Number Theory and Algorithms

Dr. Isaac Griffith Idaho State University

Inspiration

"The enemy knows the system" – Claude Shannon

Outline

The lecture is structured as follows:

- Divisibility and Modular Arithmetic
- Integer Representations
- Integer Algorithms
- Primes and GCD
- Solving Congruences
- Applications of Congruences
- Cryptography
- Program Correctness

Divisibility & Modular Arithmetic

CS 1187

Division

- When an integer is divided by another integer, the result may or may not be an integer.
  - Ex: $12/3 = 4$, $11/4 = 2.75$
- **Definition:** If a and b are integers with $a \neq 0$, we say a *divides* b if there is an integer c such that $b = ac$ (if $\frac{b}{a}$ is an integer).
  - When a divides b (written $a \mid b$) we say a is a *factor* or *divisor* of b and that b is a *multiple* of a
  - We an express $a \mid b$ logically as $\exists c\ (ac = b)$
- *Example:* Determine whether $3 \mid 7$ and whether $3 \mid 12$

$$3 \nmid 7 \text{ because } 7/3 \notin \mathbb{Z}$$
$$3 \mid 12 \ \ textbecause\ 12/3 = 4$$

ROAR

Division

- **Theorem:** Let a, b, and c be integers, where $a \neq c$. Then
  1. if $a \mid b$ and $a \mid c$, then $a \mid (b + c)$
  2. if $a \mid b$, then $a \mid bc$ for all integers c
  3. if $a \mid b$ and $b \mid c$, then $a \mid c$
- **Corollary:** if a, b, and c are integers, where $a \neq 0$, such that $a \mid b$ and $a \mid c$, then $a \mid mb + nc$ whenever m and n are integers.

The Division Algorithm

- **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 $a = dq + r$
  - d is the *divisor*
  - a is the *dividend*
  - q is the *quotient*
  - r is the *remainder*

$$q = a \textbf{ div } d \qquad r = a \textbf{ mod } d$$

- *Example:* What is the quotient and remainder when 101 is divided by 11?

$$\begin{aligned} 101 &= 11 \cdot 9 + 2 \\ q &= 101 \textbf{ div } 11 = 9 \\ r &= 101 \textbf{ mod } 11 = 2 \end{aligned}$$

Modular Arithmetic

- **Definition:** if a and b are two integers and m is a positive integer, then a is *congruent to b modulo m* if m divides $a - b$.
  - Denoted as: $a \equiv b(\textbf{mod}\, m) \Rightarrow$ called a **congruence**
  - m is its **modulus**
  - if a and b are not congruent modulo m, we write $a \not\equiv b\ (\textbf{mod}\, m)$
- **Theorem:** Let a and b be integers, and let m be a positive integer. Then $a \equiv b\ (\textbf{mod}\, m)$ iff $a\ \textbf{mod}\, m = b\ \textbf{mod}\, m$
- *Example:* Determine whether $17 \equiv 5\ (\textbf{mod}\, 6)$ and $24 \equiv 14\ (\textbf{mod}\, 6)$

$$\begin{array}{rcl} 6 \mid (17 - 5 = 12) & \rightarrow & 17 \equiv 5\ (\textbf{mod}\, 6) \\ 6 \nmid (24 - 14 = 10) & \rightarrow & 24 \not\equiv 14\ (\textbf{mod}\, 6) \end{array}$$

Modular Arithmetic

- **Theorem:** Let m be a positive integer. The integers a and b are congruent modulo m iff there is an integer k such that $a = b + km$
- **Theorem:** Let m be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then

$$a + c \equiv d \pmod{m} \quad \text{and} \quad ac = bd \pmod{m}$$

- *Example*: because $7 \equiv 2 \pmod{5}$ and $11 \equiv 1 \pmod{5}$ it follows that

$$\begin{aligned} 18 = 7 + 11 \quad &\equiv \quad 2 + 1 = 3 \pmod{5} \\ 77 = 7 \cdot 11 \quad &\equiv \quad 2 \cdot 1 = 2 \pmod{5} \end{aligned}$$

Arithmetic Modulo m

- $\mathbb{Z}_m$: set of non-negative integers less than m
- **Arithmetic Modulo m** operators:

$$a +_m b = (a + b) \textbf{ mod } m \qquad a \cdot_m b = (a \cdot b) \textbf{ mod } m$$

- Examples: find $7 +_{11} 9$ and $7 \cdot_{11} 9$

$$\begin{aligned} 7 +_{11} 9 &= (7 + 9) \textbf{ mod } 11 &= 16 \textbf{ mod } 11 &= 5 \\ 7 \cdot_{11} 9 &= (7 \cdot 9) \textbf{ mod } 11 &= 63 \textbf{ mod } 11 &= 8 \end{aligned}$$

Arithmetic Modulo m

- The operators $+_m$ and $\cdot_m$ satisfy the following properties
  - **Closure:** if a and $b \in \mathbb{Z}_m$, then $a +_m b$ and $a \cdot_m b \in \mathbb{Z}_m$
  - **Associativity:** if $a, b, c \in \mathbb{Z}_m$, then $(a +_m b) +_m c = a +_m (b +_m c)$ and $(a \cdot_m b) \cdot_m c = a \cdot_m (b \cdot_m c)$
  - **Commutativity:** if $a, b \in \mathbb{Z}_m$, then $a +_m b = b +_m a$ and $a \cdot_m b = b \cdot_m a$
  - **Identity:** The elements 0 and 1 are identity elements for $+_m$ and $\cdot_m$, respectively. If $a \in \mathbb{Z}_m$, then $a +_m 0 = 0 +_m a = a$, and $a \cdot_m 1 = 1 \cdot_m a = a$
  - **Additive Inverses:** If $a \neq 0 \in \mathbb{Z}_m$, then $m - a$ is an additive inverse of $a \textbf{ mod } m$ and 0 is its additive inverse. That is $a +_m (m - a) = 0$ and $0 +_m 0 = 0$
  - **Distributivity:** If $a, b, c \in \mathbb{Z}_m$, then $a \cdot_m (b +_m c) = (a \cdot_m b) +_m (a \cdot_m c)$ and $(a +_m b) \cdot_m c = (a \cdot_m b) +_m (b \cdot_m c)$

Integer Representations

CS 1187

Integer Representations

- **Base *b* expansion of *n*:** Let b be an integer > 1. Then if n is a positive integer, it can be expressed uniquely in the form

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

Where k is a nonnegative integer, $a_0, a_1, \ldots, a_k$ are nonnegative integers less than b, and $a_k \neq 0$

- **Notes:**
  - A binary digit is called a *bit*
  - 8 bits = 1 *byte* = 2 hexadecimal digits

Number Systems, Common Expansions to Convert to Decimal:

- *Decimal (b=10):* $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  $(716)_{10} = 7 \cdot 10^2 + 1 \cdot 10^1 + 6 = 716$
- *Octal (b=8):* $\{0, 1, 2, 3, 4, 5, 6, 7\}$
  $(7016)_8 = 7 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8 + 6 = 3598$
- *Hexadecimal (b=16):* $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
  $(2AE0B)_{16} = 2 \cdot 16^4 + 10 \cdot 16^3 + 14 \cdot 16^2 + 0 \cdot 16 + 11 = 175627$
- *Binary (b=2):* $\{0, 1\}$
  $(10110)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 = 22$

ROAR

Base Conversion

- We can construct the base b expansion of integer n as follows
  1. divide n by b to obtain a quotient and remainder ($a_0 =$ rightmost digit in expansion)
     $n = bq_0 + a_0 \; 0 \leq a_0 < b$
  2. divide q_0 by b to obtain (a_1 is second rightmost digit)
     $q_0 = bq_1 + a_1 \; 0 \leq a_1 < b$
  3. Continue using these steps moving until you end with a quotient of zero.

- Find octal expansion of $(12345)_{10}$

$$\begin{aligned} 12345 &= 8 \cdot 1543 + 1 \\ 1543 &= 8 \cdot 192 + 7 \\ 192 &= 8 \cdot 24 + 0 \\ 24 &= 8 \cdot 3 + 0 \\ 3 &= 8 \cdot 0 + 3 \\ &= (30071)_8 \end{aligned}$$

- Find the hexadecimal expansion of $(177130)_{10}$

$$\begin{aligned} 177130 &= 16 \cdot 11070 + 10 \\ 11070 &= 16 \cdot 691 + 14 \\ 691 &= 16 \cdot 43 + 3 \\ 43 &= 16 \cdot 2 + 11 \\ 2 &= 16 \cdot 0 + 2 \\ &= (2B3EA)_{16} \end{aligned}$$

Base Conversion

Algorithm: Constructing Base b Expansions (*greedy algorithm*)

procedure BASEBEXPANSION(n, b)

$q := n$

$k := 0$

while $q \neq 0$ **do**

$a_k := q$ **mod** b

$q := q$ **div** b

$k := k + 1$

return $(a_{k-1}, \ldots, a_1, a_0)$

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

Base Conversion

- Binary $\rightarrow$ Oct or Hex: easy since each octal digit is a block of 3 bits and each hex digit is a block of 4 bits:
  - Thus, we simply separate the bit string into appropriately sized groups and convert to the number system
- Conversion to binary is a simple lookup on the table

- Ex: Convert $(11111010111100)_2$ to both octal and hexadecimal

$$\begin{array}{ccccc} 011 & 111 & 010 & 111 & 100 \\ 3 & 7 & 2 & 7 & 4 \end{array}$$
$$= (37274)_8$$

$$\begin{array}{cccc} 0011 & 1110 & 1011 & 1100 \\ 3 & E & B & C \end{array}$$
$$= (3EBC)_{16}$$

- Ex: Convert $(765)_8$ and $(A8D)_{16}$ to Binary

$$\begin{aligned} (765)_8 &= (111\ 110\ 101)_2 \\ (A8D)_{16} &= (1010\ 1000\ 1101)_2 \end{aligned}$$

Addition Algorithm

Algorithm:

procedure ADD(a, b: positive integers)

$\triangleright$ the binary expansions of a and b are $(a_{n-1}a_{n-1}\ldots a_1a_0)_2$ and $(b_{n-1}b_{n-1}\ldots b_1b_0)_2$

$c := 0$

for $j := 0$ **to** $n - 1$ **do**

$d := \lfloor (a_j + b_j + c)/2 \rfloor$

$s_j := a_j + b_j + c - 2d$

$c := d$

$s_n := c$

return $(s_0, s_1, \ldots, s_n)$

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

$$\begin{aligned} a_0 + b_0 &= 0 + 1 = 0 \cdot 2 + 1 & c_0 &= 0, s_0 = 1 \\ a_1 + b_1 + c_0 &= 1 + 1 + 0 = 1 \cdot 2 + 0 & c_1 &= 1, s_1 = 1 \\ a_2 + b_2 + c_1 &= 1 + 0 + 1 = 1 \cdot 2 + 0 & c_2 &= 1, s_2 = 0 \\ a_3 + b_3 + c_2 &= 1 + 1 + 1 = 1 \cdot 2 + 1 & c_3 &= 1, s_3 = 1 \\ s_4 &= c_3 = 1 & s &= a + b = (11001) \end{aligned}$$

- **Analysis:**
  - Each pair of bits and the carry requires 2 bit additions but less than twice the number of bits in the expansion
  - Therefore, **O(n)**

Multiplication Algorithm

Algorithm:

procedure MULTIPLY(a, b: positive integers)

$\triangleright$ the binary expansions of a and b are $(a_{n-1}a_{n-1}\dots a_1a_0)_2$ and $(b_{n-1}b_{n-1}\dots b_1b_0)_2$

for $j := 0$ **to** $n-1$ **do**

  if $b_j = 1$ **then** $c_j :=$ a shifted j places

  else $c_j := 0$

$\triangleright$ $c_0, c_1, \dots, c_{n-1}$ are the partial products

$p := 0$

for $j := 0$ **to** $n-1$ **do**

  $p :=$ ADD(p, c_j)

return p

- Example: Find the product of $a = (110)_2$, $b = (101)_2$

$$
\begin{aligned}
ab_0 = 2^0 = (110)_2 \cdot 1 \cdot 2^0 &= (110)_2\\
ab_1 \cdot 2^1 = (110)_2 \cdot 0 \cdot 2^1 &= (0000)_2\\
ab_2 \cdot 2^2 = (110)_2 \cdot 1 \cdot 2^2 &= (11000)_2\\
&= (11000)_2
\end{aligned}
$$

- **Analysis**
  - First for loop requires $O(n^2)$ shifts
  - Second for loop requires n $O(n)$ additions which is $O(n^2)$
  - The combination is $O(n^2) + O(n^2)$ which is $O(n^2)$

Modular Exponentiation

- Important for crypto is the ability to efficiently calculate b^n **mod** m without requiring a large amount of memory.
  - b, n, and m are integers

Algorithm:

procedure MODEXP(b: integer, $n = (a_{k-1}a_{k-2}\ldots a_1a_0)_2$, m: positive integer)
  $x := 1$
  $power := b$ **mod** m
  for $i := 0$ **to** $k-1$ **do**
    if $a_i = 1$ **then** $x := (x \cdot power)$ **mod** m
    $power := (power \cdot power)$ **mod** m
  return x

- **Analysis:** uses $O((\log m)^2 \log n)$ bit operations–>

ROAR

Modular Exponentiation

- Example: Using the algorithm to find $3^{644} \textbf{ mod } 645$

$$
\begin{aligned}
i = 0 &: a_0 = 0, x = 1, power = 3^2 \textbf{ mod } 645 = 9 \textbf{ mod } 645 = 9 \\
i = 1 &: a_1 = 0, x = 1, power = 9^2 \textbf{ mod } 645 = 81 \textbf{ mod } 645 = 81 \\
i = 2 &: a_2 = 1, x = 1 \cdot 81 \textbf{ mod } 645 = 81, power = 81^2 \textbf{ mod } 645 = 6561 \textbf{ mod } 645 = 111 \\
i = 3 &: a_3 = 0, x = 81, power = 111^2 \textbf{ mod } 645 = 12321 \textbf{ mod } 645 = 66 \\
i = 4 &: a_4 = 0, x = 81, power = 66^2 \textbf{ mod } 645 = 4356 \textbf{ mod } 645 = 486 \\
i = 5 &: a_5 = 0, x = 81, power = 486^2 \textbf{ mod } 645 = 236196 \textbf{ mod } 645 = 126 \\
i = 6 &: a_6 = 0, x = 81, power = 126^2 \textbf{ mod } 645 = 15876 \textbf{ mod } 645 = 396 \\
i = 7 &: a_7 = 1, x = (81 \cdot 396) \textbf{ mod } 645 = 471, power = 396^2 \textbf{ mod } 645 = 156816 \textbf{ mod } 645 = 81 \\
i = 8 &: a_8 = 0, x = 471, power = 81^2 \textbf{ mod } 645 = 6561 \textbf{ mod } 645 = 111 \\
i = 9 &: a_9 = 1, x = (471 \cdot 111) \textbf{ mod } 645 = 36
\end{aligned}
$$

Result: $3^{644} \textbf{ mod } 645 = 36$

Primes and GCD

CS 1187

Primes

- **Prime:** An integer p greater than 1 where the only positive factors of p are 1 and p
- **Composite:** A positive integer that is greater than one and not prime
- *Note:* 1 is not prime, as it only has one *positive* factor
- **Fundamental Theorem of Arithmetic:** Every integer greater than 1 can be written uniquely as a prime or as the product of two or more primes, where the prime factors are written in order of non-decreasing size.
- *Example:* Some prime factorizations
  - $100 = 2 \cdot 2 \cdot 5 \cdot 5 = 2^2 5^2$
  - $641 = 641$
  - $999 = 3 \cdot 3 \cdot 3 \cdot 37 = 3^3 37$
  - $1024 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^{10}$

Trial Division

- **Theorem:** If n is a composite integer, then n has a prime divisor less than or equal to $\sqrt{n}$
- This leads to a *brute-force algorithm* called **trial division** for showing a number is prime
    1. divide n by all primes not exceeding $\sqrt{n}$
    2. conclude n is prime if it is not divisible by any of these prime numbers
    3. otherwise continue dividing by primes to extract the prime factorization
- Examples:
    - Show 101 is prime: primes $< \sqrt{101}$ are $2, 3, 5, 7$ and 101 is not divisible by any of them $\therefore 101$ is prime
    - Factor 7007: $7007 = 7 \cdot 1001 = 7 \cdot 7 \cdot 143 = 7 \cdot 7 \cdot 11 \cdot 13 = 7^2 \cdot 11 \cdot 13$

Sieve of Eratosthenes

- used to find all primes not exceeding a specified positive number, *n*
- start by finding $\sqrt{n}$, which the largest prime factor of *n* cannot exceed.

Example: 100, $\sqrt{100} = 10$

1	2	3	5	7	9
11		13	15	17	19
21		23	25	27	29
31		33	35	37	39
41		43	45	47	49
51		53	55	57	59
61		63	65	67	69
71		73	75	77	79
81		83	85	87	89
91		93	95	97	99

remove all numbers divisible by 2 (except 2)

1	2	3	5	7	
11		13		17	19
		23	25		29
31			35	37	
41		43		47	49
		53	55		59
61			65	67	
71		73		77	79
		83	85		89
91			95	97	

remove all numbers divisible by 3 (except 3)

Sieve of Eratosthenes

1	2	3	5	7	
11		13		17	19
		23			29
31				37	
41		43		47	49
		53			59
61				67	
71		73		77	79
		83			89
91				97	

remove all numbers divisible by 5 (except 5)

1	2	3	5	7	
11		13		17	19
		23			29
31				37	
41		43		47	
		53			59
61				67	
71		73			79
		83			89
				97	

remove all numbers divisible by 7 (except 7), and all remaining numbers are prime

Infinitude of Primes

- **Theorem:** There are infinitely many primes
- **Mersenne Primes:** Primes of the form $2^p - 1$, where *p* is also prime
  - Note that $2^n - 1$ cannot be prime unless *n* is also prime
  - There is an extremely efficient test to determine if $2^p - 1$ is prime (Lucas-Lehmer Test)
  - Examples:
    - $2^2 - 1 = 3$, $2^3 - 1 = 7$, $2^5 - 1 = 31$, $2^7 - 1 = 127$ are all Mersenne primes
    - $2^{1}1 - 1 = 2047 = 23 \cdot 89 \therefore$ not prime

Distribution of Primes

- **Prime Number Theorem:** The ration of $\pi(x)$, the number of primes not exceeding x, and $x/\ln x$ approaches 1 as x grows without bound.
  - Using Trial Division with this theorem does provide a method for factoring and primality testing, but not a very efficient one
  - However, there is a polynomial time algorithm for determining if a number is prime. It was identified by Agrawal, Kayal, and Saxena
    - Runs in $O((\log n)^6)$ operations
  - Unfortunately factoring large numbers is still exceptionally difficult

GCD and LCM

Idaho State University | Computer Science

- **Greatest Common Divisor (gcd):** Let a and b be intergers not both zero. The *gcd* is the largest integer d such that $d \mid a$ and $d \mid b$. Denoted $\text{gcd}(a, b)$
  - Example: What is the gcd of 24 and 36?
    - Common divisors are: 1, 2, 3, 4, 6, and 12
    - Thus, $\text{gcd}(24, 36) = 12$
- **Least Common Multiple (lcm):** of the positive integers a and b is the smallest positive integer that is divisible by both a and b. Denoted $\text{lcm}(a, b)$
- **Theorem:** Let a and b be positive integers. Then

$$ab = \text{gcd}(a, b) \cdot \text{lcm}(a, b)$$

GCD, LCM, and Primes

Idaho State University | Computer Science

- **Relatively Prime:** The intgers a and b are *relatively prime* if $\gcd(a, b) = 1$
- **Pairwise Relatively Prime:** The integers $a_1, a_2, \ldots, a_n$ are *pairwise relatively prime* if $\gcd(a_i, a_j) = 1$ whenever $1 \leq i < j \leq n$
- *Examples:*
  - $\gcd(17, 22) = 1 \therefore$ 17 and 22 are relatively prime
  - Are 10, 17, 21 Pairwise relatively prime?
    - $\gcd(10, 17) = 1$
    - $\gcd(10, 21) = 1$
    - $\gcd(17, 21) = 1$
    - $\therefore$ they are pairwise relatively prime

GCD, LCM, and Primes

Idaho State University | Computer Science

- We can use the prime factorizations of the positive integers a and b to find both the $\gcd(a, b)$ and $\text{lcm}(a, b)$
- Suppose the prime factorizations of a and b are:

$$\begin{aligned} a &= p_1^{a_1} p_c^{a_2} \dots p_n^{a_n} \\ b &= p_1^{b_1} p_c^{b_2} \dots p_n^{b_n} \end{aligned}$$

  where all primes occurring in either a or b are listed (possibly with zero exponents)
- Then:

$$\begin{aligned} \gcd(a, b) &= p_1^{\min(a_1, b_1)} p_2^{\min(a_2, b_2)} \dots p_n^{\min(a_n, b_n)} \\ \text{lcm}(a, b) &= p_1^{\max(a_1, b_1)} p_2^{\max(a_2, b_2)} \dots p_n^{\max(a_n, b_n)} \end{aligned}$$

- Example: What is the lcm of $2^3 3^5 7^2$ and $2^4 3^3$?
  - $\text{lcm}(2^3 3^5 7^2, 2^4 3^3) = 2^{\max(3,4)} 3^{\max(5,3)} 7^{\max(2,0)} = 2^4 3^5 7^2$

The Euclidean Algorithm

Idaho State University | Computer Science

- An efficient algorithm for computing gcd, known since ancient times, is based on the following lemma
- **Lemma:** Let $a = bq + r$, where a, b, q, and r are integers. Then $\gcd(a, b) = \gcd(b, r)$

Algorithm:

procedure GCD(a, b: positive integers)
  $x := a$
  $y := b$
  while $y \neq 0$ **do**
    $r := x$ **mod** y
    $x := y$
    $y := r$
  return x

- If $a \geq b$, then $O(\log b)$

Example: Find $\gcd(414, 662)$

j	r_j	r_{j+1}	q_{j+1}	r_{j+2}
0	662	414	1	248
1	414	248	1	166
2	248	166	1	82
3	166	82	2	2
4	82	2	41	0

$\gcd(414, 662) = 2$

GCDs: Linear Combinations

Idaho State University | Computer Science

- **B‘{e}zout’s Theorem:** If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$
  - The coefficients s and t are called **Bézout’s coefficients** of a and b
  - The equation $\gcd(a, b) = sa + tb$ is called **Bézout’s identity**
    - This form shows that $\gcd(a, b)$ can be expressed as a *linear combination*
- **Lemma:** If a, b, and c are positive integers such that $\gcd(a, b) = 1$ and $a \mid bc$, then $a \mid c$
- **Lemma:** If p is a prime and $p \mid a_1 a_2 \ldots a_n$, where each a_i is an integer, then $p \mid a_i$ for some i
- **Theorem:** Let m be a positive integer and let a, b, and c be integers. If $ac \equiv bc \pmod{m}$ and $\gcd(c, m) = 1$, then $a \equiv b \pmod{m}$

Solving Congruences

CS 1187

ROAR

Linear Congruences

Idaho State University | Computer Science

- **Linear Congruence:** A congruence of the form

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

  Where:
  - m is a positive integer
  - a, b are integers
  - x is a variable
- **Theorem:** If a and m are relatively prime integers and $m > 1$, then an *inverse* of $a \bmod m$ exists. Furthermore, this inverse is unique modulo m
  - That si, there is a unique positive integer $\bar{a}$ less than m that is the inverse of $a \bmod m$ and every other inverse of $a \bmod m$ is congruent to $\bar{a} \bmod m$
- Once we have an inverse, $\bar{a}$ of $a \bmod m$, we can solve the congruence $ax \equiv b \pmod{m}$ by multiplying both sides by $\bar{a}$

Linear Congruences

- Example: Find an inverse of $3 \textbf{ mod } 7$ by first finding Bézout coefficients of 3 and 7
  - Because gcd$(3, 7) = 1$, an inverse of $3 \textbf{ mod } 7$ exists
    - gcd$(3, 7) \Rightarrow 7 = 2 \cdot 3 + 1$
    - from this $-2 \cdot 3 + 1 \cdot 7 = 1$
  - Bézout coefficients are -2 and 1
  - Then, -2 is an inverse of $3 \textbf{ mod } 7$

Chinese Remainder Theorem

- **Chinese Remainder Theorem:** Let $m_1, m_2, \ldots, m_n$ be pairwise relatively prime positive integers greater than 1 and $a_1, a_2, \ldots, a_n$ arbitrary integers. Then the system

$$\begin{aligned} x &\equiv a_1 \pmod{m_1}, \\ x &\equiv a_2 \pmod{m_2}, \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

has a unique solution modulo $m = m_1 m_2 \ldots m_n$

Fermat's Little Theorem

Idaho State University | Computer Science

- **Fermat's Little Theorem:** If p is prime and a is an integer not divisible by p, then

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

  Furthermore, for every integer a we know

$$a^p \equiv a \pmod{p}$$

- *Example:* find $7^{222} \bmod 11$
  - Using Fermat's Little Theorem we know that $7^{10} \equiv 1 \pmod{11}$, so $(7^{10})^k \equiv 1 \pmod{11}\ \forall k \in \mathbb{Z}^+$
  - We then divide 222 by 10 finding $222 = 22 \cdot 10 + 2$
  - We can then see that $7^{222} = 7^{22 \cdot 10 + 2} \equiv (1)^{22} \cdot 40 \equiv 5 \pmod{11}$
  - Thus, $7^{222} \bmod 11 = 5$–>

Pseudoprimes

- **Pseudoprime:** Let b be a positive integer. If n is a composite positive integer, and $b^{n+1} \equiv 1 \pmod{n}$, then n is called *pseudoprime to the base b*
- **Carmichael Number:** A composite integer n that satisfies the congruence $b^{n-1} \equiv 1 \pmod{n}$ for all positive integer b with $\gcd(b, n) = 1$

Applications of Congruences

CS 1187

Hashing Functions

- **Hashing Functions:** a hashing function h assigns memory location $h(k)$ to the record that has k as its key.
  - Many different hashing functions are used in practice, one of the most common is:

    $$h(k) = k \textbf{ mod } m$$

    Where m is the number of memory locations
  - Should be easy to evaluate
  - Should be onto, so all memory locations are pairwise
- The functions are not one-to-one (more possible keys than memory locations) thus **collisions** may occur
  - Collision handling is necessary
  - Assign first free location following memory location assigned by hashing function:

    $$h(k,i) = h(k) + i \textbf{ mod } m \quad 0 \leq i \leq m-1$$

Hashing Functions

- Example: Find the memory location assigned by the hashing function $h(k) = k \textbf{ mod } 111$ to the records of customers with social security numbers 064212848 and 037149212.

$$\begin{aligned} h(064212848) &= 064212848 \textbf{ mod } 111 = 14 \\ h(037149212) &= 037149212 \textbf{ mod } 111 = 65 \end{aligned}$$

- Example: Now assign a memory location to the record of the customer with SSN 107405723

$$h(107405723) = 107405723 \textbf{ mod } 111 = 14$$

This caused a collision. However, 15 is unassigned, thus we can assign 10740523 to 15 instead.

Pseudorandom Numbers

- **Pseudorandom Numbers:** A sequence of numbers systematically generated and having several properties of randomly selected numbers, without being truly random.
  - Need for computer simulations
- **Linear Congruential Method:** most commonly used procedure for generating pseudorandom numbers. Uses the following recursively defined function:

$$x_{n+1} = (ax_n + c) \text{ mod } m$$

Where: we select the following integers

- m is the *modulus*
- a is the *multiplier*
- c is the *increment*
- x_0 is the *seed* (initial value)
- and $2 \leq a < m$, $0 \leq c < m$, and $0 \leq x_0 < m$

Pseudorandom Numbers

- Example: Using the function $x_{n+1} = (7x_n + 4) \textbf{ mod } 9$, $x_0 = 3$, we find that

$$
\begin{aligned}
x_1 &= 7x_0 + 4 \textbf{ mod } 0 = 7 \cdot 3 + 4 \textbf{ mod } 9 = 25 \textbf{ mod } 9 = 7 \\
x_2 &= 7x_1 + 4 \textbf{ mod } 0 = 7 \cdot 7 + 4 \textbf{ mod } 9 = 53 \textbf{ mod } 9 = 8 \\
x_3 &= 7x_2 + 4 \textbf{ mod } 0 = 7 \cdot 8 + 4 \textbf{ mod } 9 = 60 \textbf{ mod } 9 = 6 \\
x_4 &= 7x_3 + 4 \textbf{ mod } 0 = 7 \cdot 6 + 4 \textbf{ mod } 9 = 46 \textbf{ mod } 9 = 1 \\
x_5 &= 7x_4 + 4 \textbf{ mod } 0 = 7 \cdot 1 + 4 \textbf{ mod } 9 = 11 \textbf{ mod } 9 = 2 \\
x_6 &= 7x_5 + 4 \textbf{ mod } 0 = 7 \cdot 2 + 4 \textbf{ mod } 9 = 18 \textbf{ mod } 9 = 0 \\
x_7 &= 7x_6 + 4 \textbf{ mod } 0 = 7 \cdot 0 + 4 \textbf{ mod } 9 = 4 \textbf{ mod } 9 = 4 \\
x_8 &= 7x_7 + 4 \textbf{ mod } 0 = 7 \cdot 4 + 4 \textbf{ mod } 9 = 32 \textbf{ mod } 9 = 5 \\
x_9 &= 7x_8 + 4 \textbf{ mod } 0 = 7 \cdot 5 + 4 \textbf{ mod } 9 = 39 \textbf{ mod } 9 = 3
\end{aligned}
$$

- This generates the sequence: **3**,7,8,6,1,2,0,4,5,**3**,7,8,6,1,2,0,4,5,**3**,. . .
- The sequence has a cyle of 9 before repeating

Cryptography

CS 1187

ROAR

Classical Cryptography

- **Encryption:** The process of making a message secret
- **Decryption:** The process of determining the original message from the encrypted text
- **Caesar's Encryption Method**
  - Replace each letter with its corresponding number from $\mathbb{Z}_26$
  - Encrypt using the function $f(p)$, where p is a nonnegative integer less than or equal to 25.

$$f(p) = (p + 3) \textbf{ mod } 26$$

  - Replace the encrypted numbers with their corresponding letters
  - This is a form of a *shift cipher*

Classical Cryptography

- Example: Use Caesar's cipher to encrypt th emessage "MEET YOU IN THE PARK"

```
       M E E T    Y  O  U   I  N   T H  E   P A  R  K
1.)   12 4 4 19  24 14 20   8 13  19 7  4  15 0 17 10  // letters -> num
2.)   15 7 7 22   1 17 23  11 16  22 10 7  18 3 20 13  // encrypt with f(p)
3.)    P H H  W   B  R  X   L  Q   W  K H   S D  U  N  // num -> letters
```

- We can recover the original message, by shifting the letters back by 3 using:

$$f^{-1}(p) = (p - 3) \textbf{ mod } 26$$

Generalized Shift Cipher

- *Encryption:* Shift letters by k letters

$$f(p) = (p + k) \textbf{ mod } 26$$

- *Decryption:* Shifts letters by $-k$ letters

$$f^{-1}(p) = (p - k) \textbf{ mod } 26$$

- **Affine Cipher:** An enahncement of the shift cipher which provides additional security and uses the following formula

$$f(p) = (ap + b) \textbf{ mod } m$$

  Where a and b are selected so that f is a bijection. Such a mapping is called an *affine transformation*
- **Cryptanalysis:** The process of recovering plaintext from ciphertext without knowledge of both the encryption method and the key.
  - A technique used agains shift and affine ciphers is the frequency attack

Block Ciphers and Beyond

- **Character or Monoalphabetic Ciphers:** ciphers which proceed by replacing each letter of the alphabet by another letter of the alphabet
  - Shift and affine ciphers are of this type
  - Vulnerable to attack
- To combat these deficiencies better ciphers hve been developed
  - For example, Block Ciphers
  - However, they are still prone to attack
- To improve our ability to encrypt and keep data safe, we have developed better methods such as the AES and RSA private key cryptosystems
- Additionally, we have developed public-private key cryptosystems such as gpg.
- If you are interested in these topic I would suggest starting with DMA Chapter 4.6

Are there any questions?