
  - $(a + b) \pmod{c} = ((a \pmod{c}) + (b \pmod{c})) \pmod{c}$
- Multiplication
  - $ab \pmod{c} = ((a \pmod{c})(b \pmod{c})) \pmod{c}$

## Modular Operations Exercise

- $(14 + 17) \pmod{5}$  and  $((14 \pmod{5}) + (17 \pmod{5})) \pmod{5}$
- $((-7) + (-10)) \pmod{7}$
- $71 * 122 \pmod{11}$
- $(46 * 71 * 122) \pmod{11}$
- $(-45 * 77) \pmod{17}$

## Modular Operations

- Exponentiation
  - $a^b \pmod{c} = (a \pmod{c})^b \pmod{c}$
  - b can get very large, making  $a^b$  massively large and taking a long time to calculate the mod.
- For quicker calculation, divide and conquer can be used.
- Examples:  $2^{90} \pmod{13}$ 
  - $(2^{50} * 2^{40}) \pmod{13}$
  - $(2^{50} \pmod{13} * 2^{40} \pmod{13}) \pmod{13} \rightarrow (4 * 3) \pmod{13}$

## 2. Integer Representations

- In the modern world, we use **decimal**, or **base 10**, notation to represent integers. For example when we write 965, we mean  $9 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0$ .
- We can represent numbers using any base  $b$ , where  $b$  is a positive integer greater than 1.
- The bases  $b = 2$  (binary),  $b = 8$  (octal) , and  $b = 16$  (hexadecimal) are important for computing and communications.
- The ancient Mayans (2000 BC) used base 20.
- The ancient Babylonians (1895 BC – 539 BC) used base 60.

## Base $b$ Representations

- We can use positive integer  $b$  greater than 1 as a base
- **Theorem 1:** Let  $b$  be a positive integer greater than 1. Then if  $n$  is a positive integer, it can be expressed uniquely in the form:

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

where  $k$  is a nonnegative integer,  $a_0, a_1, \dots, a_k$  are nonnegative integers less than  $b$ , and  $a_k \neq 0$ . The  $a_j, j = 0, \dots, k$  are called the base- $b$  digits of the representation.

- The representation of  $n$  given in Theorem 1 is called the *base  $b$  expansion of  $n$*  and is denoted by  $(a_k a_{k-1} \dots a_1 a_0)_b$ .

## Binary Expansions (base 2)

- Most computers represent integers and do arithmetic with binary (base 2) expansions of integers. In these expansions, the only digits used are 0 and 1.
- Example 1: What is the decimal expansion of the integer that has  $(10101111)_2$  as its binary expansion?
- Example 2: What is the decimal expansion of the integer that has  $(11011)_2$  as its binary expansion?

## Octal Expansions (base 8)

- The octal expansion (base 8) uses the digits {0, 1, 2, 3, 4, 5, 6, 7}.
- Example 1: What is the decimal expansion of the number with octal expansion  $(7016)_8$ ?
- Example 2: What is the decimal expansion of the number with octal expansion  $(111)_8$ ?

## Hexadecimal Expansions (base 16)

- The hexadecimal expansion needs 16 digits, but our decimal system provides only 10. So letters are used for the additional symbols.
- The hexadecimal system uses the digits {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}. The letters A through F represent the decimal numbers 10 through 15.
- Example 1: What is the decimal expansion of the number with hexadecimal expansion  $(2AE0B)_{16}$ ?
- Example 2: What is the decimal expansion of the number with hexadecimal expansion  $(E5)_{16}$ ?

## Base Conversion Algorithm

```
Algorithm base_b_expansion(n, b: positive integers with b > 1)
```

```
q = n, k = 0
```

```
while (q ≠ 0):
```

```
    ak = q mod b
```

```
    q = q div b
```

```
    k = k + 1
```

```
return(ak-1, ..., a1, a0) where {(ak-1 ... a1a0)b is base b expansion of n}
```

- $q$  represents the quotient obtained by successive divisions by  $b$ , starting with  $q = n$ .
- The digits in the base  $b$  expansion are the remainders of the division given by  $q \text{ mod } b$ .

## Base Conversion Algorithm

To construct the base  $b$  expansion of an integer  $n$ :

- Divide  $n$  by  $b$  to obtain a quotient and remainder.
$$n = bq_0 + a_0 \quad 0 \leq a_0 \leq b$$
- The remainder,  $a_0$ , is the rightmost digit in the base  $b$  expansion of  $n$ .
- Next, divide  $q_0$  by  $b$ .
$$q_0 = bq_1 + a_1 \quad 0 \leq a_1 \leq b$$
- The remainder,  $a_1$ , is the second digit from the right in the base  $b$  expansion of  $n$ .
- Continue by successively dividing the quotients by  $b$ , obtaining the additional base  $b$  digits as the remainder. The process terminates when the quotient is 0.

## Base Conversion Example 1

**Example:** Find the octal expansion of  $(12345)_{10}$

**Solution:** Successively dividing by 8 gives:

$$12345 = 8 * 1543 + 1$$

$$1543 = 8 * 192 + 7$$

$$192 = 8 * 24 + 0$$

$$24 = 8 * 3 + 0$$

$$3 = 8 * 0 + 3$$

The remainders are the digits from right to left yielding  $(30071)_8$ .

## Base Conversion Example 2

**Example:** Find the hexadecimal expansion of  $(177130)_{10}$

## Base Conversion Example 3

**Example:** Find the binary expansion of  $(241)_{10}$

## Base Conversion

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

Each octal digit corresponds to a block of 3 binary digits.

Each hexadecimal digit corresponds to a block of 4 binary digits.

So, conversion between binary, octal, and hexadecimal is easy.

## Base Conversion Example 4

**Example 1:** Find the octal and hexadecimal expansions of  $(11\ 1110\ 1011\ 1100)_2$ .

**Example 2:** Find the binary expansions of  $(765)_8$  and  $(A8D)_{16}$ .

## 2.1 Integer Operations Algorithms

- Algorithms for performing operations with integers using their binary expansions are extremely important as computer chips work with binary numbers. Each digit is called a bit.
- Suppose the binary expressions of  $a$  and  $b$  are

$$a = (a_{n-1}, a_{n-2}, \dots, a_0)_2 \text{ and } b = (b_{n-1}, b_{n-2}, \dots, b_0)_2$$

where  $a$  and  $b$  each have  $n$  bits.

## Binary Addition Algorithm

To add two integers  $a$  and  $b$  in binary notation:

- First, add their rightmost bits.
$$a_0 + b_0 = c_0 * 2 + s_0$$
- The rightmost bit,  $s_0$  is the result of  $a + b$  and  $c_0$  is the **carry**.
- Next, add the next pair of bits and the carry.
$$a_1 + b_1 + c_0 = c_1 * 2 + s_1$$
- $s_1$  is next bit from the right.
- Continue by successively adding the corresponding bits and the carry. The leading bit of the sum is  $s_n = c_{n-1}$ .

## Binary Addition Algorithm Example

**Example:** Add  $a = (1110)_2$  and  $b = (1011)_2$ .

$$\begin{array}{r} 1 \ 1 \ 1 \\ 1 \ 1 \ 1 \ 0 \\ + 1 \ 0 \ 1 \ 1 \\ \hline 1 \ 1 \ 0 \ 0 \ 1 \end{array}$$

## Binary Addition Algorithm

```
Algorithm binary_add(a, b: positive integers)
    # where a = (a_{n-1}, a_{n-2}, ..., a_0)_2 and b = (b_{n-1}, b_{n-2}, ..., b_0)_2
    c = 0          # carry
    for j in range(0, n):
        d = L(a_j + b_j + c)/2
        s_j = a_j + b_j + c - 2d
        c = d
    s_n = c
    return(s_0, s_1, ..., s_n) where {the binary expansion of the sum is (s_n, s_{n-1}, ..., s_0)_2}
```

## Binary Multiplication Algorithm

To multiply two integers  $a$  and  $b$  in binary notation, distributive law can be used:

$$\begin{aligned} ab &= a(b_0 2^0 + b_1 2^1 + \dots + b_{n-1} 2^{n-1}) \\ &= a(b_0 2^0) + a(b_1 2^1) + \dots + a(b_{n-1} 2^{n-1}) \end{aligned}$$

- Note that  $ab_j = a$  if  $b_j = 1$  and  $ab_j = 0$  if  $b_j = 0$ .
- Each time when multiply, the binary expansion is shifted one place to the left and add a zero at the tail end of the expression.

## Binary Multiplication Algorithm Example

**Example:** Multiply  $a = (110)_2$  and  $b = (101)_2$ .

$$\begin{array}{r} 110 \\ \times 101 \\ \hline 110 \\ 000 \\ 110 \\ \hline 11110 \end{array}$$

## Binary Multiplication Algorithm

```
Algorithm binary_multiply(a, b: positive integers)
    # where a = (a_{n-1}, a_{n-2}, ..., a_0)_2 and b = (b_{n-1}, b_{n-2}, ..., b_0)_2
for j in range(0, n):
    if b_j = 1 then c_j = a shifted j places to the left and add 0s at the tail
    else c_j = 0
{c_0, c_1, ..., c_{n-1} are the partial products = ตัวผล}
p = 0
for j in range (0, n):      # Add
    p = p + c_j
return p where {p is the value of ab}
```

## Integer Operations Algorithms

- The number of additions of bits used by the algorithm to add two  $n$ -bit integers is  $O(n)$ .
- The number of additions of bits used by the algorithm to multiply two  $n$ -bit integers is  $O(n^2)$ .

## 2.2 Modular Operations

- Exponentiation
  - $a^b \text{ mod } c = (a \text{ mod } c)^b \text{ mod } c$
  - b can get very large, making  $a^b$  massively large and taking a long time to calculate the mod.
- For quicker calculation, divide and conquer can be used.
- Examples:  $2^{90} \text{ mod } 13$ 
  - $(2^{50} * 2^{40}) \text{ mod } 13$
  - $(2^{50} \text{ mod } 13 * 2^{40} \text{ mod } 13) \text{ mod } 13 \rightarrow (4 * 3) \text{ mod } 13$

## Modular Operations

- Method #1: **Divide and conquer**
  - $a^b \text{ mod } c = (a \text{ mod } c)^b \text{ mod } c$
- If  $b$  is an even number,
  - $a^b \text{ mod } c = (a^2)^{b/2} \text{ mod } c$   
 $= (a^2 \text{ mod } c)^{b/2} \text{ mod } c$
- If  $b$  is an odd number,
  - $a^b \text{ mod } c = (aa^{b-1}) \text{ mod } c$   
 $= (a * (a^2)^{(b-1)/2}) \text{ mod } c$   
 $= (a * a^2 \text{ mod } c)^{(b-1)/2} \text{ mod } c$

## Modular Exponentiation Example

**Example:** Compute  $5^{50} \text{ mod } 19$  using divide and conquer strategy.

## Binary Modular Exponentiation

- In cryptography, it is important to be able to find  $b^n \text{ mod } c$  efficiently without using an excessive amount of memory, where  $b$ ,  $n$ , and  $c$  are large integers.
- Method # 2: Use the **binary** expansion of  $n$ ,  $n = (a_{k-1}, \dots, a_1, a_0)_2$ , to compute  $b^n$ .

$$b^n = b^{a_{k-1} \cdot 2^{k-1} + \dots + a_1 \cdot 2 + a_0} = b^{a_{k-1} \cdot 2^{k-1}} \dots b^{a_1 \cdot 2} \cdot b^{a_0}$$

- Therefore, to compute  $b^n$ , we need only compute the values of  $b$ ,  $b^2$ ,  $(b^2)^2 = b^4$ ,  $(b^4)^2 = b^8$ , ...,  $b^{2^k}$  and the multiply the terms  $b^{2^j}$  in this list, where  $a_j = 1$ .

## Binary Modular Exponentiation Example

**Example:** Compute  $3^{11}$  using binary modular exponentiation.

**Solution:**  $11 = (1011)_2$

$$\begin{aligned} \text{so that } 3^{11} &= 3^8 3^2 3^1 = ((3^2)^2)^2 3^2 3^1 \\ &= (9^2)^2 9^3 = (81)^2 9^3 \\ &= 6561 * 9^3 \\ &= 117,147. \end{aligned}$$

## Fast Modular Exponentiation Algorithm

- The algorithm successively finds  $b \bmod c, b^2 \bmod c, b^4 \bmod c, \dots, b^{2^{k-1}} \bmod c$ , and multiplies together the terms  $b^{2^j} \bmod c$  where  $a_j = 1$ .

```
Algorithm modular_expo(b: integer, n = (ak-1ak-2...a1a0)2, c: positive integers)
x = 1
power = b mod c
for i in range(0, k):
    if ai = 1 then x = (x*power) mod c
    power = (power*power) mod c
return x where {x equals bn mod c }
```

- $O((\log c)^2 \log n)$  bit operations are used to find  $b^n \bmod c$ .

## Fast Modular Exponentiation Example

**Example:** Compute  $3^{644} \bmod 645$  using fast (binary) modular exponentiation.

## Fast Modular Exponentiation Exercise

**Example:** Compute  $5^{600} \bmod 13$  using fast (binary) modular exponentiation.