An algorithm is a finite set of instructions for performing a computation or solving a problem. It follows a sequence of steps that lead to the desired answer.

**Example**: Finding the maximum element in a finite sequence.

```plaintext
procedure max (a_1, a_2, ..., a_n: integers)
    max := a_1;
    for i := 2 to n
        if max < a_i then max := a_i;
```

**Properties of an algorithm**

- **Input**
- **Output**
- **Definiteness**: Steps are precise.
- **Correctness**: Produces correct answer
- **Finiteness**: Terminates within a finite number of steps
- **Effectiveness**: Fast and memory-effective
- **Generality**: Useful for all problems of the same nature.
Before writing a program for a given problem, one has to design an efficient algorithm for the problem, and then write the program for the problem.

For some problems, efficient algorithms exist, but for some problems no efficient algorithms have been found.

Some problems are unsolvable. I.e. no algorithm exists for them.
The Halting Problem

Is there a computer program that takes as input a computer program P and input I to P and determines whether or not P will stop when running with I?

Answer: No such a program exists, i.e. the halting problem is unsolvable.

Informal Proof

By contradiction.

Suppose there is a solution to the halting problem, a program \( H(P,I) \). It generates "Yes" if \( P \) stops with input \( I \), and "No" otherwise.

We construct a program \( K(P) \) as follows:
Now suppose we provide $K$ as input to program $K$.

If $H(K, K)$ produces "yes", then $K(K)$ loops forever.

If $H(K, K)$ produces "no", then $K(K)$ produces "yes".

In both cases we have a contradiction.
The growth of Functions

We can estimate the growth of a function without worrying about constant multipliers or smaller order terms.

Big-O Notation

Let \( f \) and \( g \) be functions from the set of integers or the set of real numbers to the set of real numbers. We say that \( f(x) \) is \( O(g(x)) \) if there exist constants \( C \) and \( k \) such that

\[
|f(x)| \leq C |g(x)|
\]

where \( x > k \).

Example

\( f(x) = x^2 + 2x + 1 \) is \( O(x^2) \)

Proof

\[
x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2
\]

whenever \( x > 1 \). We choose \( C = 4 \), \( k = 1 \).
FIGURE 1 The Function $x^2 + 2x + 1$ is $O(x^2)$.

Remark: $x^2$ is $O(x^2 + 2x + 1)$, because $x^2 < x^2 + 2x + 1$ for $x > 1$. Then, $C = 1$ and $k = 1$.

$x^2$ is $O(x^3)$, but $x^3$ is not $O(x^2)$.

"$f(x)$ is $O(g(x))$" is sometimes written $f(x) = O(g(x))$ or $f(x) \in O(g(x))$.

We say that $f(x)$ and $g(x)$ are of the same order if $f(x) = O(g(x))$ and $g(x) = O(f(x))$; i.e. they have the same growing rate.
The part of the graph of $f(x)$ that satisfies $f(x) < Cg(x)$ is shown in color.

**Figure 2.** The function $f(x)$ is $O(g(x))$.

If $|f(x)| \leq C|g(x)|$ for $x > k$, and $|g(x)| \leq C|h(x)|$ for $x > k$, then $f(x)$ is $O(h(x))$.

**Example.** Show $n^2$ is not $O(n)$.

In order for $n^2$ to be $O(n)$, $n^2 \leq Cn$ for $n > k$, i.e., $n \leq C$ for some $k$ where $n > k$.

But $n \leq C$ cannot hold for all $n$ with $n > k$ no matter what $C$ and $k$ are.

**Some important Big-O results.**

1. Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$, where $a_i$'s are real numbers. Then $f(x) = O(x^n)$ (see proof in the textbook)
2. \[ \sum_{i=1}^{n} i \leq \sum_{i=1}^{n} n = n^2. \text{ Therefore, } \sum_{i=1}^{n} i = O(n^2) \]

3. \[ n! = 1 \cdot 2 \cdot 3 \cdots n < \underbrace{n \cdot n \cdot n \cdots n}_{n} = n^n \]
   \[ \text{Therefore, } n! = O(n^n) \]

4. \[ \log n! \leq \log n^n = n \log n. \text{ Therefore} \]
   \[ \log n! = O(n \log n) \]
   \[ \text{(note: the base of } \log \text{ is 2)} \]

5. Let the base be a constant \( b \neq 2 \).
   \[ \text{Then, } \log_b n = \frac{\log n}{\log b} = O(\log n) \]
   \[ (C = \frac{1}{\log b}, \ k = 1) \]

**FIGURE 3** A Display of the Growth of Functions Commonly Used in Big-O Estimates.
**Facts**

1. If \( f_1(x) = O(g_1(x)), f_2(x) = O(g_2(x)) \),
then \( f_1(x) + f_2(x) = \max \{ g_1(x), g_2(x) \} \)

2. If \( f_1(x) = O(g_1(x)), f_2(x) = O(g_2(x)) \),
then \( f_1(x) \cdot f_2(x) = O(g_1(x) \cdot g_2(x)) \)

For proofs of 1 and 2, see the textbook.

**Example:** \( f(n) = \underbrace{f_1(n)}_{\log(n)} + \underbrace{f_2(n)}_{(n^2 + 3) \log(n)} = O(n^2 \log n) \)

\( \log(n) = O(n \log n) \).

\( 3 \log(n!) = O(n^2 \log n) \).

\( (n^2 + 3) \log n = O(n^2 \log n) \).

\[ f(n) = f_1(n) + f_2(n) = \max \{ n^2 \log n, n^2 \log n^3 \} = O(n^2 \log n) \]

**Example:** \( f(x) = \underbrace{f_1(x)}_{(x+1) \log(x^2+1)} + \underbrace{f_2(x)}_{3x^2} = O(x^2) \).

\( \log(x^2+1) \leq \log(2x^2) = 3 \log x \).

\( f_1(x) \leq 2x \log(x^2+1) \leq 6x \log x = O(x \log x) \).

\( f_2(x) = 3x^2 = O(x^2) \).

\( f_1(x) + f_2(x) = O(x^2) \).
Let $f$ and $g$ be the functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $Ω(g(x))$ (denoted by $f(x) = Ω(g(x))$ or $f(x) ∈ Ω(g(x))$) if there are positive constants $C$ and $k$ s.t. $|f(x)| ≥ C|g(x)|$ whenever $x > k$.

We say that $f(x)$ is $Θ(g(x))$ (denoted by $f(x) = Θ(g(x))$ or $f(x) ∈ Θ(g(x))$) if $f(x) = O(g(x))$ and $f(x) = Ω(x)$, and say that $f(x)$ is of order $g(x)$.

Fact: Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$, where $a_i$s are real numbers with $a_n \neq 0$.

Then $f(x) = Θ(x^n)$, i.e. $f(x)$ is of order $x^n$.

Proof: We already knew $f(x) = O(x^n)$. We only need to show $f(x) = Ω(x^n)$.
Let \( m = \max \{ a_i \mid 0 \leq a_i < n^2 \} \), and

\[ k = \max \{ 1, \frac{2nm}{|a_n|} \}. \]

Then, \( \frac{a_{n-i}}{x^i} < \frac{|a_n|}{2n} \) for \( x > k \). Now we have for all \( x > k \)

\[
|f(x)| = |a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0|
\]

\[
= x^n \left| a_n + \frac{a_{n-1}}{x} + \ldots + \frac{a_1}{x^{n-1}} + \frac{a_0}{x^n} \right|
\]

\[
\geq x^n \left| \frac{a_n}{2} \right|
\]

Therefore, \( f(x) = \Omega(x^n) \) with

\[
c = \left| \frac{a_n}{2} \right| \quad \text{and} \quad k = \max \{ 1, \frac{2nm}{|a_n|} \}. \]
Integer division

$a, d, q, r$: integers.

Consider $\frac{a}{d}$.

$a$: dividend
$d$: divisor

Quotient: \( q = a \div d \)
Remainder: \( r = a \mod d \)

**Example**

\( a = 101, \ d = 11 \)

\[
101 = 11 \cdot 9 + 2
\]

\( q = 9 \)

\( r = 2 \)

\( a = -11, \ d = 3 \)

\[
-11 = 3 \cdot (-4) + 1
\]

\( q = -4 \)

\( r = 1 \)

Let \( a \) and \( b \) be integers, and \( m \) be a positive integer. Then \( a \) is congruent to \( b \) modulo \( m \) if \( m \) divides \( a - b \). This is denoted by \( a \equiv b \pmod{m} \).

We write \( a \not\equiv b \pmod{m} \) if \( a \) and \( b \) are not congruent modulo \( m \).
Fact: \( a \equiv b \pmod{m} \) iff \( a \mod m = b \mod m \).

Example: \( 17 \equiv 5 \pmod{6} \)
\[
\begin{array}{c}
\uparrow \\
17 \mod 6 = 5 = 5 \mod 6
\end{array}
\]

Fact: \( a \equiv b \pmod{m} \) iff there exists an integer \( k \) s.t. \( a = b + km \).

Fact: If \( a \equiv b \pmod{m} \) and \( c \equiv d \pmod{m} \), then \( a+c \equiv b+d \pmod{m} \), and \( ac \equiv bd \pmod{m} \).

Fact: \((a+b) \mod m = ((a \mod m)+(b \mod m)) \mod m\) and \((ab) \mod m = ((a \mod m)(b \mod m)) \mod m\).

Example: \( 7 \equiv 2 \pmod{5} \), \( 11 \equiv 1 \pmod{5} \)
\[
\begin{array}{c}
\uparrow \\
a \\
\uparrow \\
b \\
\uparrow \\
c \\
\uparrow \\
d
\end{array}
\]
\[
\begin{align*}
7+11 &= 18 \equiv 2+1 = 3 \pmod{5} \\
7 \cdot 11 &= 77 \equiv 2 \cdot 1 = 2 \pmod{5} \\
(7+2) \mod 5 &= 4 = (7 \mod 5 + 2 \mod 5) \mod 5 \\
&= 4 \mod 5 \\
(7 \cdot 2) \mod 5 &= 4 = ((7 \mod 5) \cdot (2 \mod 5)) \mod 5 \\
&= 4 \mod 5.
\end{align*}
\]
Applications of Congruences

1. Hashing Table

Used for fast information retrieval. Records are associated with keys. Social Security numbers can be used as keys.

Simple hashing function:
\[ h(k) = k \mod m \]

\( k \): a key  
\( m \): number of available table entries.

\( h(k) \): entry index.

Multiple keys can be mapped to the same table entry, causing collision.

Each entry can be associated with a list of records with the same hashing value.
Random numbers are needed for computer simulations. Computer generated random numbers are called pseudorandom numbers, because they are not truly random.

A simple pseudorandom number generator:

4 integers

\[ m \]
\[ a : \ 2 \leq a < m \]
\[ c : \ 0 \leq c < m \]
\[ x_0 : \ 0 \leq x_0 < m \]

A sequence of numbers \( x_1, x_2, \ldots, x_n, \ldots \) is generated by

\[ x_1 = (a \times x_0 + c) \mod m \]
\[ x_{n+1} = (a \times x_n + c) \mod m \]

For pseudorandom number between 0 and 1, we can use

\[ \frac{x_0}{m}, \frac{x_1}{m}, \frac{x_2}{m}, \ldots, \frac{x_n}{m}, \ldots \]
For instance, the sequence of pseudorandom numbers generated by choosing \( m = 9, \ a = 7, \ c = 4, \) and \( x_0 = 3, \) can be found as follows:

\[
\begin{align*}
    x_1 &= 7x_0 + 4 \mod 9 = 7 \cdot 3 + 4 \mod 9 = 25 \mod 9 = 7 \\
    x_2 &= 7x_1 + 4 \mod 9 = 7 \cdot 7 + 4 \mod 9 = 53 \mod 9 = 8 \\
    x_3 &= 7x_2 + 4 \mod 9 = 7 \cdot 8 + 4 \mod 9 = 60 \mod 9 = 6 \\
    x_4 &= 7x_3 + 4 \mod 9 = 7 \cdot 6 + 4 \mod 9 = 46 \mod 9 = 1 \\
    x_5 &= 7x_4 + 4 \mod 9 = 7 \cdot 1 + 4 \mod 9 = 11 \mod 9 = 2 \\
    x_6 &= 7x_5 + 4 \mod 9 = 7 \cdot 2 + 4 \mod 9 = 18 \mod 9 = 0 \\
    x_7 &= 7x_6 + 4 \mod 9 = 7 \cdot 0 + 4 \mod 9 = 4 \mod 9 = 4 \\
    x_8 &= 7x_7 + 4 \mod 9 = 7 \cdot 4 + 4 \mod 9 = 32 \mod 9 = 5 \\
    x_9 &= 7x_8 + 4 \mod 9 = 7 \cdot 5 + 4 \mod 9 = 39 \mod 9 = 3
\end{align*}
\]

Because \( x_9 = x_0 \) and because each term depends only on the previous term, this sequence is generated:

\[3, 7, 8, 6, 1, 2, 0, 4, 5, 3, 7, 8, 6, 1, 2, 0, 4, 5, 3, \ldots.\]

Let \( m = 2^{31} - 1, \ a = 7^5 = 16807, \ c = 0.\)

\[x_{n+1} = 16807 \cdot x_n \mod (2^{31} - 1)\]

can be used to generate \( 2^{31} - 2 \) numbers before repetition begins.

\[\text{(3) Cryptology (secret messages)}\]

Caesar method: a simple way.

Encryption: coding

\[0 \ (A), \ 1 \ (B), \ 2 \ (C), \ \ldots, \ 25 \ (Z)\]

Encryption function:

\[
f(p) = (p + 3) \mod 26 = c
\]

Decryption: decoding

\[
f^{-1}(c) = (c - 3) \mod 26 = p
\]
Example

Message:

"MEET YOU IN THE PARK"

12.4.4.19 24.14.20 8.13 19.7.4 15.0.17.10

\[ f(p) = (p+3) \mod 26 = c \] message sent

15.7.7.22 1.17.23 11.16 22.10.7 18.3.20.13

\[ f(p) = (c-3) \mod 26 = p \] message recovered

12.4.4.19 24.14.20 8.13 19.7.4 15.0.17.10

A better way

\[ f(p) = (2p+b) \mod 26 = c \]

\[ f^{-1}(c) = (2c-b) \mod 26 = p \]

More complicated ways have been proposed and used.
A positive integer $p$ greater than 1 is called a prime number if the only positive factors of $p$ are 1 and $p$. A positive integer that is greater than 1 and is not prime is called composite.

The fundamental Theorem of Arithmetic:

Every positive integer greater than 1 can be written uniquely as a prime or as the product of 2 or more primes where the prime factors are written in order of nondecreasing value.

**Example:** $999 = 3 \cdot 3 \cdot 3 \cdot 37 = 3^3 \cdot 37$

**Theorem**

There are infinite many primes

**Proof** By contradiction. Assume there are finite primes $p_1, p_2, \ldots, p_n$. Consider

$Q = p_1 p_2 \cdots p_n + 1$.

The none of primes $p_i$ divides $Q$ since otherwise
$p_j$ divides $Q - p_1p_2 \cdots p_n = 1$. Hence, there is a prime not in the list $p_1, p_2, \ldots, p_n$. This prime is either $Q$, if it is prime, or a prime factor of $Q$. This is a contradiction because we assumed that we have listed all primes.

**Notation**: $a \mid b$ (a divides $b$, i.e. there is $c$ s.t. $b = ac$) \(a, b, c\) are integers, $a > 0$.

Let $a$ and $b$ be integers, not both 0. The largest integer $d$ s.t. $d \mid a$ and $d \mid b$ is called the greatest common divisor of $a$ and $b$ (denoted by $\gcd(a, b)$).

Given positive integers $a$ and $b$, the least common multiple of $a$ and $b$ is the smallest positive integer $m$ s.t. $a \mid m$ and $b \mid m$ (denoted by $\operatorname{lcm}(a, b)$).
Example

\[120 = 2^3 \cdot 3 \cdot 5\]
\[500 = 2^2 \cdot 5^3\]

\[
\gcd(120, 500) = 2^{\min\{3, 2\}} \cdot 3^{\min\{1, 0\}} \cdot 5^{\min\{1, 3\}}
\]
\[= 2^2 \cdot 3^0 \cdot 5^1 = 20\]

\[
\text{lcm}(120, 500) = 2^{\max\{3, 2\}} \cdot 3^{\max\{1, 0\}} \cdot 5^{\max\{1, 3\}}
\]
\[= 2^3 \cdot 3^1 \cdot 5^3 = 3000\]

Theorem

Let \(a\) and \(b\) be positive integers. Then

\[ab = \gcd(a, b) \cdot \text{lcm}(a, b)\]

Example

\(a = 120, \ b = 500\)

\[
\gcd(120, 500) = 20
\]
\[
\text{lcm}(120, 500) = 3000
\]

\[ab = 120 \times 500 = 20 \times 3000 = 60000\]
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.

An integer \( \overline{a} \) is an inverse of a modulo \( m \) if
\[ \overline{a}a \equiv 1 \pmod{m}. \]

Integers \( a \) and \( b \) are relatively prime if
\[ \gcd(a, b) = 1. \]

**Theorem** If \( a \) and \( m \) are relatively prime integers and \( m > 1 \), then an inverse of a modulo \( m \) exists. Furthermore, this inverse is unique modulo \( m \). (I.e. there is a unique positive integer \( \overline{a} \) less than \( m \) that is an inverse of a modulo \( m \).)
Example

Find an inverse of 3 modulo 7

Solution:

3 and 7 are relative prime to each other (i.e. \( \gcd(3, 7) = 1 \)). By the above theorem such an inverse exists.

Since \( 7 = 2 \cdot 3 + 1 \), we have

\[-2 \cdot 3 + 1 \cdot 7 = 1.\]

Then \(-2\) is an inverse of 3 modulo 7, i.e.

\[-2 \cdot 3 = 1 \mod 7.\]

Every integer \( i \) s.t. \( i = -2 \mod 7 \) is an inverse of 3 modulo 7, such as 5, -9, 12, ...
The Chinese Remainder Theorem

Let \( m_1, m_2, \ldots, m_n \) be positive pairwise relative primes and \( a_1, a_2, \ldots \), an arbitrary integers. Then the system

\[
\begin{align*}
x &\equiv a_1 \pmod{m_1} \\
x &\equiv a_2 \pmod{m_2} \\
& \vdots \\
x &\equiv a_n \pmod{m_n}
\end{align*}
\]

has a unique solution modulo \( m_1 m_2 \cdots m_n \equal m \) (That is, there is a solution \( x \) with \( 0 \leq x < m \) and all other solutions are congruent modulo \( m \) to this solution)

Fermat's Little Theorem

If \( p \) is prime and \( a \) is an integer not divisible by \( p \), then

\[
a^{p-1} \equiv 1 \pmod{p}
\]

Furthermore, for every integer \( a \)

\[
a^p \equiv a \pmod{p}
\]
RSA Public-key Cryptosystem

Secure communication between "Alice" (A) and "Bob" (B).

Pa : public key for Alice
Sa : secret (private) key for Alice
Pb : public key for Bob
Sb : secret key for Bob

Consider the case Bob wishes to send Alice a message M:

1. Bob obtains Alice's Pa (from public directory)
2. Bob computes ciphertext $C = P_a(M)$, and sends $C$ to Alice
3. When Alice receives $C$, she applies her secret key $S_a$ to retrieve $M = S_a(C)$
Alice's public key is \((e, n)\)

Alice's secret key is \((d, n)\)

\[ P_a(M) \equiv M^e \pmod{n} = C \]

\[ S_a(C) \equiv C^d \pmod{n} = M \]

where \(n = p \cdot q\), \(p\) and \(q\) are large primes;

\(e\) is relatively prime to \((p-1)(q-1)\), and

\(d\) is an inverse of \(e\) modulo \((p-1)(q-1)\).

**Correctness:**

\[ S_a(P_a(M)) = M^{ed} \pmod{n} \]

\[ ed = 1 + k(p-1)(q-1) \text{ for some } k. \]

By Fermat's Theorem,

\[ M^{ed} \equiv M(M^{p-1})^{k(q-1)} \pmod{p} \]

\[ \equiv M(1)^{k(q-1)} \pmod{q} \]

and

\[ M^{ed} \equiv M \pmod{pq} \]

By the Chinese Remainder theorem

\[ M^{ed} \equiv M \pmod{n} \]
The RSA Cryptosystem

A participant creates her public and secret keys by the following procedure:

1. Select 2 large prime numbers $p$ and $q$ (say 200 digits each)
2. Compute $n = pq$
3. Select a small number $e$ that is relatively prime to $(p-1)(q-1)$
4. Compute $d$ that is an inverse of $e$ modulo $(p-1)(q-1)$
5. Publish $(e, n)$ as her public key
6. Keep $(d, n)$ as her secret (private) key

Anyone who wants to find out values $d$ here to find $p$ and $q$ be caused $d$ depends on $p$ and $q$. To factor a 400-digit integer may require billions of years!