INTRODUCTION TO DISCRETE STRUCTURES

14.2 Integer Representations and Algorithms

1.1–1.2

@2021 A.D. Gunawardena

14.2 Integer Representations & Algorithms

- Representation of Integers
- Algorithms for integer Operations
- Modular Exponentiation

Integer Representations

- Integers can be expressed using any integer greater than one as a base
- we commonly use
  - decimal (base 10)
  - binary (base 2)
  - octal (base 8)
  - hexadecimal (base 16)

Representation

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

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

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

Decimals, Hexadecimals, Binary and Octals

Decimal	0	1	2	3	4	5	6	7
Hexadecimal								
Binary								
Octal								

Decimal	8	9	10	11	12	13	14	15
Hexadecimal								
Binary								
Octal								

Base Conversions

Convert 1011 0111 to decimal, octal, and hexadecimals

From Decimals to Binary, Octal and Hexadecimals

Convert 24680 to Binary, Octal and Hexadecimals

General Algorithm for constructing Base-b expression

ALGORITHM 1 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 the base b expansion of $n\}$

INTRODUCTION TO
DISCRETE STRUCTURES

14.2 Integer Representations & Algorithms

- Representation of Integers
- Algorithms for integer Operations
- Modular Exponentiation

Adding Binary Numbers

ALGORITHM 2 Addition of Integers.

procedure *add*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1}a_{n-2} \ldots a_1a_0)_2$
  and $(b_{n-1}b_{n-2} \ldots b_1b_0)_2$, 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 $(s_ns_{n-1} \ldots s_0)_2$}

Exercise. Modify the algorithm to handle addition of octal or hexadecimal numbers

Examples

Add the binary numbers 100011 and 110011

Add the octal numbers 745 and 123

Multiplication of Numbers

ALGORITHM 3 Multiplication of Integers.

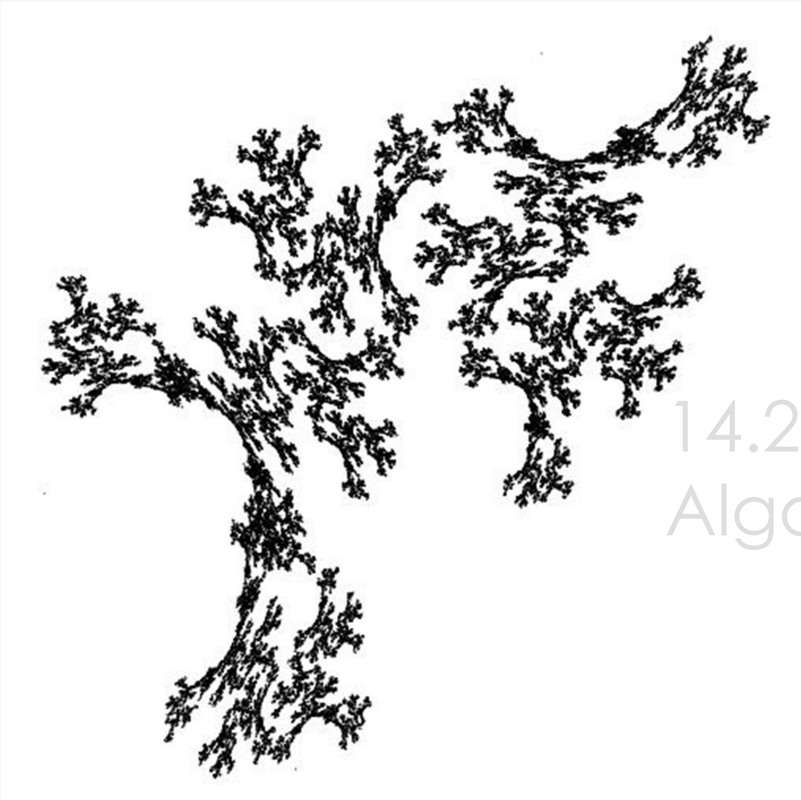
procedure *multiply*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1}a_{n-2}\ldots a_1a_0)_2$
and $(b_{n-1}b_{n-2}\ldots b_1b_0)_2$, respectively}
for $j := 0$ **to** $n-1$
    if $b_j = 1$ **then** $c_j := a$ shifted j places
    else $c_j := 0$
{$c_0, c_1, \ldots, c_{n-1}$ are the partial products}
$p := 0$
for $j := 0$ **to** $n-1$
    $p := p + c_j$
return p {p is the value of ab}

Div and Mod

ALGORITHM 4 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$\}$

Credits. Rosen Textbook



INTRODUCTION TO DISCRETE STRUCTURES

14.2 Integer Representations & Algorithms

- Representation of Integers
- Algorithms for integer Operations
- Modular Exponentiation

INTRODUCTION TO DISCRETE STRUCTURES

14.2 Integer Representations and Algorithms

1.1–1.2