3. The Fundamentals
3.1 Algorithms

Definition 1 An algorithm is a definite procedure for solving a problem using a finite number of steps.

Searching Algorithms
Problem: Locate an element $x$ in a list of distinct elements $a_1, a_2, \ldots, a_n$ or determine that it is not in the list.
Linear Search Algorithm

procedure linear_search(x: integer, a₁, ..., aₙ: distinct integers)
1. i := 1
2. while (i ≤ n and x ≠ aᵢ)
3. i := i + 1
4. if i ≤ n then location := i
5. else location := 0 {x is not in the list}
**Binary Search Algorithm**

procedure binary_search($x$:integer, $a_1, a_2, \ldots, a_n$ integers in increasing order)

1. $i := 1$
2. $j := n$
3. while $i < j$
4. begin
5. \[ m := \lfloor (i + j)/2 \rfloor \] \{middle element\}
6. if $x > a_m$ then $i := m + 1$
7. else $j := m$
8. end
9. if $x = a_i$ then $location := i$
10. else $location := 0$
**Sorting Algorithms**

**Problem:** Put a list $a_1, a_2, \ldots, a_n$ in the increasing order.

**The Bubble Sort**

procedure bubblesort($a_1, \ldots, a_n$)
1. for $i := 1$ to $n - 1$
2. for $j := 1$ to $n - i$
3. if $a_j > a_{j+1}$ then interchange $a_j$ with $a_{j+1}$
The Insertion Sort

procedure insertionsort($a_1, \ldots, a_n$)
1. for $j := 2$ to $n$
2. begin
3. \hspace{1em} $i := 1$
4. \hspace{1em} while $a_j > a_i$
5. \hspace{2em} $i := i + 1$
6. \hspace{1em} $m := a_j$
7. \hspace{1em} for $k := 0$ to $j - i - 1$
8. \hspace{2em} $a_{j-k} := a_{j-k-1}$
9. \hspace{1em} $a_i := m$
10. end
Greedy Algorithms
Greedy algorithm makes the "best" decision for the current step when solving an optimization problem.

3.2 The Growth of Functions

Definition 2 Let $f$ and $g$ be two functions from the set of integer (or real) numbers. We say that $f(x)$ is $O(g(x))$ (which we write as $f(x) = O(g(x))$) if there are two constants $C$ and $k$ such that

$$|f(x)| \leq C|g(x)|,$$

whenever $x > k$. 
**Theorem 1** Let \( f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 \) where \( a_0, \ldots, a_n \) are real numbers. Then \( f(x) = O(x^n) \).

**Fact 2** 1. Suppose that \( f_1(x) = O(g(x)) \) and \( f_2(x) = O(g(x)) \). Then \( (f_1 + f_2)(x) = O(g(x)) \).

2. Suppose that \( f_1(x) = O(g_1(x)) \) and \( f_2(x) = O(g_2(x)) \). Then \( (f_1 f_2)(x) = O(g_1(x)g_2(x)) \).
Big Omega notation:

**Definition 3** We say that \( f(x) = \Omega(g(x)) \) if there exist positive constants \( C \) and \( k \) such that

\[
|f(x)| \geq C|g(x)|,
\]

for all \( x > k \).

Big Theta notation:

**Definition 4** We say that \( f(x) = \Theta(g(x)) \) if \( f(x) = O(g(x)) \) and \( f(x) = \Omega(g(x)) \).
3.3 Complexity

1. Time used to solve a problem (time complexity).

2. Memory space used (space complexity).

Time analysis:

1. Worst-case analysis.

2. Average-case analysis.
3.4 The Integers and Division

**Definition 5** Let $a$ and $b$ be integers with $a \neq 0$. We say that $a$ divides $b$ if there is an integer $c$ such that $b = ac$. Then $a$ is called a factor of $b$ and $b$ is called a multiple of $a$.

**Fact 3** Let $a, b, c$ be integers. Then

- If $a \mid b$ and $a \mid c$ then $a \mid (b + c)$.

- If $a \mid b$ then $a \mid (bc)$ for any integer $c$.

- If $a \mid b$ and $b \mid c$ then $a \mid c$. 

Modular Arithmetic

**Definition 6** Let $a$ be an integer and $m$ a positive integer. We denote by $a \mod m$ the remainder when $a$ is divided by $m$.

**Definition 7** If $a$ and $b$ are integers and $m$ is a positive integer then $a$ is congruent to $b$ modulo $m$ if $m \mid (a - b)$.

**Fact 4** Let $m$ be a positive integer and let $a$ and $b$ be two integers.

1. $a \equiv b \pmod{m}$ if and only if $a \mod m = b \mod m$. 
2. $a \equiv b \pmod{m}$ if and only if there is an integer $k$ such that $a = b + mk$
3.5 Prime numbers and greatest common divisor

**Prime Numbers**

**Definition 8** A positive integer $p$ greater than one is called prime if the only positive factors of $p$ are 1 and $p$. A positive integer greater than one that is not prime is called composite.

**Theorem 5 (The Fundamental of Arithmetic)** Every positive integer can be written uniquely as the product of primes.

**Example 1** Prove that if $n$ is a composite integer, then $n$ has a prime divisor less than or equal to $\sqrt{n}$. 
The Division Algorithm

**Theorem 6** Let $a$ be an integer and let $d$ be a positive integer. Then there exist unique integers $q$ and $r$ such that

1. $a = dq + r$ and

2. $0 \leq r < d$. 
Greatest Common Divisor and Least Common Multiple

**Definition 9** Let $a$, $b$ be two integers not both zero. The largest integer $d$ such that $d \mid a$ and $d \mid b$ is called the greatest common divisor of $a$ and $b$. It is denoted by $gcd(a, b)$.

**Definition 10** The least common multiple of the positive integers $a$ and $b$ is the smallest positive integer that is divisible by both $a$ and $b$. It is denoted by $lcm(a, b)$.

**Theorem 7** Let $a$, $b$ be positive integers. Then

$$a \cdot b = gcd(a, b)lcm(a, b).$$
**Fact 8** Let $m$ be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then

1. $(a + c) \equiv (b + d) \pmod{m}$

2. $(ac) \equiv (bd) \pmod{m}$. 
3.6 Integers and Algorithms

Euclidean Algorithm

**Goal:** Find \( gcd(a, b), \ a \geq b. \)

**Method:** Let \( r_0 = a, r_1 = b. \) Then
\[
\begin{align*}
    r_0 &= r_1q_1 + r_2 \text{ where } 0 < r_2 < r_1 \\
    r_1 &= r_2q_2 + r_3 \text{ where } 0 < r_3 < r_2 \\
    & \cdots \\
    r_{n-2} &= r_1q_{n-1} + r_n \text{ where } 0 < r_n < r_{n-1} \\
    r_{n-1} &= r_nq_n
\end{align*}
\]
Euclidean Algorithm

procedure gcd$(a, b$: positive integers$)$
1. $x := a$
2. $y := b$
3. while$(y \neq 0)$
4. begin
5. $r := x \mod y$
6. $x := y$
7. $y := r$
8. end
9. return $gcd(a, b) = x$

Lemma 9 Let $a = bq + r$ where $a, b, q$ and $r$ are integers. Then $gcd(a, b) = gcd(b, r)$
Claim 10  \( \gcd(a, b) = r_n \)

Proof
Indeed, by Lemma 9,

\[
\gcd(a, b) = \gcd(r_0, r_1) = \gcd(r_1, r_2) = \ldots
\]

\[
\ldots = \gcd(r_{n-1}, r_n) = \gcd(r_n, 0) = r_n.
\]
Representation of Integers

- \( b = 2 \) - binary expansion

- \( b = 8 \) - octal expansion

- \( b = 16 \) - hexadecimal expansion. In this case \( a = 10, b = 11, c = 12, d = 13, e = 14, \) and \( f = 15. \)

**Theorem 11** Let \( b \) be a positive integer greater than one. 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 that are less than \( b \).
Algorithms for integer operations

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

- Addition of $a$ and $b$:

  procedure add ($a, b$: positive integers)
  1. $c := 0$
  2. for $j := 0$ to $n - 1$
  3. begin
  4. $d := \lfloor (a_j + b_j + c_j)/2 \rfloor$
  5. $s_j := a_j + b_j + c_j - 2d$
  6. $c := d$
  7. end
  8. $s_n := c$
  
  Number of additions $= O(n)$.
• Multiplication.

procedure multiply\(a, b: \text{ positive integers}\)
1. for \(j := 0\) to \(n - 1\)
2. begin
3. if \(b_j = 1\) then \(c_j := a\) shifted \(j\) places
4. else \(c_j := 0\)
5. end
6. \(p := 0\)
7. for \(j := 0\) to \(n - 1\)
8. \(p := p + c_j\)

Number of arithmetic operations = \(O(n^2)\)

**Question:** Can you do better than \(n^2\)?