CS215 DISCRETE MATH

Dr. Qi Wang

Department of Computer Science and Engineering
Office: Room413, CoE South Tower
Email: wangqi@sustech.edu.cn

Representations of Integers

- We may use *decimal* (*base* 10) or *binary* or *octal* or *hexadecimal* or other notations to represent integers.

- Let $b > 1$ be an integer. 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 nonnegative, a_i's are nonnegative integers less than b. The representation of n is called *the base-b expansion of n* and is denoted by $(a_k a_{k-1} \ldots a_1 a_0)_b$.

Base-b Expansions

- To get the decimal expansion is easy.

Base-b Expansions

- To get the decimal expansion is easy.

Example

- ◇ $(101011111)_2 = 2^8 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 351$
- ◇ $(7016)_8 = 7 \cdot 8^3 + 1 \cdot 8 + 6 = 3598$

Base-b Expansions

- To get the decimal expansion is easy.

Example

◇ $(101011111)_2 = 2^8 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 351$

◇ $(7016)_8 = 7 \cdot 8^3 + 1 \cdot 8 + 6 = 3598$

- Conversions between binary, octal, hexadecimal expansions are easy.

Base-b Expansions

- To get the decimal expansion is easy.

  Example

  ◇ $(101011111)_2 = 2^8 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 351$

  ◇ $(7016)_8 = 7 \cdot 8^3 + 1 \cdot 8 + 6 = 3598$

- Conversions between binary, octal, hexadecimal expansions are easy.

  Example

  ◇ $(101011111)_2 = (\underline{101}\overline{011}\underline{111}) = (537)_8$

  ◇ $(7016)_8 = (\underline{111}\overline{000}\underline{001}\overline{110})_2$
  $= (\underline{1110}\overline{0000}\underline{1110})_2 = (E0E)_{16}$

Base-b Expansions

$$
\begin{aligned}
n &= a_k b^k + a_{k-1} b^{k-1} + a_{k-2} b^{k-2} + \cdots + a_2 b^2 + a_1 b + a_0 \\
&= b(a_k b^{k-1} + a_{k-1} b^{k-2} + a_{k-2} b^{k-3} + \cdots + a_2 b + a_1) + {\color{red} a_0} \\
&= b(b(a_k b^{k-2} + a_{k-1} b^{k-3} + a_{k-2} b^{k-4} + \cdots + a_2) + {\color{red} a_1}) + {\color{blue} a_0} \\
&= \cdots
\end{aligned}
$$

Base-b Expansions

$$\begin{aligned} n &= a_k b^k + a_{k-1} b^{k-1} + a_{k-2} b^{k-2} + \cdots + a_2 b^2 + a_1 b + a_0 \\ &= b(a_k b^{k-1} + a_{k-1} b^{k-2} + a_{k-2} b^{k-3} + \cdots + a_2 b + a_1) + a_0 \\ &= b(b(a_k b^{k-2} + a_{k-1} b^{k-3} + a_{k-2} b^{k-4} + \cdots + a_2) + a_1) + a_0 \\ &= \cdots \end{aligned}$$

To construct the base-b expansion of an integer n,

- Divide n by b to obtain $n = bq_0 + a_0$, with $0 \leq a_0 < b$
- The remainder a_0 is the rightmost digit in the base-b expansion of n. Then divide q_0 by b to get $q_0 = bq_1 + a_1$ with $0 \leq a_1 < b$
- a_1 is the second digit from the right. Continue by successively dividing the quotients by b until **the quotient is** 0

Algorithm: Constructing Base-b Expansions

procedure *base b expansion*(n, b: positive integers with $b > 1$)
$q := n$
$k := 0$
while ($q \neq 0$)
    $a_k := q$ **mod** b
    $q := q$ **div** b
    $k := k + 1$
return($a_{k-1}, \ldots, a_1, a_0$){$(a_{k-1} \ldots a_1 a_0)_b$ is base b expansion of n}

Example

- $(12345)_{10} = (30071)_8$

Example

- $(12345)_{10} = (30071)_8$

$$12345 = 8 \cdot 1543 + 1$$

$$1543 = 8 \cdot 192 + 7$$

$$192 = 8 \cdot 24 + 0$$

$$24 = 8 \cdot 3 + 0$$

$$3 = 8 \cdot 0 + 3$$

Binary Addition of Integers

$a = (a_{n-1}a_{n-2}\ldots a_1 a_0)$, $b = (b_{n-1}b_{n-2}\ldots b_1 b_0)$

procedure *add*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1},a_{n-2},\ldots,a_0)_2$ and $(b_{n-1},b_{n-2},\ldots,b_0)_2$, respectively}
$c := 0$
for $j := 0$ to $n-1$
    $d := \lfloor (a_j + b_j + c)/2 \rfloor$
    $s_j := a_j + b_j + c - 2d$
    $c := d$
$s_n := c$
return$(s_0, s_1, \ldots, s_n)$ {the binary expansion of the sum is $(s_n, s_{n-1}, \ldots, s_0)_2$}

Binary Addition of Integers

$a = (a_{n-1}a_{n-2}\ldots a_1a_0)$, $b = (b_{n-1}b_{n-2}\ldots b_1b_0)$

procedure *add*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1},a_{n-2},\ldots,a_0)_2$ and $(b_{n-1},b_{n-2},\ldots,b_0)_2$, respectively}
$c := 0$
for $j := 0$ to $n - 1$
    $d := \lfloor (a_j + b_j + c)/2 \rfloor$
    $s_j := a_j + b_j + c - 2d$
    $c := d$
$s_n := c$
return$(s_0, s_1, \ldots, s_n)$ {the binary expansion of the sum is $(s_n, s_{n-1}, \ldots, s_0)_2$}

$O(n)$ bit additions

Algorithm: Binary Multiplication of Integers

$a = (a_{n-1}a_{n-2}\ldots a_1 a_0)_2,\ b = (b_{n-1}b_{n-2}\ldots b_1 b_0)_2$

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

procedure *multiply*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1},a_{n-2},\ldots,a_0)_2$ and $(b_{n-1},b_{n-2},\ldots,b_0)_2$, respectively}
for $j := 0$ to $n-1$
    if $b_j = 1$ **then** $c_j = a$ shifted j places
    else $c_j := 0$
{$c_0, c_1, \ldots, c_{n-1}$ are the partial products}
$p := 0$
for $j := 0$ to $n-1$
    $p := p + c_j$
return p {p is the value of ab}

Algorithm: Binary Multiplication of Integers

$a = (a_{n-1}a_{n-2}\ldots a_1a_0)_2,\ b = (b_{n-1}b_{n-2}\ldots b_1b_0)_2$

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

procedure *multiply*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1},a_{n-2},\ldots,a_0)_2$ and $(b_{n-1},b_{n-2},\ldots,b_0)_2$, respectively}
for $j := 0$ to $n - 1$
    if $b_j = 1$ **then** $c_j = a$ shifted j places
    else $c_j := 0$
{$c_0, c_1, \ldots, c_{n-1}$ are the partial products}
$p := 0$
for $j := 0$ to $n - 1$
  $p := p + c_j$
return p {p is the value of ab}

$O(n^2)$ shifts and $O(n^2)$ bit additions

Algorithm: Computing div and mod

procedure *division algorithm* (a: integer, d: positive integer)
$q := 0$
$r := |a|$
while $r \geq d$
    $r := r - d$
    $q := q + 1$
if $a < 0$ and $r > 0$ **then**
    $r := d - r$
    $q := -(q+1)$
return (q, r) {$q = a$ **div** d *is the quotient*, $r = a$ **mod** d *is the remainder* }

Algorithm: Computing div and mod

procedure *division algorithm* (a: integer, d: positive integer)
$q := 0$
$r := |a|$
while $r \geq d$
    $r := r - d$
    $q := q + 1$
if $a < 0$ and $r > 0$ **then**
    $r := d - r$
    $q := -(q+1)$
return (q, r) {$q = a$ **div** d *is the quotient*, $r = a$ **mod** d *is the remainder* }

$O(q \log a)$ bit operations. But there exist more efficient algorithms with complextiy $O(n^2)$, where $n = \max(\log a, \log d)$

Algorithm: Computing div and mod (cont)

- **procedure** *division*2 $(a, d \in \mathbb{N},\ d \geq 1)$
  if $a < d$
      return $(q, r) = (0, a)$
  $(q, r) = division2(\lfloor a/2 \rfloor, d)$
  $q = 2q,\ r = 2r$
  if a is odd
      $r = r + 1$
  if $r \geq d$
      $r = r - d$
      $q = q + 1$
  return (q, r)

Algorithm: Computing div and mod (cont)

- **procedure** $division2\ (a, d \in \mathbb{N},\ d \geq 1)$
  if $a < d$
      return $(q, r) = (0, a)$
  $(q, r) = division2(\lfloor a/2 \rfloor, d)$
  $q = 2q,\ r = 2r$
  if a is odd
      $r = r + 1$
  if $r \geq d$
      $r = r - d$
      $q = q + 1$
  return (q, r)

  $O(\log^2 a)$ bit operations.

Algorithm: Binary Modular Exponentiation

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

Successively finds $b \bmod m$, $b^2 \bmod m$, $b^4 \bmod m$, ..., $b^{2^{k-1}} \bmod m$, and multiplies together the terms b^{2^j} where $a_j = 1$.

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

Algorithm: Binary Modular Exponentiation

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

Successively finds $b \bmod m$, $b^2 \bmod m$, $b^4 \bmod m$, $\ldots$, $b^{2^{k-1}} \bmod m$, and multiplies together the terms b^{2^j} where $a_j = 1$.

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

$O((\log m)^2 \log n)$ bit operations

Primes

- A positive integer p that is greater than 1 and is divisible only by 1 and by itself is called a *prime*.

Primes

- A positive integer p that is greater than 1 and is divisible only by 1 and by itself is called a *prime*.

- A positive integer p that is greater than 1 and is not a prime is called a *composite*.

Primes

- A positive integer p that is greater than 1 and is divisible only by 1 and by itself is called a *prime*.

- A positive integer p that is greater than 1 and is not a prime is called a *composite*.

- **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 nondecreasing size.

Primes and Composites

- How to determine whether a number is a prime or a composite?

Primes and Composites

- How to determine whether a number is a prime or a composite?

  Approach 1: test if each number $x < n$ divides n.

Primes and Composites

- How to determine whether a number is a prime or a composite?

Approach 1: test if each number $x < n$ divides n.

Approach 2: test if each prime number $x < n$ divides n.

Primes and Composites

- How to determine whether a number is a prime or a composite?

Approach 1: test if each number $x < n$ divides n.

Approach 2: test if each prime number $x < n$ divides n.

Approach 3: test if each prime number $x \leq \sqrt{n}$ divides n.

Primes and Composites

- If n is composite, then n has a prime divisor less than or equal to $\sqrt{n}$.

- If n is composite, then n has a prime divisor less than or equal to $\sqrt{n}$.

Proof.

◇ if n is composite, then it has a positive integer factor a such that $1 < a < n$ by definition. This means that $n = ab$, where b is an integer greater than 1.

◇ assume that $a > \sqrt{n}$ and $b > \sqrt{n}$. Then $ab > n$, contradiction. So either $a \leq \sqrt{n}$ or $b \leq \sqrt{n}$.

◇ Thus, n has a divisor less than $\sqrt{n}$.

◇ By the Fundamental Theorem of Arithmetic, this divisor is either prime, or is a product of primes. In either case, n has a prime divisor less than $\sqrt{n}$.

Primes

- There are infinitely many primes.

Proof (by contradiction)

Greatest Common Divisor (GCD)

- Let a and b be integers, not both 0. The largest integer d such that $d|a$ and $d|b$ is called the *greatest common divisor* of a and b, denoted by $\gcd(a, b)$.

Greatest Common Divisor (GCD)

- Let a and b be integers, not both 0. The largest integer d such that $d|a$ and $d|b$ is called the *greatest common divisor* of a and b, denoted by $\gcd(a, b)$.

  The integers a and b are *relatively prime* if their greatest common divisor is 1.

Greatest Common Divisor (GCD)

- Let a and b be integers, not both 0. The largest integer d such that $d|a$ and $d|b$ is called the *greatest common divisor* of a and b, denoted by $\gcd(a, b)$.

  The integers a and b are *relatively prime* if their greatest common divisor is 1.

  A systematic way to find the gcd is **factorization**. Let $a = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}$ and $b = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n}$. Then $\gcd(a, b) = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \cdots p_n^{\min(a_k,b_k)}$

Least Common Multiple (LCM)

- Let a and b be integers. The *least common multiple* of a and b is the smallest positive integer that is divisible by both a and b, denoted by $\operatorname{lcm}(a, b)$.

Least Common Multiple (LCM)

- Let a and b be integers. The *least common multiple* of a and b is the smallest positive integer that is divisible by both a and b, denoted by $\text{lcm}(a, b)$.

We can also use **factorization** to find the lcm. Let $a = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}$ and $b = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n}$. Then $\text{lcm}(a, b) = p_1^{\max(a_1, b_1)} p_2^{\max(a_2, b_2)} \cdots p_n^{\max(a_k, b_k)}$

Euclidean Algorithm

- Factorization can be **cumbersome** and **time consuming** since we need to find all factors of the two integers.

Euclidean Algorithm

- Factorization can be **cumbersome** and **time consuming** since we need to find all factors of the two integers.

- Luckily, we have an efficient algorithm, called **Euclidean algorithm**. This algorithm has been known since ancient times and named after the ancient Greek mathmaticain Euclid.

Euclidean Algorithm

- For two integers 287 and 91, we want to find $\gcd(287, 91)$.

Euclidean Algorithm

- For two integers 287 and 91, we want to find $\gcd(287, 91)$.

Step 1: $287 = 91 \cdot 3 + 14$

Euclidean Algorithm

- For two integers 287 and 91, we want to find $\gcd(287, 91)$.

Step 1: $287 = 91 \cdot 3 + 14$

Step 2: $91 = 14 \cdot 6 + 7$

Euclidean Algorithm

- For two integers 287 and 91, we want to find $\gcd(287, 91)$.

Step 1: $287 = 91 \cdot 3 + 14$

Step 2: $91 = 14 \cdot 6 + 7$

Step 3: $14 = 7 \cdot 2 + 0$

Euclidean Algorithm

- For two integers 287 and 91, we want to find $\gcd(287, 91)$.

Step 1: $287 = 91 \cdot 3 + 14$

Step 2: $91 = 14 \cdot 6 + 7$

Step 3: $14 = 7 \cdot 2 + 0$

$$\gcd(287, 91) = \gcd(91, 14) = \gcd(14, 7) = 7$$

Euclidean Algorithm

- The Euclidean algorithm in pseudocode

ALGORITHM 1 The Euclidean Algorithm.

procedure $gcd(a, b$: positive integers)
$x := a$
$y := b$
while $y \neq 0$
    $r := x$ **mod** y
    $x := y$
    $y := r$
return x {gcd(a, b) is x}

Euclidean Algorithm

- The Euclidean algorithm in pseudocode

ALGORITHM 1 The Euclidean Algorithm.

procedure $gcd(a, b$: positive integers)
$x := a$
$y := b$
while $y \neq 0$
    $r := x$ **mod** y
    $x := y$
    $y := r$
return $x\{\gcd(a, b)$ is $x\}$

The number of divisions required to find $\gcd(a, b)$ is $O(\log b)$, where $a \geq b$. (this will be proved later.)

Correctness of Euclidean Algorithm

- **Lemma** Let $a = bq + r$, where a, b, q and r are integers. Then $\gcd(a, b) = \gcd(b, r)$.

Correctness of Euclidean Algorithm

- **Lemma** Let $a = bq + r$, where a, b, q and r are integers. Then $\gcd(a, b) = \gcd(b, r)$.

Proof.

◇ suppose that $d|a$ and $d|b$. Then d also divides $a - bq = r$. Hence, any common divisor of a and b must also be any common divisor of b and r.

◇ suppose that $d|b$ and $d|r$. Then d also divdes $bq + r = a$. Hence, any common divisor of b and r must also be a common divisor of a and b.

◇ Therefore, $\gcd(a, b) = \gcd(b, r)$.

Correctness of Euclidean Algorithm

- Suppose that a and b are positive integers with $a \geq b$. Let $r_0 = a$ and $r_1 = b$.

Correctness of Euclidean Algorithm

- Suppose that a and b are positive integers with $a \geq b$. Let $r_0 = a$ and $r_1 = b$.

$$
\begin{array}{lll}
r_0 & = r_1 q_1 + r_2 & 0 \leq r_2 < r_1, \\
r_1 & = r_2 q_2 + r_3 & 0 \leq r_3 < r_2, \\
 & \vdots & \\
r_{n-2} & = r_{n-1} q_{n-1} + r_n & 0 \leq r_n < r_{n-1}, \\
r_{n-1} & = r_n q_n .
\end{array}
$$

Correctness of Euclidean Algorithm

- Suppose that a and b are positive integers with $a \geq b$. Let $r_0 = a$ and $r_1 = b$.

$$
\begin{aligned}
r_0 &= r_1 q_1 + r_2 & 0 \leq r_2 < r_1, \\
r_1 &= r_2 q_2 + r_3 & 0 \leq r_3 < r_2, \\
&\vdots \\
r_{n-2} &= r_{n-1} q_{n-1} + r_n & 0 \leq r_n < r_{n-1}, \\
r_{n-1} &= r_n q_n .
\end{aligned}
$$

$$\gcd(a, b) = \gcd(r_0, r_1) = \cdots = \gcd(r_{n-1}, r_n) = \gcd(r_n, 0) = r_n$$

GCD as Linear Combinations

- **Bezout's Theorem** If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$. This is called *Bezout's identity*.

GCD as Linear Combinations

- **Bezout's Theorem** If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$. This is called *Bezout's identity*.

We may use *extended Euclidean algorithm* to find Bezout's identity.

GCD as Linear Combinations

- **Bezout's Theorem** If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$. This is called *Bezout's identity*.

We may use *extended Euclidean algorithm* to find Bezout's identity.

Example: Express 1 as the linear combination of 503 and 286.

GCD as Linear Combinations

- **Bezout's Theorem** If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$. This is called *Bezout's identity*.

We may use *extended Euclidean algorithm* to find Bezout's identity.

Example: Express 1 as the linear combination of 503 and 286.

$$503 = 1 \cdot 286 + 217$$
$$286 = 1 \cdot 217 + 69$$
$$217 = 3 \cdot 69 + 10$$
$$69 = 6 \cdot 10 + 9$$
$$10 = 1 \cdot 9 + 1$$

GCD as Linear Combinations

- **Bezout's Theorem** If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$. This is called *Bezout's identity*.

We may use *extended Euclidean algorithm* to find Bezout's identity.

Example: Express 1 as the linear combination of 503 and 286.

$$503 = 1 \cdot 286 + 217$$
$$286 = 1 \cdot 217 + 69$$
$$217 = 3 \cdot 69 + 10$$
$$69 = 6 \cdot 10 + 9$$
$$10 = 1 \cdot 9 + 1$$

$$\begin{aligned} 1 &= 10 - 1 \cdot 9 \\ &= 7 \cdot 10 - 1 \cdot 69 \\ &= 7 \cdot 217 - 22 \cdot 69 \\ &= 29 \cdot 217 - 22 \cdot 286 \\ &= 29 \cdot 503 - 51 \cdot 286 \end{aligned}$$

Corollaries of Bezout's Theorem

- If a, b, c are positive integers such that $\gcd(a, b) = 1$ and $a | bc$, then $a | c$.

Corollaries of Bezout's Theorem

- If a, b, c are positive integers such that $\gcd(a, b) = 1$ and $a | bc$, then $a | c$.

  Proof. Since $\gcd(a, b) = 1$, by Bezout's Theorem there exist s and t such that $1 = sa + tb$. This yields $c = sac + tbc$. Since $a | bc$, we have $a | tbc$, and then $a | (sac + tbc) = c$.

Corollaries of Bezout's Theorem

- If a, b, c are positive integers such that $\gcd(a, b) = 1$ and $a | bc$, then $a | c$.

  Proof. Since $\gcd(a, b) = 1$, by Bezout's Theorem there exist s and t such that $1 = sa + tb$. This yields $c = sac + tbc$. Since $a | bc$, we have $a | tbc$, and then $a | (sac + tbc) = c$.

- If p is prime and $p | a_1 a_2 \cdots a_n$, then $p | a_i$ for some i.

Corollaries of Bezout's Theorem

- If a, b, c are positive integers such that $\gcd(a, b) = 1$ and $a|bc$, then $a|c$.

  Proof. Since $\gcd(a, b) = 1$, by Bezout's Theorem there exist s and t such that $1 = sa + tb$. This yields $c = sac + tbc$. Since $a|bc$, we have $a|tbc$, and then $a|(sac + tbc) = c$.

- If p is prime and $p|a_1a_2 \cdots a_n$, then $p|a_i$ for some i.

  Proof. by induction. Will be given later.

Uniqueness of Prime Factorization

- We prove that a prime factorization of a positive integer where the primes are in nondecreasing order is unique.

Uniqueness of Prime Factorization

- We prove that a prime factorization of a positive integer where the primes are in nondecreasing order is unique.

Proof. (by contradiction) Suppose that the positive integer n can be written as a product of primes in two distinct ways:

$n = p_1 p_2 \cdots p_s$ and $n = q_1 q_2 \cdots q_t$

Remove all common primes from the factorizations to get

$p_{i_1} p_{i_2} \cdots p_{i_u} = q_{j_1} q_{j_2} \cdots q_{j_v}$

It then follows that p_{i_1} divides q_{j_k} for some k, contradicting the assumption that p_{i_1} and q_{j_k} are distinct primes.

Dividing Congruences by an Integer

- **Theorem** Let m be a positive integer and let a, b, c be integers. If $ac \equiv bc \pmod{m}$ and $\gcd(c, m) = 1$, then $a \equiv b \pmod{m}$.

Dividing Congruences by an Integer

- **Theorem** Let m be a positive integer and let a, b, c be integers. If $ac \equiv bc \pmod{m}$ and $\gcd(c, m) = 1$, then $a \equiv b \pmod{m}$.

Proof. Since $ac \equiv bc \pmod{m}$, we have $m | ac - bc = c(a - b)$. Because $\gcd(c, m) = 1$, it follows that $m | a - b$.

Mersenne Primes

- Prime numbers of the form $2^p - 1$, where p is a prime.

Marin Mersenne

Mersenne Primes

- Prime numbers of the form $2^p - 1$, where p is a prime.

  $\diamond$ $2^2 - 1 = 3$, $2^3 - 1 = 7$, $2^5 - 1 = 31$, $2^7 - 1 = 127$ are Mersenne primes.

  $\diamond$ $2^{11} - 1 = 2047 = 23 \cdot 89$ is not a Mersenne prime.

Marin Mersenne

Mersenne Primes

- Prime numbers of the form $2^p - 1$, where p is a prime.

  ◇ $2^2 - 1 = 3$, $2^3 - 1 = 7$, $2^5 - 1 = 31$, $2^7 - 1 = 127$ are Mersenne primes.

  ◇ $2^{11} - 1 = 2047 = 23 \cdot 89$ is not a Mersenne prime.

  ◇ The largest known prime numbers are Mersenne primes.

Marin Mersenne

Mersenne Primes

- Prime numbers of the form $2^p - 1$, where p is a prime.
  - $2^2 - 1 = 3$, $2^3 - 1 = 7$, $2^5 - 1 = 31$, $2^7 - 1 = 127$ are Mersenne primes.
  - $2^{11} - 1 = 2047 = 23 \cdot 89$ is not a Mersenne prime.

Marin Mersenne

Largest Known Prime, 49th Known Mersenne Prime Found!

January 7, 2016 — GIMPS celebrated its 20th anniversary with the discovery of the largest known prime number, $2^{74,207,281}$-1.

50th Known Mersenne Prime Found!

January 3, 2018 — Persistence pays off. Jonathan Pace, a GIMPS volunteer for over 14 years, discovered the 50th known Mersenne prime, $2^{77,232,917}$-1 on December 26, 2017. The prime number is calculated by multiplying together 77,232,917 twos, and then subtracting one. It weighs in at 23,249,425 digits, becoming the largest prime number known to mankind. It bests the previous record prime, also discovered by GIMPS, by 910,807 digits.

51st Known Mersenne Prime Found!

December 21, 2018 — The Great Internet Mersenne Prime Search (GIMPS) has discovered the largest known prime number, $\mathbf{2^{82,589,933}-1}$, having 24,862,048 digits. A computer volunteered by Patrick Laroche from Ocala, Florida made the find on December 7, 2018. The new prime number, also known as M82589933, is calculated by multiplying together 82,589,933 twos and then subtracting one. It is more than one and a half million digits larger than the previous record prime number.

http://www.mersenne.org/

Mersenne Primes

- Prime numbers of the form $2^p - 1$, where p is a prime.

  ◇ $2^2 - 1 = 3$, $2^3 - 1 = 7$, $2^5 - 1 = 31$, $2^7 - 1 = 127$ are Mersenne primes.

  ◇

  p

Marin Mersenne

Prime Found!

Januar … number, $2^{74,207,281}$-1.

January … e prime, $2^{77,232,917}$-1 on
December 2 … 23,249,425 digits, becoming
the largest

Decembe … 589,933-1, having 24,862,048
digits. A co … known as M82589933, is
calculated b … previous record prime
number.

http://www.mersenne.org/

Fermat's Library @fermatslibrary · 11小时

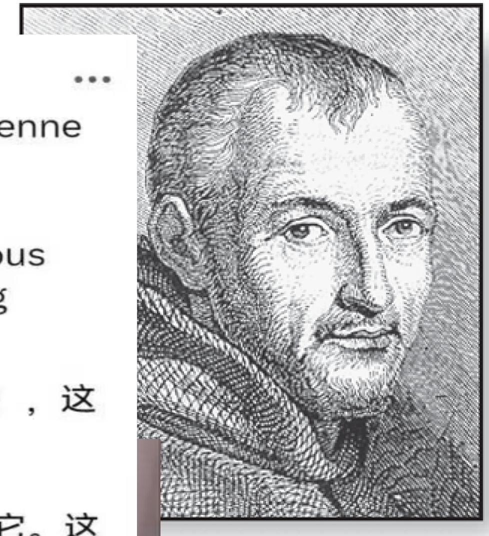
$2^{136279841}-1$, discovered today, is the largest known prime. It's a Mersenne prime (2^p-1), which are easier to find.

It took nearly 6 years for the GIMPS software to find it after the previous largest known prime. It was also the first Mersenne prime found using GPUs.

$2^{136279841}-1$，今天发现的，是已知的最大质数。它是一个梅森质数（2^p-1），这些质数更容易找到。

在找到之前已知的最大质数之后，GIMPS 软件几乎用了 6 年的时间才找到它。这也是第一个使用 GPU 找到的梅森质数。

$$2^{136279841}-1 \text{ is prime!}$$



Marin Mersenne

Jan

Prime Found!

number, $2^{74,207,281}-1$.

Janu

Decen

the lar

e prime, $2^{77,232,917}-1$ on

23,249,425 digits, becoming

Dece

589,933-1, having 24,862,048

digits.

known as M82589933, is

calcula

previous record prime

numbe

http://www.mersenne.org/

Mersenne Primes

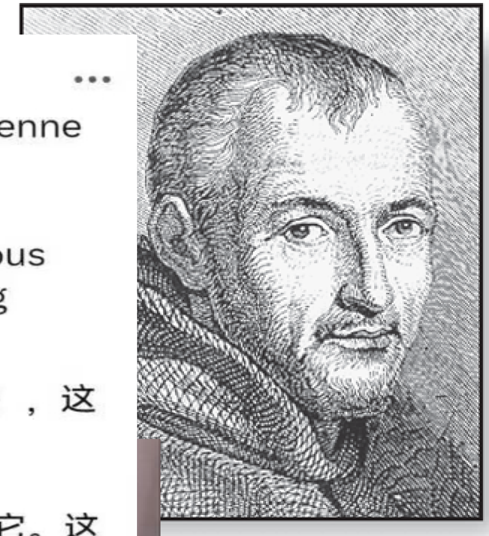
- **Fermat's Library** @fermatslibrary · 11小时

  $2^{136279841}-1$, discovered today, is the largest known prime. It's a Mersenne prime (2^p-1), which are easier to find.

  It took nearly 6 years for the GIMPS software to find it after the previous largest known prime. It was also the first Mersenne prime found using GPUs.

  $2^{136279841}-1$，今天发现的，是已知的最大质数。它是一个梅森质数（2^p-1），这些质数更容易找到。

  在找到之前已知的最大质数之后，GIMPS 软件几乎用了 6 年的时间才找到它。这也是第一个使用 GPU 找到的梅森质数。



Marin Mersenne

Prime Found!

$2^{136279841}-1$ is the New Largest Known Prime Number

October 21, 2024 — The Great Internet Mersenne Prime Search (GIMPS) has discovered a new Mersenne prime number, $2^{136279841}-1$. At 41,024,320 digits, it eclipses by more than 16 million digits the previous largest known prime number found by GIMPS nearly 6 years ago.

Janu
Decen ... e prime, $2^{77,232,917}-1$ on
the lar ... 23,249,425 digits, becoming

Dece ... $589,933-1$, having 24,862,048
digits. ... known as M82589933, is
calcula ... previous record prime
numbe

$$2^{136279841}-1 \text{ is prime!}$$

http://www.mersenne.org/

Conjectures about Primes

- *Goldbach's Conjecture* $(1+1)$: Every even integer $n > 2$, is the sum of two primes.

Conjectures about Primes

- *Goldbach's Conjecture* $(1+1)$: Every even integer $n > 2$, is the sum of two primes.

"$3+4$", "$3+3$", "$2+3$" – Y. Wang, 1956

"$1+5$" – C. Pan, 1962

"$1+4$" – Y. Wang, 1962

"$1+2$" – J. Chen, 1973

Conjectures about Primes

- *Goldbach's Conjecture* $(1+1)$: Every even integer $n > 2$, is the sum of two primes.

  "$3+4$", "$3+3$", "$2+3$" – Y. Wang, 1956

  "$1+5$" – C. Pan, 1962

  "$1+4$" – Y. Wang, 1962

  "$1+2$" – J. Chen, 1973

- *Twin-prime Conjecture*: There are infinitely many twin primes.

Linear Congruences

- A congruence of the form $ax \equiv b \pmod{m}$, where m is a positive integer, a and b are integers, and x is a variable, is called a *linear congruence*.

- A congruence of the form $ax \equiv b \pmod{m}$, where m is a positive integer, a and b are integers, and x is a variable, is called a *linear congruence*.

  The solutions to a linear congruence $ax \equiv b \pmod{m}$ are all integers x that satisfy the congruence.

Linear Congruences

- A congruence of the form $ax \equiv b \pmod{m}$, where m is a positive integer, a and b are integers, and x is a variable, is called a *linear congruence*.

  The solutions to a linear congruence $ax \equiv b \pmod{m}$ are all integers x that satisfy the congruence.

  Systems of linear congruences have been studied since ancient times.

  今有物不知其数 三三数之剩二 五五数之剩三 七七数之剩二 问物几何

  About 1500 years ago, the Chinese mathematician Sun-Tsu asked: "There are certain things whose number is unknown. When divided by 3, the remainder is 2; when divided by 5, the remainder is 3; when divided by 7, the remainder is 2. What will be the number of things?"

Modular Inverse

- An integer $\bar{a}$ such that $\bar{a}a \equiv 1 \pmod{m}$ is said to be an *inverse* of a modulo m.

Modular Inverse

- An integer $\bar{a}$ such that $\bar{a}a \equiv 1 \pmod{m}$ is said to be an *inverse* of a modulo m.

One method of solving linear congruences makes use of an inverse $\bar{a}$ if it exists. From $ax \equiv b \pmod{m}$, it follows that $\bar{a}ax \equiv \bar{a}b \pmod{m}$ and then $x \equiv \bar{a}b \pmod{m}$.

Modular Inverse

- An integer $\bar{a}$ such that $\bar{a}a \equiv 1 \pmod{m}$ is said to be an *inverse* of a modulo m.

  One method of solving linear congruences makes use of an inverse $\bar{a}$ if it exists. From $ax \equiv b \pmod{m}$, it follows that $\bar{a}ax \equiv \bar{a}b \pmod{m}$ and then $x \equiv \bar{a}b \pmod{m}$.

  When does an inverse of a modulo m exist?

Inverse of a modulo m

- **Theorem** If a and m are relatively prime integers and $m > 1$, then an inverse of a modulo m exists. Furthermore, the inverse is unique modulo m.

Inverse of a modulo m

- **Theorem** If a and m are relatively prime integers and $m > 1$, then an inverse of a modulo m exists. Furthermore, the inverse is unique modulo m.

  Proof. Since $\gcd(a, m) = 1$, there are integers s and t such that $sa + tm = 1$. Hence $sa + tm \equiv 1 \pmod{m}$. Since $tm \equiv 0 \pmod{m}$, it follows that $sa \equiv 1 \pmod{m}$. This means that s is an inverse of a modulo m.

Inverse of a modulo m

- **Theorem** If a and m are relatively prime integers and $m > 1$, then an inverse of a modulo m exists. Furthermore, the inverse is unique modulo m.

  Proof. Since $\gcd(a, m) = 1$, there are integers s and t such that $sa + tm = 1$. Hence $sa + tm \equiv 1 \pmod{m}$. Since $tm \equiv 0 \pmod{m}$, it follows that $sa \equiv 1 \pmod{m}$. This means that s is an inverse of a modulo m.

  How to prove the uniqueness of the inverse?

How to find inverses?

- Using *extended Euclidean algorithm*

How to find inverses?

- Using *extended Euclidean algorithm*

Example. Find an inverse of 101 modulo 4620.

How to find inverses?

- Using *extended Euclidean algorithm*

Example. Find an inverse of 101 modulo 4620.

$$4620 = 45 \cdot 101 + 75$$
$$101 = 1 \cdot 75 + 26$$
$$75 = 2 \cdot 26 + 23$$
$$26 = 1 \cdot 23 + 3$$
$$23 = 7 \cdot 3 + 2$$
$$3 = 1 \cdot 2 + 1$$
$$2 = 2 \cdot 1$$

How to find inverses?

- Using *extended Euclidean algorithm*

Example. Find an inverse of 101 modulo 4620.

$$
\begin{aligned}
4620 &= 45\cdot 101 + 75\\
101 &= 1\cdot 75 + 26\\
75 &= 2\cdot 26 + 23\\
26 &= 1\cdot 23 + 3\\
23 &= 7\cdot 3 + 2\\
3 &= 1\cdot 2 + 1\\
2 &= 2\cdot 1
\end{aligned}
$$

$$
\begin{aligned}
1 &= 3 - 1\cdot 2\\
1 &= 3 - 1\cdot(23 - 7\cdot 3) = -1\cdot 23 + 8\cdot 3\\
1 &= -1\cdot 23 + 8\cdot(26 - 1\cdot 23) = 8\cdot 26 - 9\cdot 23\\
1 &= 8\cdot 26 - 9\cdot(75 - 2\cdot 26) = 26\cdot 26 - 9\cdot 75\\
1 &= 26\cdot(101 - 1\cdot 75) - 9\cdot 75\\
&= 26\cdot 101 - 35\cdot 75\\
1 &= 26\cdot 101 - 35\cdot(4620 - 45\cdot 101)\\
&= -35\cdot 4620 + 1601\cdot 101
\end{aligned}
$$

Using Inverses to Solve Congruences

- Solve the congruence $ax \equiv b \pmod{m}$ by multiplying both sides by $\bar{a}$.

Using Inverses to Solve Congruences

- Solve the congruence $ax \equiv b \pmod{m}$ by multiplying both sides by $\bar{a}$.

  Example. What are the solutions of the congruence $3x \equiv 4 \pmod{7}$?

Using Inverses to Solve Congruences

- Solve the congruence $ax \equiv b \pmod{m}$ by multiplying both sides by $\bar{a}$.

  Example. What are the solutions of the congruence $3x \equiv 4 \pmod{7}$?

  Solution: We found that -2 is an inverse of 3 modulo 7. Multiply both sides of the congruence by -2, we have $x \equiv -8 \equiv 6 \pmod{7}$.

Number of Solutions to Congruences *

- **Theorem*** Let $d = \gcd(a, m)$ and $m' = m/d$. The congruence $ax \equiv b \pmod{m}$ has solutions if and only if $d|b$. If $d|b$, then there are exactly d solutions. If x_0 is a solution, then the other solutions are given by $x_0 + m', x_0 + 2m', \ldots, x_0 + (d-1)m'$.

Number of Solutions to Congruences *

- **Theorem*** Let $d = \gcd(a, m)$ and $m' = m/d$. The congruence $ax \equiv b \pmod{m}$ has solutions if and only if $d|b$. If $d|b$, then there are exactly d solutions. If x_0 is a solution, then the other solutions are given by $x_0 + m', x_0 + 2m', \ldots, x_0 + (d-1)m'$.

Proof.
1) "only if": If x_0 is a solution, then $ax_0 - b = km$. Thus, $ax_0 - km = b$. Since d divides $ax_0 - km$, we must have $d|b$.

2) "if": Suppose that $d|b$. Let $b = kd$. There exist integers s, t such that $d = as + mt$. Multiply both sides by k. Then $b = ask + mtk$. Let $x_0 = sk$. Then $ax_0 \equiv b \pmod{m}$.

3) "$\# = d$": $ax_0 \equiv b \pmod{m}$ $ax_1 \equiv b \pmod{m}$ imply that $m|a(x_1 - x_0)$ and $m'|a'(x_1 - x_0)$. This implies further that $x_1 = x_0 + km'$, where $k = 0, 1, \ldots, d-1$.

The Chinese Remainder Theorem

- About 1500 years ago, the Chinese mathematician Sun-Tsu asked:
  "There are certain things whose number is unknown. When divided by 3, the remainder is 2; when divided by 5, the remainder is 3; when divided by 7, the remainder is 2. What will be the number of things?"

今有物不知其数 三三数之剩二 五五数之剩三 七七数之剩二 问物几何

The Chinese Remainder Theorem

- About 1500 years ago, the Chinese mathematician Sun-Tsu asked:
  "There are certain things whose number is unknown. When divided by 3, the remainder is 2; when divided by 5, the remainder is 3; when divided by 7, the remainder is 2. What will be the number of things?"

今有物不知其数 三三数之剩二 五五数之剩三 七七数之剩二 问物几何

$$
\begin{aligned}
x &\equiv 2 \pmod{3} \\
x &\equiv 3 \pmod{5} \\
x &\equiv 2 \pmod{7}
\end{aligned}
$$

- **Theorem** (*The 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

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

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

The Chinese Remainder Theorem

- **Proof** Let $M_k = m/m_k$ for $k = 1, 2, \ldots, n$ and $m = m_1 m_2 \cdots m_n$. Since $\gcd(m_k, M_k) = 1$, there is an integer y_k, an inverse of M_k modulo m_k such that $M_k y_k \equiv 1 \pmod{m_k}$. Let

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots a_n M_n y_n.$$

It is checked that x is a solution to the n congruences.

The Chinese Remainder Theorem

- **Proof** Let $M_k = m/m_k$ for $k = 1, 2, \ldots, n$ and $m = m_1 m_2 \cdots m_n$. Since $\gcd(m_k, M_k) = 1$, there is an integer y_k, an inverse of M_k modulo m_k such that $M_k y_k \equiv 1 \pmod{m_k}$. Let

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots a_n M_n y_n.$$

It is checked that x is a solution to the n congruences.

How to prove the uniqueness of the solution modulo m?

The Chinese Remainder Theorem

- **Example**

$$\begin{aligned} x &\equiv 2 \pmod 3 \\ x &\equiv 3 \pmod 5 \\ x &\equiv 2 \pmod 7 \end{aligned}$$

- **Example**

$$x \equiv 2 \pmod{3}$$
$$x \equiv 3 \pmod{5}$$
$$x \equiv 2 \pmod{7}$$

Let $m = 3 \cdot 5 \cdot 7 = 105$, $M_1 = m/3 = 35$, $M_2 = m/5 = 21$, $M_3 = m/7 = 15$.

$$35 \cdot 2 \equiv 1 \pmod{3}$$
$$21 \equiv 1 \pmod{5}$$
$$15 \equiv 1 \pmod{7}$$

- **Example**

$$\begin{aligned} x &\equiv 2 \pmod 3 \\ x &\equiv 3 \pmod 5 \\ x &\equiv 2 \pmod 7 \end{aligned}$$

Let $m = 3 \cdot 5 \cdot 7 = 105$, $M_1 = m/3 = 35$, $M_2 = m/5 = 21$, $M_3 = m/7 = 15$.

$$\begin{aligned} 35 \cdot 2 &\equiv 1 \pmod 3 & \quad y_1 &= 2 \\ 21 &\equiv 1 \pmod 5 & \quad y_2 &= 1 \\ 15 &\equiv 1 \pmod 7 & \quad y_3 &= 1 \end{aligned}$$

- **Example**

$$\begin{aligned} x &\equiv 2 \pmod{3} \\ x &\equiv 3 \pmod{5} \\ x &\equiv 2 \pmod{7} \end{aligned}$$

Let $m = 3 \cdot 5 \cdot 7 = 105$, $M_1 = m/3 = 35$, $M_2 = m/5 = 21$, $M_3 = m/7 = 15$.

$$\begin{aligned} 35 \cdot 2 &\equiv 1 \pmod{3} \qquad & y_1 = 2 \\ 21 &\equiv 1 \pmod{5} \qquad & y_2 = 1 \\ 15 &\equiv 1 \pmod{7} \qquad & y_3 = 1 \end{aligned}$$

$$x = 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 \equiv 233 \equiv 23 \pmod{105}$$

- **Example**

$$
\begin{aligned}
x &\equiv 2 \pmod 3\\
x &\equiv 3 \pmod 5\\
x &\equiv 2 \pmod 7
\end{aligned}
$$

三人同行七十稀，五树梅花廿一枝，
七子团圆正月半，除百零五便得知。
-- 程大位 《算法统要》（1593年）

Let $m = 3 \cdot 5 \cdot 7 = 105$, $M_1 = m/3 = 35$, $M_2 = m/5 = 21$, $M_3 = m/7 = 15$.

$$
\begin{aligned}
35 \cdot 2 &\equiv 1 \pmod 3 \qquad & y_1 &= 2\\
21 &\equiv 1 \pmod 5 \qquad & y_2 &= 1\\
15 &\equiv 1 \pmod 7 \qquad & y_3 &= 1
\end{aligned}
$$

$$x = 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 \equiv 233 \equiv 23 \pmod{105}$$

Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

Example

$$x \equiv 2 \pmod{3}$$
$$x \equiv 3 \pmod{5}$$
$$x \equiv 2 \pmod{7}$$

Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

Example

$$\begin{aligned} x &\equiv 2 \pmod{3} \\ x &\equiv 3 \pmod{5} \\ x &\equiv 2 \pmod{7} \end{aligned}$$

$$\begin{aligned} x &\equiv 8 \pmod{15} \\ x &\equiv 2 \pmod{21} \end{aligned}$$

Modular Arithmetic in CS

- Modular arithmetic and congruencies are used in CS:
  - Pseudorandom number generators
  - Hash functions
  - Cryptography

Pseudorandom Number Generators

- *Linear congruential method*

  We choose four numbers:

  - the modulus m
  - multiplier a
  - increment c
  - seed x_0

- *Linear congruential method*

  We choose four numbers:

  - the modulus m
  - multiplier a
  - increment c
  - seed x_0

  We generate a sequence of numbers $x_1, x_2, \ldots, x_n, \ldots$ with $0 \le x_i < m$ by using the congruence

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

Pseudorandom Number Generators

- *Linear congruential method*

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

- *Linear congruential method*

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

Example:

- Assume : m=9,a=7,c=4, $x_0 = 3$

- x_1= 7*3+4 mod 9=25 mod 9 =7
- x_2 = 53 mod 9 = 8
- x_3 = 60 mod 9 = 6
- x_4= 46 mod 9 =1
- x_5 = 11 mod 9 =2
- x_6 = 18 mod 9 =0
- ....

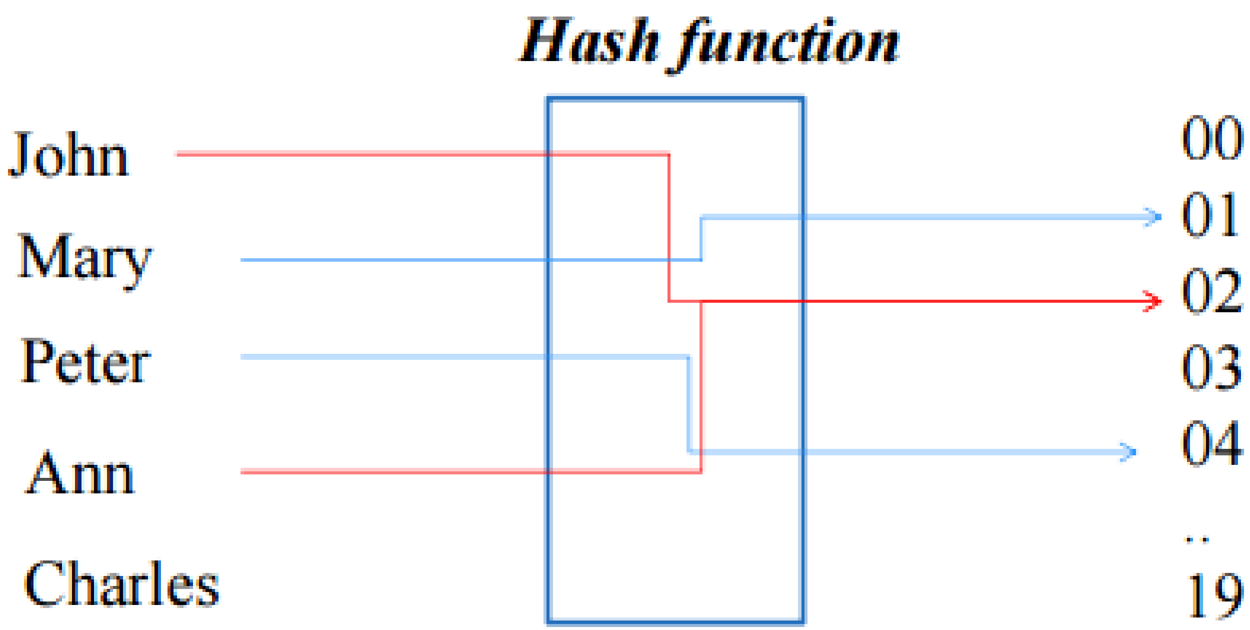
Hash Functions

- A *hash function* is an algorithm that maps data of arbitrary length to data of a fixed length. The values returned by a hash function are called *hash values* or hash codes.

Hash Functions

- A *hash function* is an algorithm that maps data of arbitrary length to data of a fixed length. The values returned by a hash function are called *hash values* or hash codes.

Example:



Hash Functions

- **Problem**: Given a large collection of records, how can we store and find a record quickly?

Hash Functions

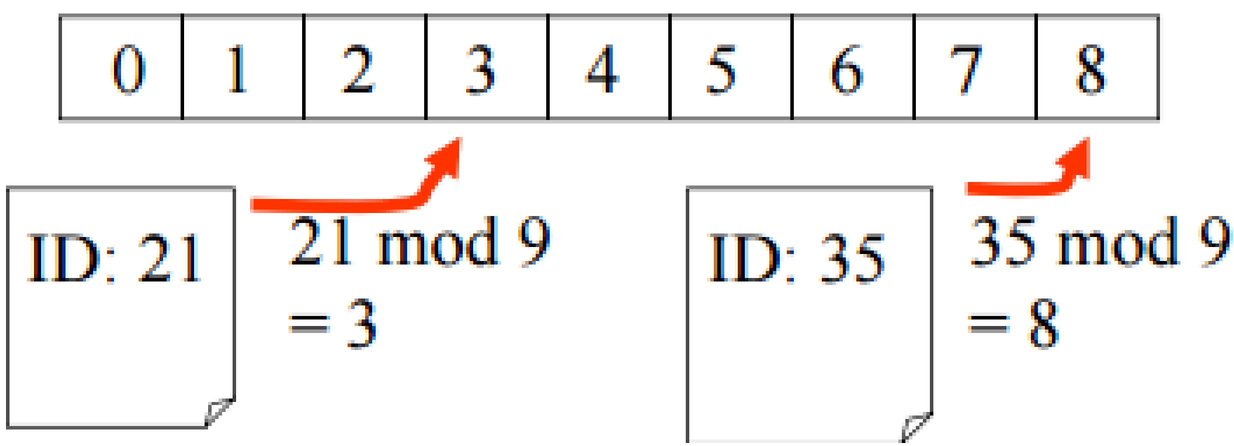
- **Problem**: Given a large collection of records, how can we store and find a record quickly?

  Solution: Use a hash function, calculate the location of the record based on the record's ID.

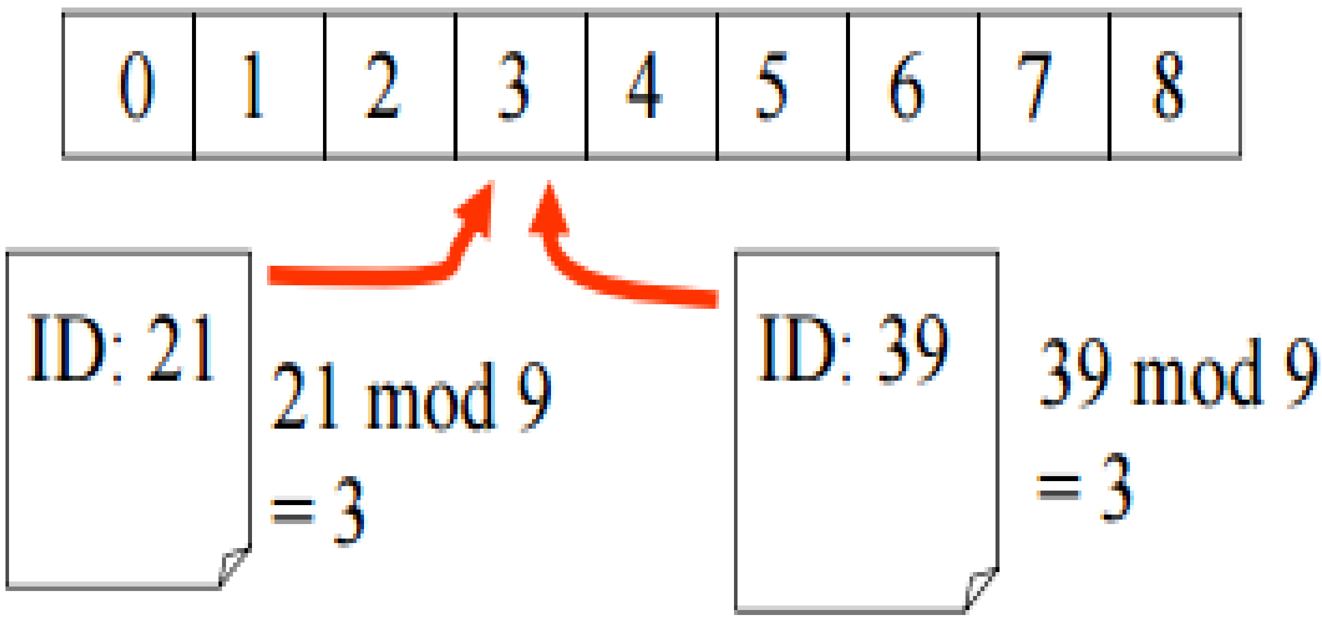
Example: A common hash function is

- $h(k) = k \textbf{ mod } n$,

where n is the number of available storage locations.



- Two records mapped to the same location

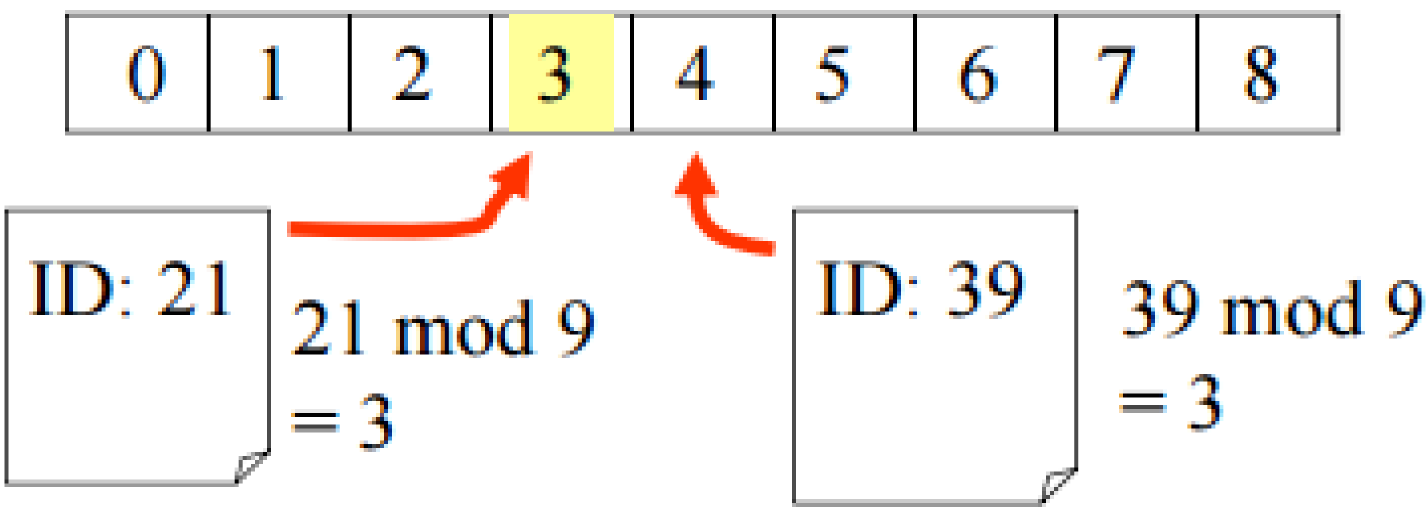


Hash Functions

- **Solution 1**: move to the next available location

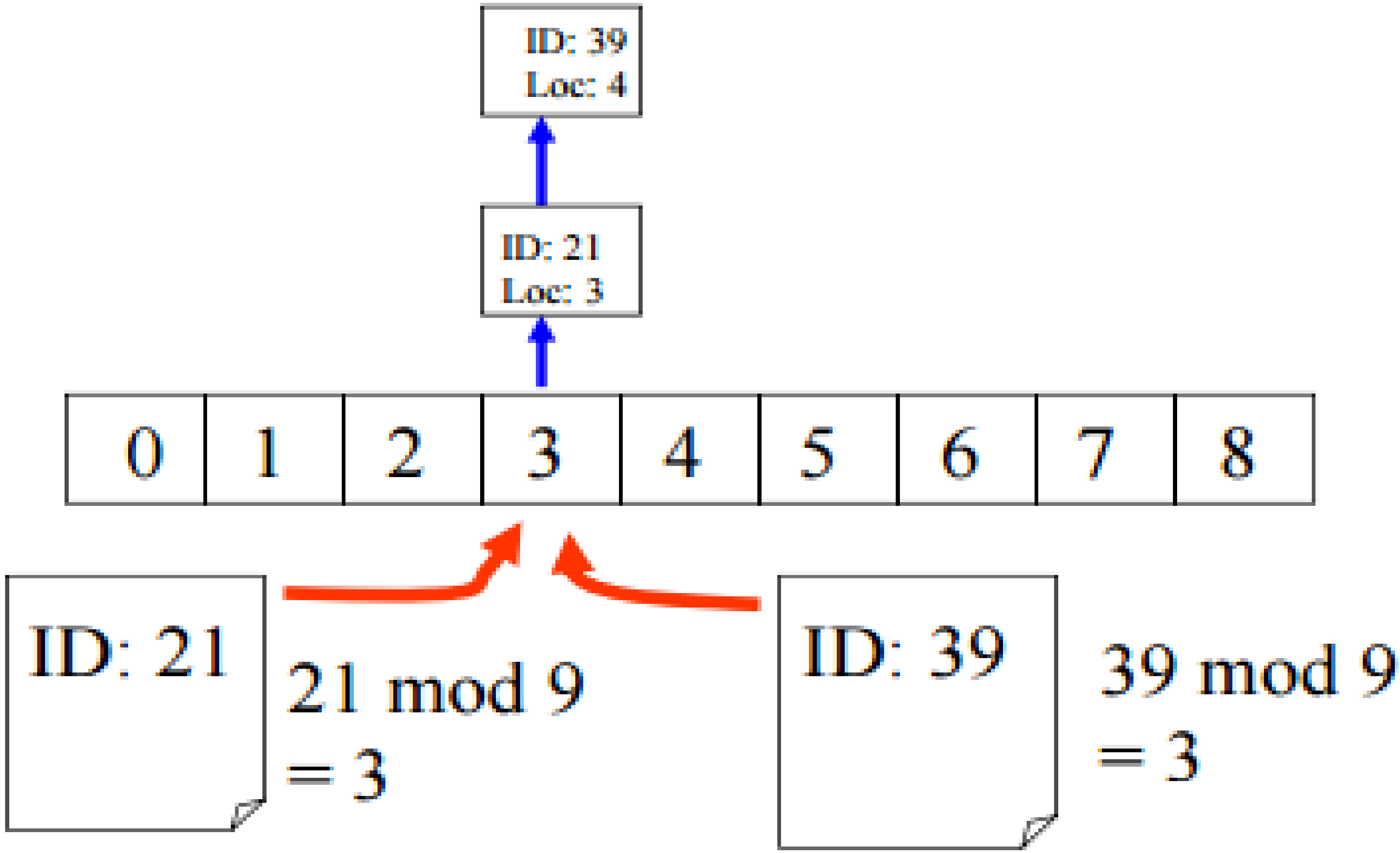
try

$$h_0(k) = k \textbf{ mod } n$$
$$h_1(k) = (k+1) \textbf{ mod } n$$
$$\ldots$$
$$h_m(k) = (k+m) \textbf{ mod } n$$



Hash Functions

- **Solution 2**: remember the exact location in a secondary structure that is searched sequentially



Next Lecture

- cryptography ...