Lecture 03 — Asymptotic Notations

COMPSCI 308 — Design and Analysis of Algorithms

Xing Shi Cai https://newptcai.gitlab.io

Duke Kunshan University, Kunshan, China

Jan-Mar 2026

Summary

Random Access Machine (RAM)

Order of Growth of Running Time

Asymptotic Analysis

Assignments[1]

Practice makes perfect!

Introduction to Algorithms (ITA)

📖 Read —

- Section 3.1

✏️ Practice —

- Exercises 3.1: 1–3.

[1] 💣 Assignments will not be collected; however, quiz problems will be selected from them. (This includes both Readings and Exercises.)

Random Access Machine (ram)

Random Access Machine (ram)

We will design and analyze algorithms on an abstract computation model Random Access Machine (ram)[2] with the following properties —

- It is an abstract 🖥.

[2] This is in Section 2.2.

Random Access Machine (RAM)

We will design and analyze algorithms on an abstract computation model Random Access Machine (RAM)[2] with the following properties —

- It is an abstract 💻.
- Each instruction is executed sequentially on RAM.

[2] This is in Section 2.2.

Random Access Machine (RAM)

We will design and analyze algorithms on an abstract computation model Random Access Machine (RAM)[2] with the following properties —

- It is an abstract 💻.
- Each instruction is executed sequentially on RAM.
- Each simple operation ($+, \times, -, =$, `if`) takes exactly one time unit.

😵 Didn't we say that $\times$ might take more than one time unit?

[2] This is in Section 2.2.

Random Access Machine (RAM)

We will design and analyze algorithms on an abstract computation model Random Access Machine (RAM)[2] with the following properties —

- It is an abstract  .
- Each instruction is executed sequentially on RAM.
- Each simple operation ($+, \times, -, =,$ `if`) takes exactly one time unit.
- Loops and calls of subroutines are *not* simple operations.

[2] This is in Section 2.2.

Random Access Machine (RAM)

We will design and analyze algorithms on an abstract computation model Random Access Machine (RAM)[2] with the following properties —

- It is an abstract 💻.
- Each instruction is executed sequentially on RAM.
- Each simple operation ($+, \times, -, =,$ `if`) takes exactly one time unit.
- Loops and calls of subroutines are *not* simple operations.
- Each memory access takes exactly one time unit.

[2] This is in Section 2.2.

Random Access Machine (RAM)

We will design and analyze algorithms on an abstract computation model Random Access Machine (RAM)[2] with the following properties —

- It is an abstract 🖥.
- Each instruction is executed sequentially on RAM.
- Each simple operation ($+, \times, -, =,$ `if`) takes exactly one time unit.
- Loops and calls of subroutines are *not* simple operations.
- Each memory access takes exactly one time unit.
- It has infinitely many memory cells.

[2] This is in Section 2.2.

What ram Can Do

The ram model contains instructions commonly found on a real 💻, and each instruction takes a constant amount of time —

- Arithmetic: ➕, ➖, ✖, ➗, remainder, floor, ceiling.
- Data movement: load, store, copy
- Control: conditional and unconditional branch, loop, subroutine call and return

But ram does *not* contain unrealistic instructions such as `sort`.

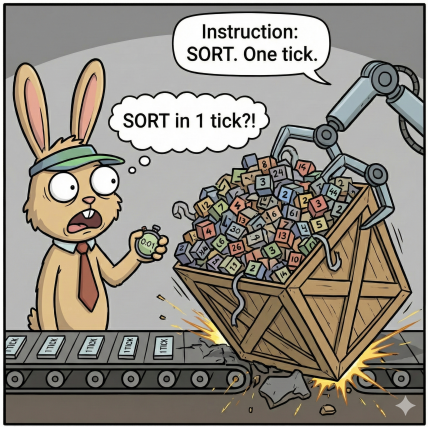


Figure 1: You cannot sort this mess instantly!

Running Time in RAM

In RAM, we count the number of instruction executions as running time, since most operations take constant time on a real 💻.

operation	example	nanoseconds
integer add	`a + b`	2.1
integer multiply	`a * b`	2.4
integer divide	`a / b`	5.4
floating-point add	`a + b`	4.6
floating-point multiply	`a * b`	4.2
floating-point divide	`a / b`	13.5
sine	`Math.sin(theta)`	91.3
arctangent	`Math.atan2(y, x)`	129.0
...	...	...

In ram, we count the number of instruction executions as running time, since most operations take constant time on a real 💻.

operation	example	nanoseconds
variable declaration	`int a`	c_1
assignment statement	`a = b`	c_2
integer compare	`a < b`	c_3
array element access	`a[i]`	c_4
array length	`a.length`	c_5
1D array allocation	`new int[N]`	$c_6 N$
2D array allocation	`new int[N][N]`	$c_7 N^2$

Figure 2: The running time of basic operations in Java on 💻

Gray Areas in the ram Model

Operations like exponentiation x^y may require multiple instructions on a real 💻.

Thus, they are *not* defined as instructions in the ram model.

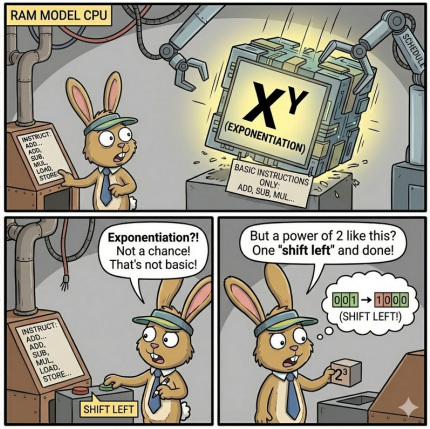


Figure 3: Is exponential an instruction?

Gray Areas in the ram Model

Operations like exponentiation x^y may require multiple instructions on a real 💻.

Thus, they are *not* defined as instructions in the RAM model.

However, specific cases such as computing 2^k with a small k can be executed as a constant-time operation through a shift left (one bit) instruction.

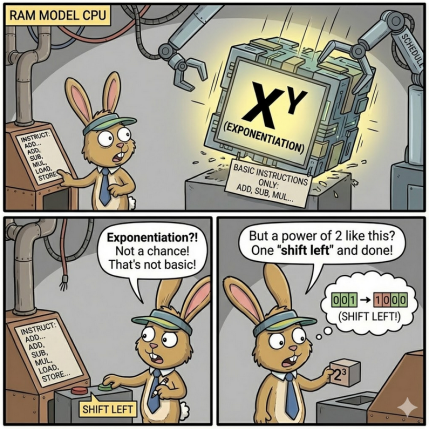


Figure 3: Is exponential an instruction?

Gray Areas in the ram Model

Operations like exponentiation x^y may require multiple instructions on a real 💻.

Thus, they are *not* defined as instructions in the ram model.

🍰 How can we compute 5^2 using *shift left*?

$$5 = 101_2 \xrightarrow{\text{shift left } ? \text{ bits}} \_\_\_\_\_ = 5^2$$

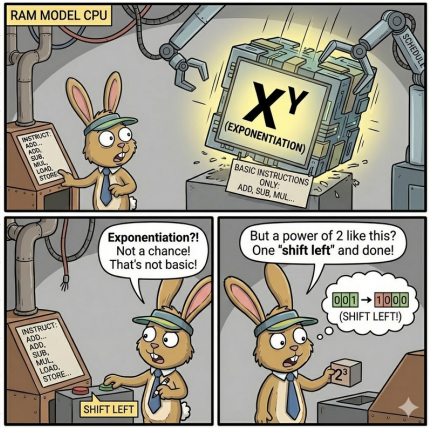


Figure 3: Is exponential an instruction?

Gray Areas in the ram Model

Operations like exponentiation x^y may require multiple instructions on a real 🖥️.

Thus, they are *not* defined as instructions in the RAM model.

💡 We will adhere to the set of instructions defined in the RAM model whenever possible and treat special cases, like 2^k, as constant-time operations.

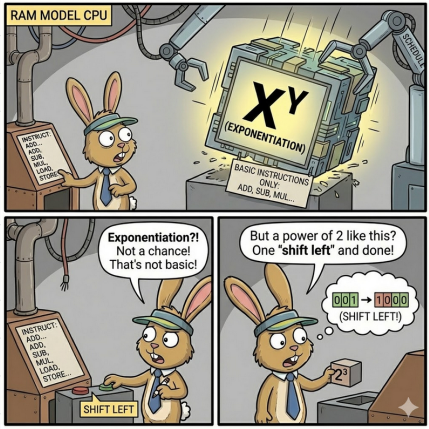


Figure 3: Is exponential an instruction?

Data Types in RAM

In the RAM model, two primary data types are used — integers and floating-point numbers.

While the focus is generally *not* on precision in floating-point arithmetic, there are scenarios where precision is critical.

Can you think of any?

What Is a Word?

In computing, a word refers to a fixed-sized piece of data that is processed as a unit by the CPU.

The size of a word varies by architecture —

- 8-bit — Zilog Z80
- 16-bit — Intel 8086
- 32-bit — Intel 80386
- 64-bit — AMD Athlon 64

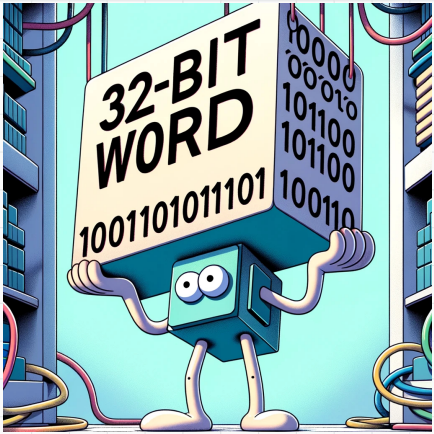


Figure 4: A 32-bit CPU

Assumptions on Word Size in ram

In ram, it is often assumed that there is a limit on the size of each word of data—

- When the input has size n, integers are typically represented by $c\lceil \lg n \rceil$ bits for some constant $c \geq 1$.

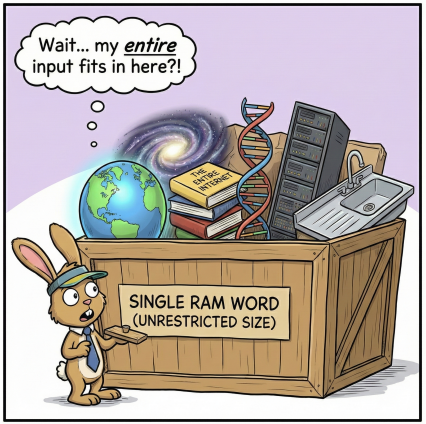


Figure 5: The universe in a single word

Assumptions on Word Size in RAM

In RAM, it is often assumed that there is a limit on the size of each word of data—

- When the input has size n, integers are typically represented by $c\lceil \lg n \rceil$ bits for some constant $c \geq 1$.[a]
- The constant $c \geq 1$ ensures that each word can hold any integer in $\{0, 1, \ldots, n\}$.

[a] In this course, $\lg x \log_2 x$.

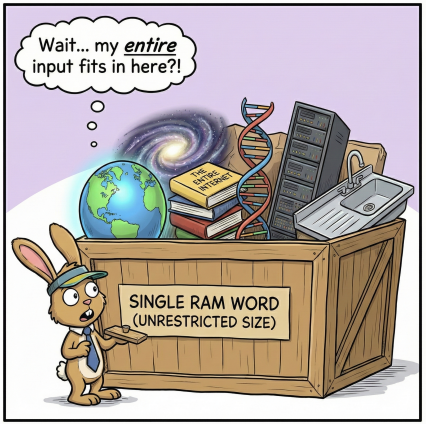


Figure 5: The universe in a single word

Assumptions on Word Size in ram

In ram, it is often assumed that there is a limit on the size of each word of data—

- When the input has size n, integers are typically represented by $c\lceil \lg n \rceil$ bits for some constant $c \geq 1$.
- The constant $c \geq 1$ ensures that each word can hold any integer in $\{0, 1, \ldots, n\}$.
- c is restricted to be a constant to prevent the word size from growing arbitrarily.

😆 Without any restriction, we can put any data in a single word!

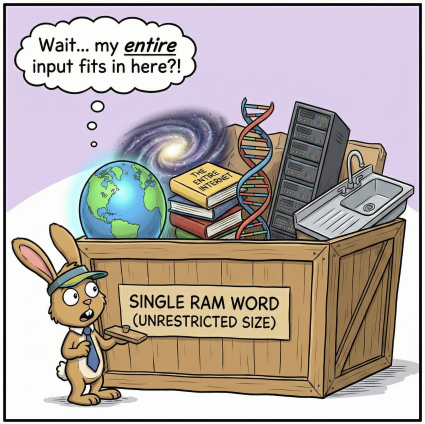


Figure 5: The universe in a single word

The Unrealistic ram Model

The ram model used in algorithm analysis is unrealistic in many ways.

For example, on a 💻:

- ✖ takes much longer than ➕.
- The compiler may unroll loops for efficiency.
- Memory access is not at all constant time, as assumed.

💬 If the ram model is unrealistic, why do we still use it in algorithm analysis?

🤪 All Models Are Wrong, Some Are Useful

Since all models are wrong, the scientist cannot obtain a "correct" one by excessive elaboration. … the ability to devise simple but evocative models is the signature of the great scientist …

— George Box

Figure 6: The flat earth model is good enough for building a house.

Order of Growth of Running Time

Order of Growth

Let $T(n)$ be the worst-case/average running time of an algorithm with input size n on RAM.

The order/rate of growth of $T(n)$ describes how fast $T(n)$ grows as n increases.

For example, if

$$T(n) = an^2 + bn + c,$$

we say its *order of growth* is

$$\Theta(n^2)$$

Typical Order of Growth

Some Functions Ordered by Growth Rate	Common Name
$\log n$	logarithmic
$\log^2 n$	polylogarithmic
$\sqrt{n}$	square root
n	linear
$n \log n$	linearithmic
n^2	quadratic
n^3	cubic
2^n	exponential

Table 1: Some Functions Ordered by Growth Rate

Why Order of Growth?

We care about the order of growth because —

- It is a simple characterization of the algorithm's efficiency.
- There are enormous differences between different orders.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Figure 7: Running Time of Algorithms with Different Orders of Growth on a Computer with 10^6 Instructions Per Second

Concerns with Precise Running Time

Why not aim for a precise running time like:

$$1.63n^2 + 3.5n + 8$$

Because —

- Achieving precise bounds can be exhausting.
- Coarser granularity reveals clearer patterns.
- Overly detailed measurement can be misleading.

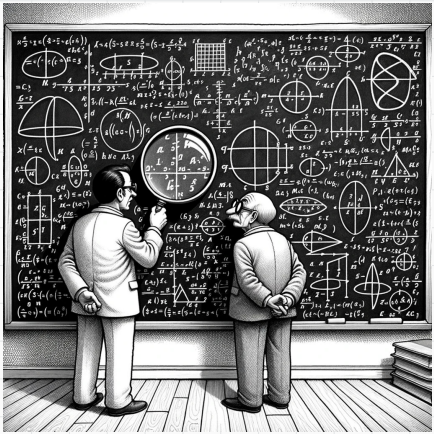


Figure 8: Too many details can be misleading

Asymptotic Analysis

In mathematics, asymptotic analysis, or asymptotics, is a method of describing the limiting behaviour of a function.

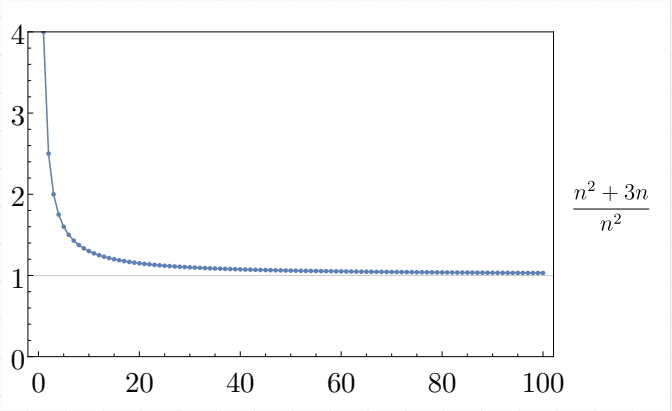


Figure 9: $(n^2 + 3)/n^2$

Asymptotic Analysis

In mathematics, asymptotic analysis, or asymptotics, is a method of describing the limiting behaviour of a function.

For example, when n is large,

$$n^2 + 3n \approx n^2$$

In other words, the term $3n$ becomes insignificant.

Asymptotic notation makes $\approx$ more precise by defining equations like

$$n^2 + 3n = \Theta(n^2).$$

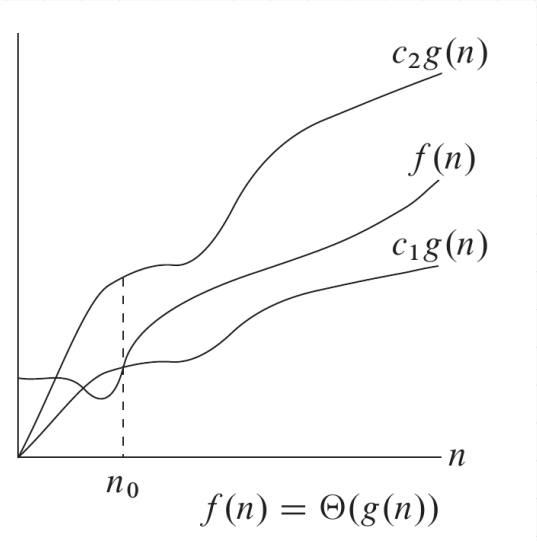
The Θ Notation

We say that $f(n) = \Theta(g(n))$ if there exist positive constants n_0, c_1 and c_2 such that

$$c_1 g(n) \leq f(n) \leq c_2 g(n),$$

for all $n \geq n_0$.

💡 Normally, we pick a $g(n)$ that is a bit simpler than $f(n)$.



📌 Example of Θ Notation

🍰 How can we show that

$$\frac{1}{2}n^2 - 3n = \Theta(n^2).$$

Quadratic Functions

For $a > 0$, we have

$$an^2 + bn + c = \Theta(n^2).$$

Quadratic Functions

For $a > 0$, we have

$$an^2 + bn + c = \Theta(n^2).$$

✏️ *Proof.*—For any n,

$$an^2 - |b|n - |c| \leq an^2 + bn + c \leq an^2 + |b|n + |c|.$$

Quadratic Functions

For $a > 0$, we have

$$an^2 + bn + c = \Theta(n^2).$$

✏️ *Proof.*—For any n,

$$an^2 - |b|n - |c| \leq an^2 + bn + c \leq an^2 + |b|n + |c|.$$

Let

$$n_0 = 2 \max\left((|b|)/a, \sqrt{(|c|)/a}\right).$$

If $n \geq n_0$, then $n \geq 2|b|/a$ and $n \geq 2\sqrt{|c|/a}$, so

$$|b|n \leq \frac{a}{2}n^2 \qquad \text{and} \qquad |c| \leq \frac{a}{4}n^2.$$

For $a > 0$, we have

$$an^2 + bn + c = \Theta(n^2).$$

✏️ *Proof.*—For any n,

$$an^2 - |b|n - |c| \leq an^2 + bn + c \leq an^2 + |b|n + |c|.$$

Let

$$n_0 = 2\max\left((|b|)/a, \sqrt{(|c|)/a}\right).$$

If $n \geq n_0$, then $n \geq 2|b|/a$ and $n \geq 2\sqrt{|c|/a}$, so

$$|b|n \leq \frac{a}{2}n^2 \qquad \text{and} \qquad |c| \leq \frac{a}{4}n^2.$$

Thus,

$$\frac{a}{4}n^2 \leq an^2 + bn + c \leq \frac{7a}{4}n^2, \qquad \text{for all } n \geq n_0.$$

Given that

$$p(n) = \sum_{i=0}^{d} a_i n^i,$$

where $a_d > 0$, we have

$$p(n) = \Theta(n^d).$$

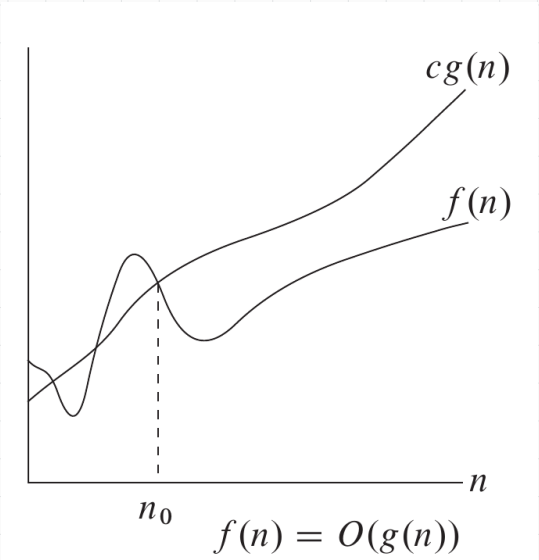
The O Notation

We say that $f(n) = O(g(n))$ if there exist positive constants n_0, c such that

$$0 \leq f(n) \leq cg(n),$$

for all $n \geq n_0$.

💡 This is to say that $f(n)$ increases at most as fast as $g(n)$.



📌 Example of O Notation

🍰 How can we show that

$$an^2 + bn + c = O(n^2),$$

if $a > 0$?

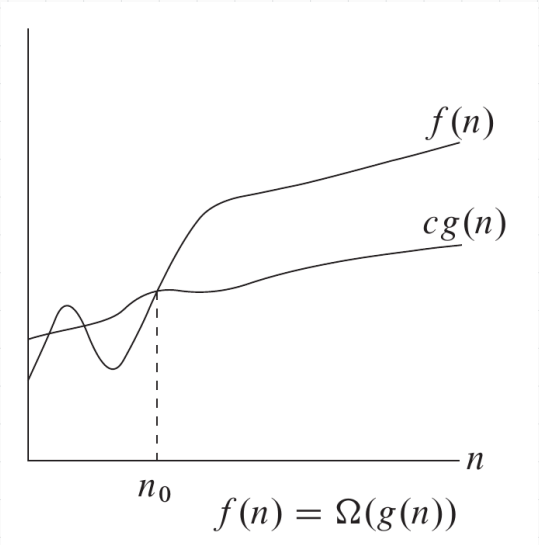
The Ω Notation

We say that $f(n) = \Omega(g(n))$ if there exist positive constants n_0, c such that

$$f(n) \geq cg(n),$$

for all $n \geq n_0$.

💡 This is to say that $f(n)$ increases at least as fast as $g(n)$.



🍰 Which of the following are correct?

1. $\log(n) + \log(n)^2 = O(n^2)$
2. $n \log(n) = O(n^2)$
3. $\log\left(n^k\right) = O(\log(n))$ for any constant $k > 0$
4. $2^n = O(n!)$
5. $\sqrt{n} + n = O(n \log n)$
6. $n^{0.5} = O(\log(n))$

📌 Example of Ω Notation

🍰 How can we show that

$$an^2 + bn + c = \Omega(n^2),$$

if $a > 0$?

Theorem 3.1

For any two functions $f(n)$ and $g(n)$, we have

$$f(n) = \Theta(g(n))$$

if and only if

$$f(n) = O(g(n)) \qquad \text{and} \qquad f(n) = \Omega(g(n)).$$

What are your main takeaways today? Any questions?