2.5 INTEGERS AND ALGORITHMS

We accelerate evaluation of gcd's, of arithmetic operations, and of monomials and polynomials.

POSITIONAL REPRESENTATION of INTEGERS

Although arithmetic algorithms are much more complicated for numbers in positional notation than for numbers in monadic notation, they pay benefits in execution time.

(1) Addition algorithm execution time decreases from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$.

(2) Multiplication algorithm execution time decreases from $\mathcal{O}(nm)$ to $\mathcal{O}(\log n \log m)$.

Theorem 2.5.1. *Let $b > 1$ and $n \geq 0$ be integers. Let k be the maximum integer such that $b^k \leq n$. Then there is a unique set of nonnegative integers $a_k, a_{k-1}, \ldots, a_0 < b$ such that*

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

Proof: Apply the division algorithm to n and b to obtain a quotient and remainder a_0. Then apply the division algorithm to that quotient and b to obtain a new quotient and remainder a_1. Etc. ♢

NUMBER BASE CONVERSION

The algorithm in the proof of Theorem 2.5.1 provides a method to convert any positive integer from one base to another.

Example 2.5.1: Convert 1215_{10} to base-7.

n	d	q	r
1215	7	173	4
173	7	24	5
24	7	3	3
3	7	0	3

Solution: 3354_7

EVALUATION OF MONOMIALS

Example 2.5.2: Calculate 13^n, e.g. 13^{19}.

Usual method: $13 \times 13 \times 13 \times \cdots \times 13$
time $= \Theta(n)$.

Better method:
$13, 13^2, 13^4, 13^8, 13^{16}$ takes $\Theta(\log n)$ steps
$13 \times 13^2 \times 13^{16}$ takes $\Theta(\log n)$ steps

EVALUATION OF POLYNOMIALS

Evaluate $f(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$

Usual method of evaluation takes $\Theta(n)$:

n multiplications to calculate n powers of x
n multplications by coefficients
n additions

Horner's method (due to _____):

$a_n x + a_{n-1}$
$(a_n x + a_{n-1})x + a_{n-2}$ etc.

requires only n multiplications and n additions.

EUCLIDEAN ALGORITHM

Lemma 2.5.2. *Let $d\backslash m$ and $d\backslash n$. Then $d\backslash m-n$ and $d\backslash m+n$.*

Proof: Suppose $m = dp$ and $n = dq$. Then $m - n = d(p - q)$ and $m + n = d(p + q)$. $\diamondsuit$

Corollary 2.5.3. $\gcd(m, n) = \gcd(m - n, n)$.

Proof: In three steps.

A1. $\gcd(m, n)$ is a common div of $m - n$ and n, and $\gcd(m - n, n)$ is a common div of m and n.
Pf. Both parts by Lemma 1.

A2. $\gcd(m, n) \leq gcd(m - n, n)$
and $\gcd(m - n, n) \leq gcd(m, n)$.
Pf. Both parts by A1 and def of gcd ("greatest").
A3. $\gcd(m, n) = \gcd(m - n, n)$.
Pf. Immediate from A2. $\diamondsuit$ Cor 2.5.3

Cor 2.5.4. $\gcd(m, n) = \gcd(n, m \bmod n)$.

Proof: The number $m \bmod n$ is obtained from m by subtracting a multiple of n. Iteratively apply Cor 2.5.3. $\diamondsuit$

Algorithm 2.5.1: Euclidean Algorithm

Input: positive integers $m \geq 0, n > 0$
Output: $\gcd(n, m)$

If $m = 0$ **then return**(n)
else return $\gcd(m, n \bmod m)$

Time-Complexity: $\mathcal{O}(\log(\min(n, m)))$.
Much better than Naive GCD algorithm.

Example 2.5.3: Euclidean Algorithm

$$\gcd(210, 111) = \gcd(111, 210 \bmod 111) =$$
$$\gcd(111, 99) = \gcd(99, 111 \bmod 99) =$$
$$\gcd(99, 12) = \gcd(12, 99 \bmod 12) =$$
$$\gcd(12, 3) = \gcd(3, 12 \bmod 3) =$$
$$\gcd(3, 0) = 3$$

Example 2.5.4: Euclidean Algorithm

$$\begin{aligned}
\gcd(42,26) &= \gcd(26, 42 \bmod 26) = \\
\gcd(26,16) &= \gcd(16, 26 \bmod 16) = \\
\gcd(16,10) &= \gcd(10, 16 \bmod 10) = \\
\gcd(10,6) &= \gcd(6, 10 \bmod 6) = \\
\gcd(6,4) &= \gcd(4, 6 \bmod 4) = \\
\gcd(4,2) &= \gcd(2, 4 \bmod 2) = \\
\gcd(2,0) &= 2
\end{aligned}$$