Module #8 – Number Theory

University of Florida
Dept. of Computer & Information Science & Engineering

COT 3100
Applications of Discrete Structures
Dr. Michael P. Frank

Slides for a Course Based on the Text
Discrete Mathematics & Its Applications
(5th Edition)
by Kenneth H. Rosen

Module #8 – Number Theory

Module #8:
Basic Number Theory

Rosen 5th ed., §2.4-2.6
~31 slides, ~2 lectures
Module #8 – Number Theory

2.4: The Integers and Division

- Of course you already know what the integers are, and what division is...
- **But:** There are some specific notations, terminology, and theorems associated with these concepts which you may not know.
- These form the basics of number theory.
  - Vital in many important algorithms today (hash functions, cryptography, digital signatures).

---

Divides, Factor, Multiple

- Let $a, b \in \mathbb{Z}$ with $a \neq 0$.
- $a | b \equiv \text{“}a \text{ divides } b\text{”} :\equiv \exists c \in \mathbb{Z}: b = ac$
  - “There is an integer $c$ such that $c$ times $a$ equals $b$.”
  - Example: $3 | -12 \iff \text{True}$, but $3 | 7 \iff \text{False}$.
- Iff $a$ divides $b$, then we say $a$ is a factor or a divisor of $b$, and $b$ is a multiple of $a$.
- “$b$ is even” $\iff 2 | b$. Is 0 even? Is -4?
Facts re: the Divides Relation

- ∀a, b, c ∈ Z:
  1. a|0
  2. (a|b ∧ a|c) → a | (b + c)
  3. a|b → a|bc
  4. (a|b ∧ b|c) → a|c

- **Proof** of (2): a|b means there is an s such that b = as, and a|c means that there is a t such that c = at, so b + c = as + at = a(s + t), so a|(b + c) also. □

More Detailed Version of Proof

- Show ∀a, b, c ∈ Z: (a|b ∧ a|c) → a | (b + c).
- Let a, b, c be any integers such that a|b and a|c, and show that a | (b + c).
- By defn. of |, we know ∃s: b = as, and ∃t: c = at. Let s, t, be such integers.
- Then b + c = as + at = a(s + t), so ∃u: b + c = au, namely u = s + t. Thus a|(b + c).
Module #8 – Number Theory

Prime Numbers

- An integer \( p > 1 \) is prime iff it is not the product of any two integers greater than 1:
  \[ p > 1 \land \neg \exists a, b \in \mathbb{N}: a > 1, b > 1, \ ab = p. \]
- The only positive factors of a prime \( p \) are 1 and \( p \) itself. Some primes: 2, 3, 5, 7, 11, 13...
- Non-prime integers greater than 1 are called composite, because they can be composed by multiplying two integers greater than 1.

Review of §2.4 So Far

- \( a \mid b \iff \text{“} a \text{ divides } b \text{”} \iff \exists c \in \mathbb{Z}: b = ac \)
- \text{“} p \text{ is prime”} \iff \[ p > 1 \land \neg \exists a \in \mathbb{N}: (1 < a < p \land a \mid p) \]
- Terms factor, divisor, multiple, composite.
Module #8 – Number Theory

Fundamental Theorem of Arithmetic

Its "Prime Factorization"

- Every positive integer has a unique representation as the product of a non-decreasing series of zero or more primes.
  - $1 = (\text{product of empty series}) = 1$
  - $2 = 2$ (product of series with one element 2)
  - $4 = 2 \cdot 2$ (product of series 2, 2)
  - $2000 = 2 \cdot 2 \cdot 2 \cdot 5 \cdot 5 \cdot 5$; $2001 = 3 \cdot 23 \cdot 29$;
  - $2002 = 2 \cdot 7 \cdot 11 \cdot 13$; $2003 = 2003$

Module #8 – Number Theory

An Application of Primes

- When you visit a secure web site (https:... address, indicated by padlock icon in IE, key icon in Netscape), the browser and web site may be using a technology called RSA encryption.
- This public-key cryptography scheme involves exchanging public keys containing the product $pq$ of two random large primes $p$ and $q$ (a private key) which must be kept secret by a given party.
- So, the security of your day-to-day web transactions depends critically on the fact that all known factoring algorithms are intractable!
  - Note: There is a tractable quantum algorithm for factoring; so if we can ever build big quantum computers, RSA will be insecure.
Module #8 – Number Theory

The Division “Algorithm”

• Really just a *theorem*, not an algorithm…
  – The name is used here for historical reasons.
• For any integer *dividend* \(a\) and *divisor* \(d \not= 0\), there is a unique integer *quotient* \(q\) and *remainder* \(r \in \mathbb{N} \) \(a = dq + r\) and \(0 \leq r < |d|\).
• \(\forall a,d \in \mathbb{Z}, d > 0: \exists ! q,r \in \mathbb{Z}: 0 \leq r < |d|, a = dq + r\).  
• We can find \(q\) and \(r\) by: \(q \left\lfloor \frac{a}{d} \right\rfloor, r = a - qd\).

Module #8 – Number Theory

Greatest Common Divisor

• The *greatest common divisor* \(\gcd(a,b)\) of integers \(a,b\) (not both 0) is the largest (most positive) integer \(d\) that is a divisor both of \(a\) and of \(b\).
\[
d = \gcd(a,b) = \max(d: d|a \land d|b) \iff d|a \land d|b \land \forall e \in \mathbb{Z}. (e|a \land e|b) \land d = e
\]
• Example: \(\gcd(24,36) = ?\)
  Positive common divisors: 1,2,3,4,6,12… Greatest is 12.
### GCD shortcut

If the prime factorizations are written as

\[ a = p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n} \quad \text{and} \quad b = p_1^{b_1} p_2^{b_2} \ldots p_n^{b_n}, \]

then the GCD is given by:

\[ \gcd(a, b) = p_1^{\min(a_1, b_1)} p_2^{\min(a_2, b_2)} \ldots p_n^{\min(a_n, b_n)}. \]

**Example:**
- \( a = 84 = 2^2 \cdot 3 \cdot 7 \)
- \( b = 96 = 2^5 \cdot 3 \cdot 7^0 \)
- \( \gcd(84, 96) = 2^2 \cdot 3^1 \cdot 7^0 = 2 \cdot 2 \cdot 3 = 12. \)

### Relative Primality

Integers \( a \) and \( b \) are called *relatively prime* or *coprime* iff their \( \gcd = 1. \)

- Example: Neither 21 and 10 are prime, but they are *coprime*. 21=3\( \cdot 7 \) and 10=2\( \cdot 5 \), so they have no common factors > 1, so their \( \gcd = 1. \).

A *set* of integers \{\( a_1, a_2, \ldots \)\} is (*pairwise*) *relatively prime* if all pairs \( a_i, a_j, i\neq j \), are relatively prime.
Module #8 – Number Theory

Least Common Multiple

- lcm(a, b) of positive integers a, b, is the smallest positive integer that is a multiple both of a and of b. E.g. lcm(6, 10) = 30

\[ m = \text{lcm}(a, b) = \min(m : a \mid m \land b \mid m) \iff \]
\[ a \mid m \land b \mid m \land \forall n \in \mathbb{Z}: (a \mid n \land b \mid n) \iff (m = n) \]

- If the prime factorizations are written as

\[ a = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n} \quad \text{and} \quad b = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n}, \]

then the LCM is given by

\[ \text{lcm}(a, b) = p_1^{\max(a_1, b_1)} p_2^{\max(a_2, b_2)} \cdots p_n^{\max(a_n, b_n)}. \]

Module #8 – Number Theory

The mod operator

- An integer “division remainder” operator.
- Let \( a, d \in \mathbb{Z} \) with \( d > 1 \). Then \( a \mod d \) denotes the remainder \( r \) from the division “algorithm” with dividend \( a \) and divisor \( d \); i.e. the remainder when \( a \) is divided by \( d \). (Using e.g. long division.)
- We can compute \( (a \mod d) \) by: \( a - d \lfloor a/d \rfloor \).
- In C programming language, “\%” = mod.
Module #8 – Number Theory

**Modular Congruence**

- Let $\mathbb{Z}^+ = \{n \in \mathbb{Z} \mid n > 0\}$, the positive integers.
- Let $a, b \in \mathbb{Z}$, $m \in \mathbb{Z}^+$.
- Then $a$ is congruent to $b$ modulo $m$, written “$a \equiv b \pmod{m}$”, iff $m \mid a - b$.
- Also equivalent to: $(a - b) \mod m = 0$.
- (Note: this is a different use of “$\equiv$” than the meaning “is defined as” I’ve used before.)

---

Example shown: modulo-5 arithmetic

- Spiral Visualization of $\pmod{5}$

---

(c)2001-2003, Michael P. Frank
Module #8 – Number Theory

Useful Congruence Theorems

- Let $a,b \in \mathbb{Z}$, $m \in \mathbb{Z}^+$. Then:
  $$a \equiv b \pmod{m} \iff \exists k \in \mathbb{Z} \ a = b + km.$$  
- Let $a,b,c,d \in \mathbb{Z}$, $m \in \mathbb{Z}^+$. Then if
  $$a \equiv b \pmod{m} \text{ and } c \equiv d \pmod{m},$$  
  then:
  - $a + c \equiv b + d \pmod{m},$ and
  - $ac \equiv bd \pmod{m}$

Module #8 – Number Theory

Rosen §2.5: Integers & Algorithms

- Topics:
  - Euclidean algorithm for finding GCD’s.
  - Base-$b$ representations of integers.
    - Especially: binary, hexadecimal, octal.
    - Also: Two’s complement representation of negative numbers.
  - Algorithms for computer arithmetic:
    - Binary addition, multiplication, division.
Module #8 – Number Theory

Euclid’s Algorithm for GCD

- Finding GCDs by comparing prime factorizations can be difficult if the prime factors are unknown.
- Euclid discovered: For all integers \(a, b\),
  \[ \gcd(a, b) = \gcd((a \mod b), b). \]
- Sort \(a, b\) so that \(a > b\), and then (given \(b > 1\)) \((a \mod b) < a\), so problem is simplified.

Euclid of Alexandria
325-265 B.C.

Euclid’s Algorithm Example

- \(\gcd(372, 164) = \gcd(372 \mod 164, 164)\).
  - \(372 \mod 164 = 372 - 164 \cdot \lfloor 372/164 \rfloor = 372 - 164 \cdot 2 = 372 - 328 = 44.\)
- \(\gcd(164, 44) = \gcd(164 \mod 44, 44)\).
  - \(164 \mod 44 = 164 - 44 \cdot \lfloor 164/44 \rfloor = 164 - 44 \cdot 3 = 164 - 132 = 32.\)
- \(\gcd(44, 32) = \gcd(44 \mod 32, 32) = \gcd(12, 32) = \gcd(32 \mod 12, 12) = \gcd(8, 12) = \gcd(12 \mod 8, 8) = \gcd(4, 8) = \gcd(8 \mod 4, 4) = \gcd(0, 4) = 4.\)
Module #8 – Number Theory

Euclid’s Algorithm Pseudocode

procedure \( \text{gcd}(a, b): \text{positive integers} \)

while \( b \neq 0 \)

\[ r := a \mod b; \quad a := b; \quad b := r \]

return \( a \)

Sorting inputs not needed b/c order will be reversed each iteration.

Fast! Number of while loop iterations turns out to be \( O(\log(\max(a, b))) \).

Module #8 – Number Theory

Base-\( b \) number systems

• Ordinarily we write base-10 representations of numbers (using digits 0-9).
• 10 isn’t special; any base \( b > 1 \) will work.
• For any positive integers \( n, b \) there is a unique sequence \( a_k a_{k-1} \ldots a_1 a_0 \) of digits \( a_i < b \) such that

\[ n = \sum_{i=0}^{k} a_i b^i \]

The “base \( b \) expansion of \( n \)”

See module #12 for summation notation.
Module #8 – Number Theory

Particular Bases of Interest

- Base \( b=10 \) (decimal):
  10 digits: 0,1,2,3,4,5,6,7,8,9.
- Base \( b=2 \) (binary):
  2 digits: 0,1. (“Bits” = “binary digits.”)
- Base \( b=8 \) (octal):
  8 digits: 0,1,2,3,4,5,6,7.
- Base \( b=16 \) (hexadecimal):
  16 digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Used internally in all modern computers
Octal digits correspond to groups of 3 bits
Hex digits give groups of 4 bits

Exercise for student: Write this out in pseudocode…

Module #8 – Number Theory

Converting to Base \( b \)

(Algorithm, informally stated)
- To convert any integer \( n \) to any base \( b>1 \):
- To find the value of the rightmost (lowest-order) digit, simply compute \( n \mod b \).
- Now replace \( n \) with the quotient \( \lfloor n/b \rfloor \).
- Repeat above two steps to find subsequent digits, until \( n \) is gone (=0).
Module #8 – Number Theory

**Addition of Binary Numbers**

```plaintext
procedure add(a\_{n-1}\ldots a_0, b\_{n-1}\ldots b_0; binary representations of non-negative integers a, b)
carry := 0
for bitIndex := 0 to n - 1  \{ go through bits \}
    bitSum := a\_{bitIndex} + b\_{bitIndex} + carry  \{ 2-bit sum \}
    s\_{bitIndex} := bitSum \mod 2  \{ low bit of sum \}
    carry := \lfloor bitSum / 2 \rfloor  \{ high bit of sum \}
s\_n := carry
return s\_n\ldots s\_0: binary representation of integer s
```

Module #8 – Number Theory

**Two’s Complement**

- In binary, negative numbers can be conveniently represented using *two’s complement notation*.
- In this scheme, a string of \( n \) bits can represent any integer \( i \) such that \( -2^{n-1} < i < 2^{n-1} \).
- The bit in the highest-order bit-position (\( n-1 \)) represents a coefficient multiplying \( -2^{n-1} \);
  - The other positions \( i < n-1 \) just represent \( 2^i \), as before.
- The negation of any \( n \)-bit two’s complement number \( a = a_{n-1}\ldots a_0 \) is given by \( \overline{a_{n-1}\ldots a_0} + 1 \).

The bitwise logical complement of the \( n \)-bit string \( a_{n-1}\ldots a_0 \).
Module #8 – Number Theory

Correctness of Negation Algorithm

- **Theorem:** For an integer \( a \) represented in two’s complement notation, \( -a = \overline{a} + 1 \).
- **Proof:** \( a = -a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + a_02^0 \),
  so \( -a = a_{n-1}2^{n-1} - a_{n-2}2^{n-2} - \ldots - a_02^0 \).
  Note \( a_{n-1}2^{n-1} = (1-\overline{a}_{n-1})2^{n-1} = 2^{n-1} - \overline{a}_{n-1}2^{n-1} \).
  But \( 2^{n-1} = 2^{n-2} + \ldots + 2^0 + 1 \). So we have
  \[ -a = -\overline{a}_{n-1}2^{n-1} + (1-\overline{a}_{n-2})2^{n-2} + \ldots + (1-\overline{a}_0)2^0 + 1 = \overline{a} + 1. \]

Module #8 – Number Theory

Subtraction of Binary Numbers

**procedure** `subtract(a_{n-1} \ldots a_0, b_{n-1} \ldots b_0`;
  binary two’s complement representations of integers \( a, b \)
  
  **return** `add(a, add(\overline{b}, 1))` \{ \( a + (-b) \) \}

This fails if either of the adds causes a carry into or out of the \( n-1 \) position, since \( 2^{n-2} + 2^{n-2} \square - 2^{n-1} \), and \( -2^{n-1} + (-2^{n-1}) = -2^n \) isn’t representable!
Module #8 – Number Theory

**Multiplication of Binary Numbers**

**procedure** multiply\(a_{n-1}…a_0, b_{n-1}…b_0;\)

binary representations of \(a,b\in\mathbb{N}\)

\(\text{product} := 0\)

\(\text{for } i := 0 \text{ to } n-1\)

\(\text{if } b_i = 1 \text{ then}\)

\(\text{product} := \text{add}(a_{n-1}…a_00^i, \text{product})\)

\(\text{return } \text{product}\)

---

Module #8 – Number Theory

**Binary Division with Remainder**

**procedure** div-mod\((a,d \in \mathbb{Z}^+)\) \{Quotient & rem. of \(a/d\)\}

\(n := \max(\text{length of } a \text{ in bits, length of } d \text{ in bits})\)

\(\text{for } i := n-1 \text{ downto } 0\)

\(\text{if } a = d0^i \text{ then}\) \{Can we subtract at this position?\}

\(q_i := 1\) \{This bit of quotient is 1.\}

\(a := a - d0^i\) \{Subtract to get remainder.\}

\(\text{else}\)

\(q_i := 0\) \{This bit of quotient is 0.\}

\(r := a\)

\(\text{return } q,r\) \{\(q = \text{quotient}, r = \text{remainder}\}\)