CSC10004: Data Structure and Algorithms

Lecture 2: Asymptotic notations

Lecturer: Bùi Văn Thạch
TA: Ngô Đình Hy/Lê Thị Thu Hiền
{bvthach,ndhy}@fit.hcmus.edu.vn, lththien@hcmus.edu.vn

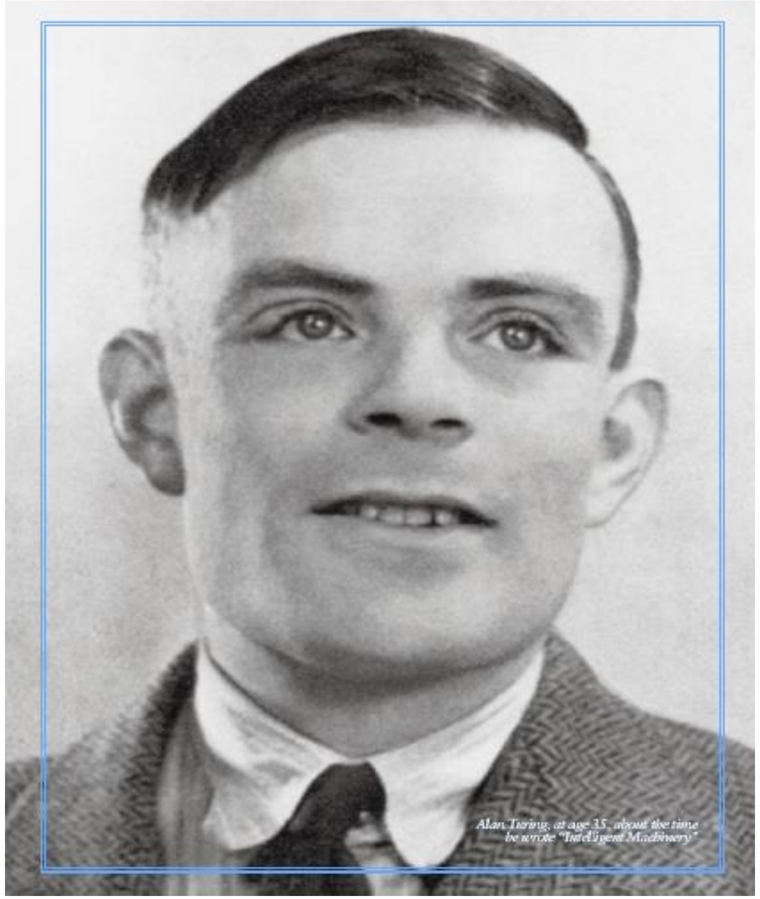
Outline

- Turing machines
- RAM model
- What is an algorithm?
- Asymptotic notations

Outline

- Turing machines
- RAM model
- What is an algorithm?
- Asymptotic notations

- The theory of computation and the practical application it made possible — the computer — was developed by an Englishman called Alan Turing.



Definition

Turing's machine — which came to be called the Turing machine — was this:

1. A tape of infinite length.
2. Finitely many squares of the tape have a single symbol from a finite language.
3. Someone (or something) that can read the squares and write in them.
4. At any time, the machine is in one of a finite number of internal states.
5. The machine has instructions that determine what it does given its internal state and the symbol it encounters on the tape. It can

– change its internal state;

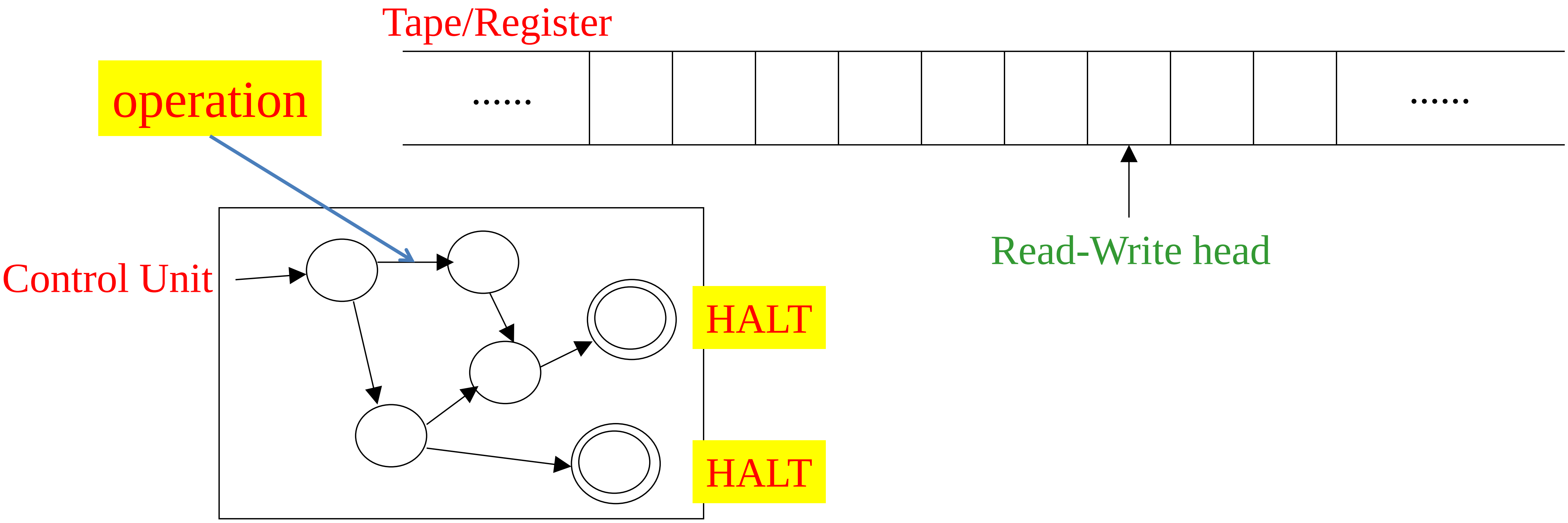
– change the symbol on the square;

– move forward;

– move backward;

– halt (i.e. stop).

- It is essential to the idea of a Turing machine that it is not a physical machine, but an abstract one — a set of procedures.



Outline

- Turing machines
- RAM model
- What is an algorithm?
- Asymptotic notations

Random-access machine (RAM) model

- Simple operations (arithmetic, comparison, conditional, etc.) each take the same, constant amount of time.

- Data stored in an infinite array of registers $(0, 1, 2, \ldots)$, each of which can hold $c \log x$ bits, where
  - x: problem size
  - c: some constant independent of x

Outline

- Turing machines
- RAM model
- What is an algorithm?
- Asymptotic notations

Definition

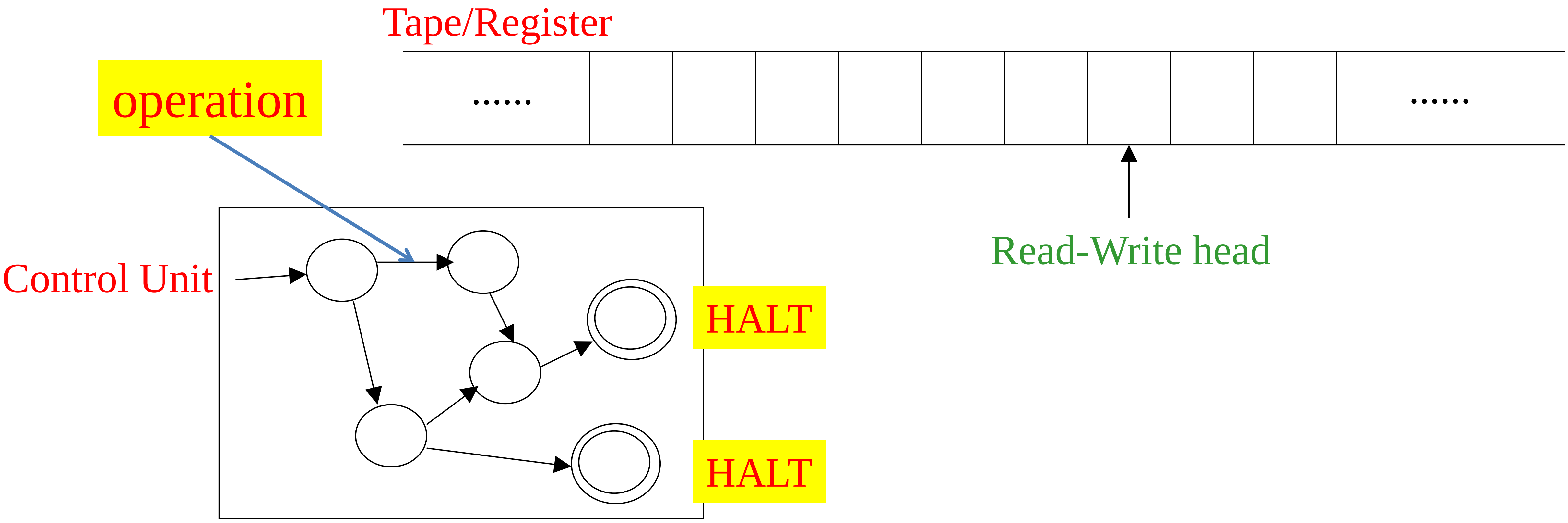
- An algorithm is a sequence of unambiguous instructions for solving a problem, that is, for obtaining a required output for any legitimate input in a finite amount of time.

$$\text{instance} \longrightarrow \boxed{\text{Algorithm}} \longrightarrow \text{output}$$

- Analysis of algorithms is the quantitative study of the performance of algorithms, in terms of their run time, memory usage, or other properties.

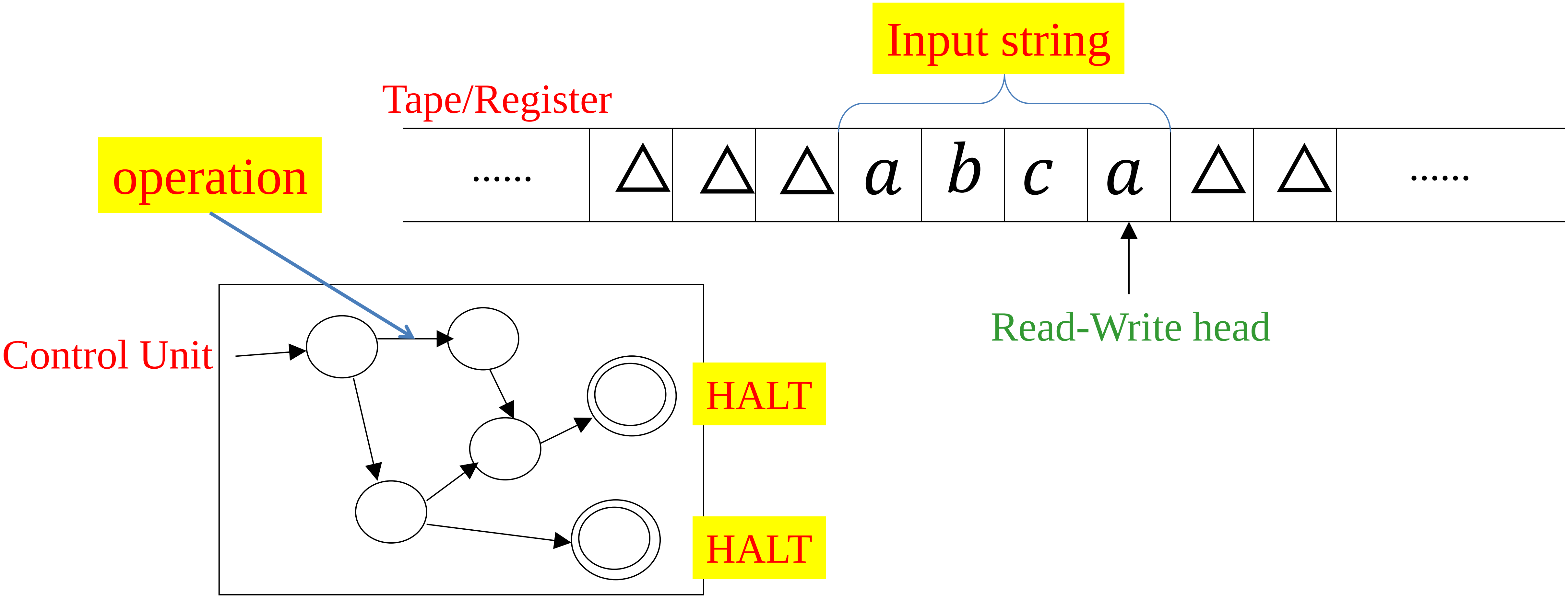
Informal descriptions

- Run time = #operations
- Memory usage = tape length



Informal descriptions

- The problem size = the length of the input string in the tape



Question (1/4)

Do these two algorithms have the same asymptotic running time?

```
int AlgA(int n) {
    int sum = 0;
    for(int i=0; i<n; i++) {
        for(int j=0; j<i; j++) {
            sum++;
        }
    }
    return sum;
}
```

```
int AlgB(int n) {
    int sum = 0;
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            sum++;
        }
    }
    return sum;
}
```

Question (2/4)

- In an iterative algorithm, let a_i be the number of operations, e.g, comparisons and assignments, at iteration i.
- A common tool for analyzing the iterative algorithms is the summation:

$$\sum_{i=\ell}^{n} a_i = a_\ell + a_{\ell+1} + \cdots + a_{n-1} + a_n$$

- If the upper limit is infinite, we interpret this as an implicit limit:

$$\sum_{i=\ell}^{\infty} a_i = \lim_{n\to\infty} \sum_{i=\ell}^{n} a_i$$

Question (3/4)

$$A(n) = \sum_{i=0}^{n-1}\sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$$
$$= \Theta(n^2)$$

$$B(n) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} 1 = \sum_{i=0}^{n-1} n = n^2$$
$$= \Theta(n^2)$$

```
int AlgA(int n) {
    int sum = 0;
    for(int i=0; i<n; i++) {
        for(int j=0; j<i; j++) {
            sum++;
        }
    }
    return sum;
}
```

```
int AlgB(int n) {
    int sum = 0;
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            sum++;
        }
    }
    return sum;
}
```

Question (4/4)

Do these two algorithms have the same asymptotic running time?

➔Yes, the run times are both $\Theta(n^2)$.

```
int AlgA(int n) {
    int sum = 0;
    for(int i=0; i<n; i++) {
        for(int j=0; j<i; j++) {
            sum++;
        }
    }
    return sum;
}
```

```
int AlgB(int n) {
    int sum = 0;
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            sum++;
        }
    }
    return sum;
}
```

Exercise

Do these two algorithms have the same asymptotic running time?

```
int AlgC(int n) {
    int sum = 0;
    for(int i=0; i<n; i*=2) {
        for(int j=0; j<i; j++) {
            sum++;
        }
    }
    return sum;
}
```

```
int AlgD(int n) {
    int sum = 0;
    for(int i=0; i<n; i*=2) {
        for(int j=0; j<n; j++) {
            sum++;
        }
    }
    return sum;
}
```

Solution

- NO, the asymptotic run times are different. Observe that i is always a power of 2. Let $i = 2k$, so that $k = \log_2 n$.
- $C(n) = \sum_{k=0}^{\lceil \log_2 n \rceil - 1} \sum_{j=0}^{2^k - 1} 1 = \sum_{k=0}^{\lceil \log_2 n \rceil - 1} 2^k = 2^{\lceil \log_2 n \rceil} - 1 = \Theta(n)$
- $D(n) = \sum_{k=0}^{\lceil \log_2 n \rceil - 1} \sum_{j=0}^{n-1} 1 = \sum_{k=0}^{\lceil \log_2 n \rceil - 1} n = n\lceil \log_2 n \rceil = \Theta(n \ln n)$

Outline

- Turing machines
- RAM model
- What is an algorithm?
- Asymptotic notations
  - Big-Oh (O)
  - Big-Omega (Ω)
  - Big-Theta (Θ)
  - Little-Oh (o)
  - Little-Omega (ω)

- In the analysis of algorithms, we are usually interested in how the performance of our algorithm changes as the problem input size increases.
- The primary tools for measuring the growth rate of a function that describes the run time of an algorithm are the asymptotic notations.
- This provides a way of studying the algorithms themselves, independent of any specific hardware, operating system, compiler, programmer, etc.

Overview

- Let $L = \lim_{x\to\infty} \frac{f(x)}{g(x)}$
- We have:

		$f < g$	$f \geq g$	$f = g$	$f \leq g$	$f > g$
	$f(x) =$	$o(g(x))$	$\Omega(g(x))$	$\Theta(g(x))$	$O(g(x))$	$\omega(g(x))$
L						
0		$\times$			$\times$	
$(0, \infty)$			$\times$	$\times$	$\times$	
∞			$\times$			$\times$

Big-Oh notation (1/3): definition

- f is asymptotically bounded ABOVE by g up to constant factor c.
- Write: $f(x) = O\big(g(x)\big)$ or $f(x) \in O\big(g(x)\big)$.
- Mathematically, $\exists c > 0 \text{ and } \exists x_0 > 0 : \forall x \geq x_0, |f(x)| \leq cg(x)$.
- To prove that $f(x) \in O(g(x))$, we need to provide the existence of a pair (c, x_0).

Landau, Edmund. *Handbuch der Lehre von der Verteilung der Primazahlen*. [*Handbook on the theory of the distribution of the primes*] Vol. 1. BG Teubner, 1909.

Big-Oh notation (2/3): examples

- $x^2 + 2x + 5 = O(x^2)$.

  Proof: Select $(x_0 = 1, c = 10)$ then $x^2 + 2x + 5 \leq 10x^2\ \forall x \geq x_0$.

- $x^2 + 2x + 5 \notin O(x)$.

  Proof: Assume there exists a pair $(x_0, c > 0)$ such that $x^2 + 2x + 5 \leq cx\ \forall x \geq x_0$. Then for all $x \geq x_0$:

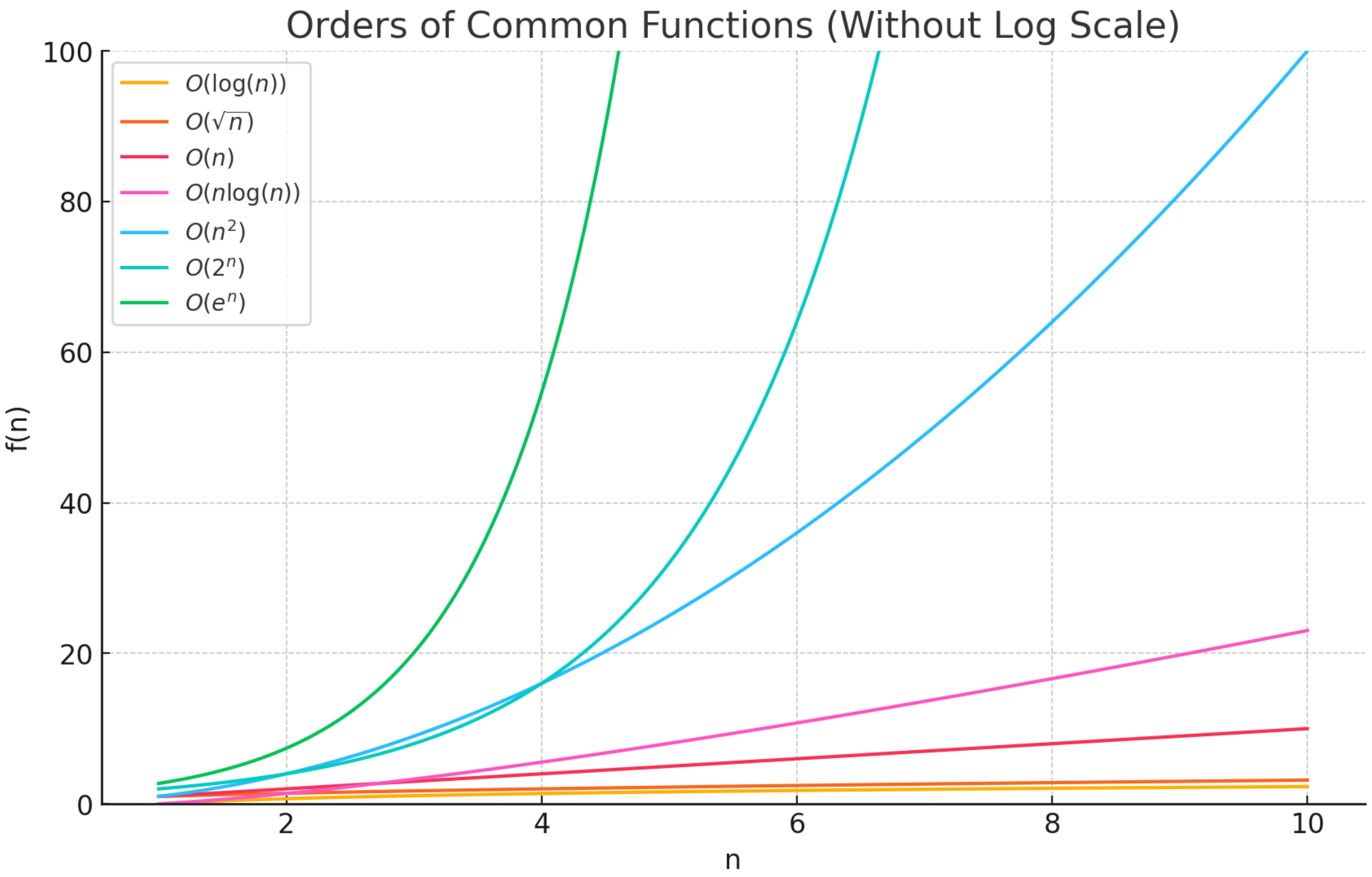
  $$x^2 + (2 - c)x + 5 = \left(x - \left(1 - \frac{c}{2}\right)\right)^2 + \left(5 - \left(1 - \frac{c}{2}\right)^2\right) \leq 0$$

  This inequality does not hold when x goes to infinity.

Big-Oh notation (3/3): common functions

Notation	Name (+ function)	Example
$O(1)$	Constant	
$O(\log\log n)$	double logarithmic	
$O(\log n)$	logarithmic	
$O(\log^c n), c > 1$	polylogarithmic	
$O(n^\alpha), 0 < \alpha < 1$	fractional power	
$O(n)$	Linear	
$O(n\log n)$	Quasilinear	
$O(n^2)$	Quadratic	
$O(n^c), 1 < c$	Polynomial	
$O(c^n), c > 1$	Exponential	
$O(n!)$	factorial	

Big-Oh notation (3/3): plot of common function orders



Big-Omega notation (1/2): definition

- f is asymptotically bounded BELOW by g.
- Write: $f(x) = \Omega\big(g(x)\big)$ or $f(x) \in \Omega\big(g(x)\big)$.
- Mathematically, $\exists c > 0 \text{ and } \exists x_0 > 0 : \forall x \geq x_0, f(x) \geq cg(x)$.
- To prove that $f(x) \in \Omega(g(x))$, we need to provide the existence of a pair (c, x_0).
- Note that: $f(x) = \Omega\big(g(x)\big) \Leftrightarrow g(x) = O\big(f(x)\big)$.

Knuth, Donald E. "Big omicron and big omega and big theta." *ACM Sigact News* 8.2 (1976): 18-24.

Big-Omega notation (2/2): examples (1/2)

- $x^2 + 2x + 5 = \Omega(x)$.

  Proof: Select $(x_0 = 1, c = 1)$ then $x^2 + 2x + 5 \geq 1 \cdot x \; \forall x \geq x_0$.

- $x^2 + 2x + 5 = \Omega(x^2)$.

  Proof: Select $(x_0 = 1, c = 1)$ then $x^2 + 2x + 5 \geq 1 \cdot x^2 \; \forall x \geq x_0$.

Big-Omega notation (2/2): examples (2/2)

- $x^2 + 2x + 5 \notin \Omega(x^3)$.

  Proof: Assume there exists a pair $(x_0 > 0, c > 0)$ such that $x^2 + 2x + 5 \geq cx^3 \ \forall x \geq x_0$. Then for all $x \geq x_0$:

  $$-\mathrm{x}^2(\mathrm{c}x - 1) + x + 5 \geq 0$$

  This inequality does not hold because when x goes to infinity, the right inequality goes to negative infinity.

Big-Theta notation (1/2): definition

- f is asymptotically bounded by g both ABOVE (with constant factor c_2) and BELOW (with constant factor c_1).

- Write: $f(x) = \Theta\big(g(x)\big)$ or $f(x) \in \Theta\big(g(x)\big)$.

- Mathematically, $\exists c_1, c_2 > 0 \text{ and } \exists x_0 > 0 : \forall x \geq x_0, c_1 g(x) \leq f(x) \leq c_2 g(x)$.

- To prove that $f(x) \in \Omega(g(x))$, we need to provide the existence of a triple (c_1, c_2, x_0).

- Note that: $f(x) = \Theta\big(g(x)\big) \Leftrightarrow g(x) = O\big(f(x)\big) \text{ and } f(x) = O\big(g(x)\big)$.

Knuth, Donald E. "Big omicron and big omega and big theta." *ACM Sigact News* 8.2 (1976): 18-24.

Big-Theta notation (2/2): examples (1/2)

- $x^2 + 2x + 5 = \Theta(x^2)$.

  Proof: Select $(c_1 = 1, c_2 = 10, x_0 = 1)$ then $1 \cdot x^2 \leq x^2 + 2x + 5 \leq 10x^2 \ \forall x \geq x_0$.

- $x^2 + 2x + 5 \neq \Theta(x)$.

  Proof: Assume there exists a triple $(c_1 > 0, c_2 > 0, x_0)$ such that $c_1 x \leq x^2 + 2x + 5 \leq c_2 x \ \forall x \geq x_0$. Then for all $x \geq x_0$:

  $$x^2 + (2 - c)x + 5 = \left(x - \left(1 - \frac{c_2}{2}\right)\right)^2 + \left(5 - \left(1 - \frac{c_2}{2}\right)^2\right) \leq 0$$

  This inequality does not hold when x goes to infinity.

Big-Theta notation (2/2): examples (2/2)

- $x^2 + 2x + 5 \notin \Theta(x^3)$.

  Proof: Assume there exists a pair $(x_0, c > 0)$ such that $x^2 + 2x + 5 \geq cx^3\ \forall x \geq x_0$. Then for all $x \geq x_0$:
  $$-\mathrm{x}^2(\mathrm{c}x - 1) + x + 5 \geq 0$$

  This inequality does not hold because when x goes to infinity, the right inequality goes to negative infinity.

Little-Oh notation (1/2): definition

- f is asymptotically dominated by g (for ANY constant factor c).
- Write: $f(x) = o\big(g(x)\big)$ or $f(x) \in o\big(g(x)\big)$.
- Mathematically, $\forall c > 0 \text{ and } \exists x_0 > 0: \forall x \geq x_0, f(x) \leq cg(x)$.
- To prove that $f(x) \in o(g(x))$, we need to provide the existence of x_0 for every $c > 0$.
- Note that: $f(x) = o\big(g(x)\big) \Leftrightarrow \lim_{x \to \infty} \frac{f(x)}{g(x)} = 0$.

Landau, Edmund. *Handbuch der Lehre von der Verteilung der Primazahlen*. [*Handbook on the theory of the distribution of the primes*] Vol. 1. BG Teubner, 1909.

Little-Oh notation (2/2): example

- $x^2 + 2x + 5 = o(x^3)$.

  Proof: $\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{x^2+2x+5}{x^3} = \lim_{x\to\infty} \left(\frac{1}{x} + \frac{2}{x^2} + \frac{5}{x^3}\right) = 0$.

- $x^2 + 2x + 5 \notin o(x)$.

  Proof: $\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{x^2+2x+5}{x} = \lim_{x\to\infty} \left(x + 2 + \frac{5}{x}\right) = \infty$.

Little-Omega notation (1/2): definition

- f asymptotically dominates g.
- Write: $f(x) = \omega\big(g(x)\big)$ or $f(x) \in \omega\big(g(x)\big)$.
- Mathematically, $\forall c > 0$ and $\exists x_0 > 0 \colon \forall x \geq x_0, f(x) \geq cg(x)$.
- To prove that $f(x) \in o(g(x))$, we need to provide the existence of x_0 for every $c > 0$.
- Note that: $f(x) = \omega\big(g(x)\big) \Leftrightarrow \lim\limits_{x \to \infty} \frac{f(x)}{g(x)} = \infty$.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

Little-Oh notation (2/2): example

- $x^2 + 2x + 5 = \omega(x)$.

- Proof: $\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{x^2+2x+5}{x} = \lim_{x\to\infty} \left(x + 2 + \frac{5}{x}\right) = \infty$.

- $x^2 + 2x + 5 \notin o(x^3)$.

  Proof: $\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{x^2+2x+5}{x^3} = \lim_{x\to\infty} \left(\frac{1}{x} + \frac{2}{x^2} + \frac{5}{x^3}\right) = 0$.

Exercises (1/5)

Prove the following statements:

1. $n^3 + 1000n^2 = O(n^4)$.
2. $\log n = O(n)$.
3. $\log n = O(\sqrt{n})$.
4. $n! \notin O(n^c)$ for any positive constant c.
5. $n^a = O(b^n)$ for any positive constants a and $b > 1$.
6. $\log n! = O(n \log n)$ and $\log n! \geq \frac{n}{2} \log \frac{n}{2}$ to get $\log n! = \Theta(n \log n)$.
7. $1000x^3 - x^2 + 79 = \Theta(x^3)$.

Exercises (2/5)

8. Let $f(x) = \mathrm{O}(h(x))$ and $g(x) = \mathrm{O}(h(x))$. Let $a, b > 0$. Prove that $af(x) + bg(x) = \mathrm{O}(h(x))$.
9. Let $f_1(x) = \mathrm{O}(g_1(x))$ and $f_2(x) = \mathrm{O}(g_2(x))$. Prove that $f_1(x)f_2(x) = \mathrm{O}(g_1(x)g_2(x))$.
10. Prove that $x^n + a_{n-1}x^{n-1} + \cdots + a_0 = \mathrm{O}(x^n)$.
11. Prove that $x^n + a_{n-1}x^{n-1} + \cdots + a_0 = \Theta(x^n)$.
12. $\log n = O(n^c)$ where $0 < c < 1$.

Exercises (3/5)

- How many comparisons, and assignments are there in the following code fragment with the size n?

```
sum = 0;
for (i = 0; i < n; i++)
{
    cin >> x;
    sum = sum + x;
}
```

Exercises (4/5)

How many assignments are there in the following code fragment with the size *n*?

```
for (i = 0; i < n ; i++)
      for (j = 0; j < n; j++)
      {
            C[i][j] = 0;
            for (k = 0; k < n; k++)
                  C[i][j] = C[i][j] + A[i][k]*B[k][j];
      }
```

Exercises (5/5)

- Give the order of growth (as a function of N) of the running time of the following code fragment:

```
int sum = 0;
for (int n = N; n > 0; n /= 2)
    for (int i = 0; i < n; i++)
        sum++;
```

Q & A

Solutions: Problem 1

- For all $n \geq 1$:
- $|n^3| = n^3 \leq n^4$
- $|1000n^2| = 1000n^2 \leq 1000n^4$
- It implies that $|n^3 + 1000n^2| \leq |n^3| + |1000n^2| \leq 1001n^4 \Rightarrow n^3 + 1000n^2 = O(n^4)$

Solutions: Problem 2

- For all $x \geq 1$, let $f(x) = x - \log x$, then $f'(x) = 1 - \frac{1}{x} \geq 0$.
- Therefore, $f(x)$ is monotonically increasing.
- It implies that $x - \log x = f(x) \geq f(1) = 1 > 0 \Rightarrow x > \log x$
- Therefore, for all $n \geq 1$, $0 \leq \log n < n \Rightarrow |\log n| < n \Rightarrow \log n = O(n)$

Solutions: Problem 3

- For all $x \geq 1$, let $f(x) = x - \log x$, then $f'(x) = 1 - \frac{1}{x} \geq 0$.
- Therefore, $f(x)$ is monotonically increasing.
- It implies that $x - \log x = f(x) \geq f(1) = 1 > 0 \Rightarrow x > \log x$
- Then, replacing x with $\sqrt{x}$, it implies that $\sqrt{x} > \log \sqrt{x} \Rightarrow 2\sqrt{x} > \log x$
- Therefore, for all $n \geq 1$, $0 \leq \log n < 2\sqrt{n} \Rightarrow |\log n| < 2\sqrt{n} \Rightarrow \log n = O(\sqrt{n})$

Solutions: Problem 4

- Assume that $\exists n_0 > 0, k > 0, \forall n \geq n_0$:
- $n! \leq kn^c$
- Let $m_0 = \min\{x \in \mathbb{Z} | x \geq n_0\}$, $h = \min\{x \in \mathbb{Z} | x \geq k\}$, and $d = \min\{x \in \mathbb{Z} | x \geq c\}$.
- Then, let $m = m_0 + h + 2d + 1 \in \mathbb{Z}$
- $m \geq 2d + 1 \Rightarrow m! \geq (m - 2d) \ldots (m - d - 1)(m - d)(m - d + 1) \ldots m$

$$\Rightarrow m! \geq (m - d) \prod_{i=1}^{d} (m - d - i)(m - d + i)$$

$$\Rightarrow m! \geq (m - d) \prod_{i=1}^{d} \left((m - d)^2 - i^2\right)$$

- For all $1 \leq i \leq d$ and $m \geq 2d + 1$:
- $(m - d)^2 - i^2 \geq (m - d)^2 - d^2 = m^2 - 2md = m(m - 2d) \geq m$
- It implies that $m! \geq (m - d)m^d$
- However, $m - d > h$, then $m! > hm^d \geq km^c$. That contradicts the assumption.

Solutions: Problem 5

- Let $c = \frac{a}{\log b} > 0$ because $a > 0$ and $b > 1$
- For all $x \geq c > 0$, let $f(x) = x - c\log x$, then $f'(x) = 1 - \frac{c}{x} \geq 0$.
- Therefore, $f(x)$ is monotonically increasing.
- It implies that $x - c\log x = f(x) \geq f(c) = -\frac{\log k}{\log b}$ where $k = b^{-f(c)} > 0$
- $c\log x \leq x + \frac{\log k}{\log b} \Rightarrow a\log x \leq x\log b + \log k$

$$\Rightarrow \log x^a \leq \log kb^x$$
$$\Rightarrow x^a \leq kb^x$$

- So, with $c = \frac{a}{\log b}$ and $k = b^{-f(c)}$, for all $n \geq n_0 = \min\{x \in \mathbb{Z} | x \geq c\}$: $n^a \leq kb^n \Rightarrow n^a = O(b^n)$

Solutions: Problem 6

- Prove that $\log n! = \Theta(n \log n)$
- For all $n \geq 4$:
- $\log n! = \sum_{k=1}^{n} \log k < \sum_{k=1}^{n} \log n = n \log n \Rightarrow \log n! = O(n \log n)$
- $\log n! = \sum_{k=1}^{n} \log k > \sum_{k=\left\lceil \frac{n}{2} \right\rceil}^{2\left\lceil \frac{n}{2} \right\rceil - 1} \log k > \sum_{k=\left\lceil \frac{n}{2} \right\rceil}^{2\left\lceil \frac{n}{2} \right\rceil - 1} \log \left\lceil \frac{n}{2} \right\rceil = \left\lceil \frac{n}{2} \right\rceil \log \left\lceil \frac{n}{2} \right\rceil \geq \frac{n}{2} \log \frac{n}{2}$
-
- $\frac{n}{2} \log \frac{n}{2} = \frac{1}{2} n(\log n - \log 2) = \frac{1}{4} n\big(\log n + (\log n - \log 4)\big) \geq \frac{1}{4} n \log n$
- It implies that $\log n! > \frac{1}{4} n \log n \Rightarrow \log n! = \Omega(n \log n) \Rightarrow \log n! = \Theta(n \log n)$

Solutions: Problem 7

- For all $x \geq 1$
- $|1000x^3 - x^2 + 79| \leq 1000x^3 + x^2 + 79 \leq 1080x^3 \Rightarrow 1000x^3 - x^2 + 79 = O(x^3)$
- $1000x^3 - x^2 + 79 \geq 999x^3 + (x^3 - x^2) + 79 > 999x^3 > 0$

$$\Rightarrow 1000x^3 - x^2 + 79 = \Omega(x^3)$$

$$\Rightarrow 1000x^3 - x^2 + 79 = \Theta(x^3)$$

Solutions: Problem 8

- $f(x) = O\big(h(x)\big) \Leftrightarrow \exists c_1 > 0, x_1 > 0, \forall x \geq x_1 : |f(x)| \leq c_1 h(x)$
- $g(x) = O\big(h(x)\big) \Leftrightarrow \exists c_2 > 0, x_2 > 0, \forall x \geq x_2 : |g(x)| \leq c_2 h(x)$
- Then, it implies that $\exists c = ac_1 + bc_2 > 0, x_0 = \max\{x_1, x_2\} > 0, \forall x \geq x_0$:
- $|af(x) + bg(x)| \leq a|f(x)| + b|g(x)| \leq (ac_1 + bc_2)h(x) = ch(x)$
- Therefore, $af(x) + bg(x) = \mathrm{O}\big(h(x)\big)$.

Solutions: Problem 9

- $f_1(x) = O\big(g_1(x)\big) \Leftrightarrow \exists c_1 > 0, x_1 > 0, \forall x \geq x_1 : |f_1(x)| \leq c_1 g_1(x)$
- $f_2(x) = O\big(g_2(x)\big) \Leftrightarrow \exists c_2 > 0, x_2 > 0, \forall x \geq x_2 : |f_2(x)| \leq c_2 g_2(x)$
- Then, it implies that $\exists c = c_1 c_2 > 0, x_0 = \max\{x_1, x_2\} > 0, \forall x \geq x_0$:
- $|f_1(x) f_2(x)| \leq c_1 c_2 g_1(x) g_2(x) = c g_1(x) g_2(x)$
- Therefore, $f_1(x) f_2(x) = \mathrm{O}\big(g_1(x) g_2(x)\big)$.

Solutions: Problem 10

- Prove by induction:
- $+ n = 0: 1 = O(1)$.
- $+$ Assume that the statement holds with $n = k$:
- $x^k + a_k x^{k-1} + \cdots + a_1 = O(x^k)$
- And $x = O(x)$. Then, it implies that $x^{k+1} + a_k x^k + \cdots + a_1 x = O(x^{k+1})$
- And $a_0 = O(x^{k+1})$.
- So, $x^{k+1} + a_k x^k + \cdots + a_1 x + a_0 = O(x^{k+1})$.

Solutions: Problem 11

- Let $c = \frac{1}{n+1} > 0$ and $x_0 = \max\left\{(n+1)|a_{n-1}|, \sqrt{(n+1)|a_{n-2}|}, \dots, \sqrt[n]{(n+1)|a_0|}\right\}$. Then, for all $x \geq x_0$, we have:

$$x \geq (n+1)|a_{n-1}|$$
$$x^2 \geq (n+1)|a_{n-2}|$$
$$\dots$$
$$x^n \geq (n+1)|a_0|$$

- Therefore, it implies that:

$$\frac{1}{n+1}x^n \geq |a_{n-1}|x^{n-1} \geq -a_{n-1}x^{n-1}$$
$$\frac{1}{n+1}x^n \geq |a_{n-2}|x^{n-2} \geq -a_{n-2}x^{n-2}$$
$$\dots$$
$$\frac{1}{n+1}x^n \geq |a_0| \geq -a_0$$

- $\Rightarrow x^n + a_{n-1}x^{n-1} + \cdots + a_0 \geq \frac{1}{n+1}x^n = cx^n$
- So, $x^{k+1} + a_k x^k + \cdots + a_1 x + a_0 = \Omega\left(x^{k+1}\right)$.
- Therefore, $x^{k+1} + a_k x^k + \cdots + a_1 x + a_0 = \Theta\left(x^{k+1}\right)$.

Solutions: Problem 12

- For all $x \geq 1$, let $f(x) = x - \log x$, then $f'(x) = 1 - \frac{1}{x} \geq 0$.
- Therefore, $f(x)$ is monotonically increasing.
- It implies that $x - \log x = f(x) \geq f(1) = 1 > 0 \Rightarrow x > \log x$
- Then, replacing x with x^c, it implies that $x^c > \log x^c \Rightarrow \frac{1}{c} x^c > \log x$
- Therefore, for all $n \geq 1$, $0 \leq \log n < \frac{1}{c} n^c \Rightarrow |\log n| < \frac{1}{c} n^c \Rightarrow \log n = O(n^c)$