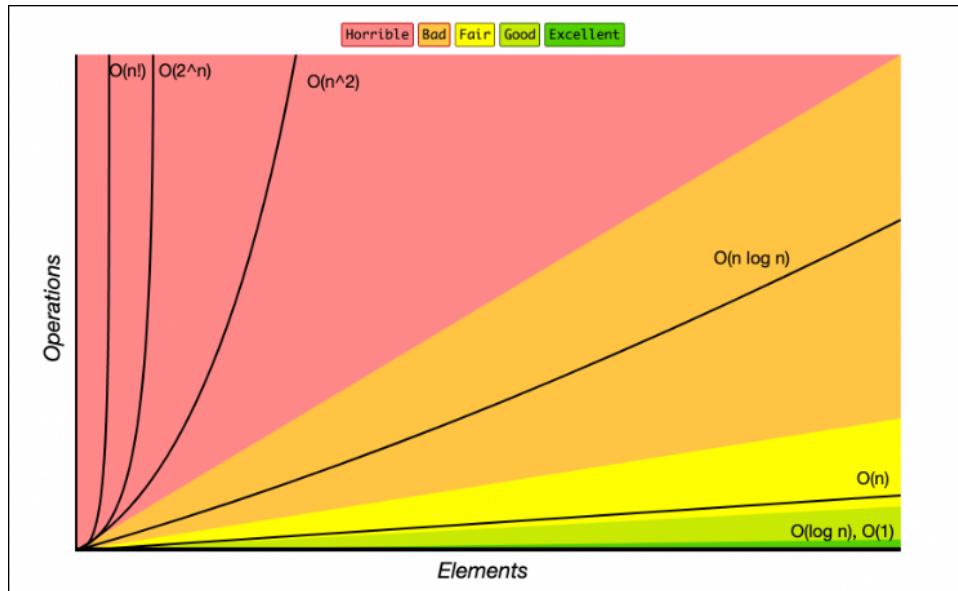


# Chapter 3: Characterizing Running Times

## Introduction

When we look at input sizes large enough to make only the order of growth of the running time relevant, we are studying **asymptotic** efficiency of algorithms. That is, we are concerned with how the running time increases with the size of the input *in the limit*, as the size of the input increases without bound.

Usually, an algorithm that is asymptotically more efficient will be the best choice for all but very small inputs.



## Asymptotic Notation: informal introduction

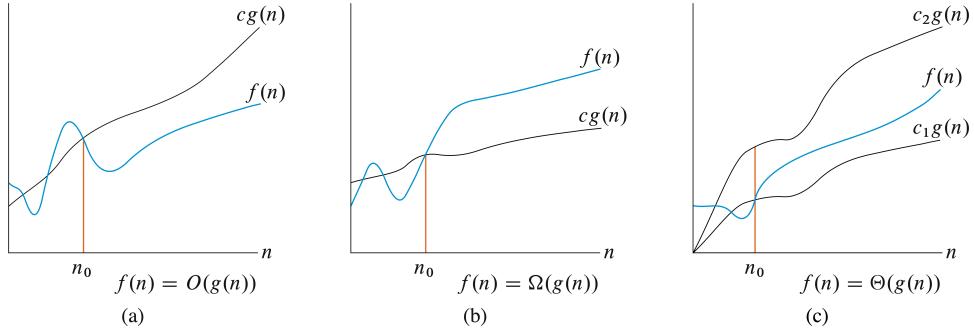
- **$O$ -notation** (big-Oh): The  $O$ -notation characterizes an upper bound on the asymptotic behavior of a function. It says that a function grows no faster than a certain rate, based upon its highest order term. Example:  $7n^3 + 100n^2 - 20n + 6$  would be  $O(n^3)$ . It is also  $O(n^4)$ .
- **$\Omega$ -notation** (big-Omega) The  $\Omega$ -notation characterizes a lower bound on the asymptotic behavior of a function. It says that a function grows at least as fast as a certain rate, based (again) upon its highest order term. Example:  $7n^3 + 100n^2 - 20n + 6$  would be  $\Omega(n^3)$ . It is also  $\Omega(n^2)$ , and  $\Omega(n)$ .
- **$\Theta$ -notation** (big-Theta) The  $\Theta$ -notation characterizes a tight bound on the asymptotic behavior of a function. It says that a function grows precisely at a certain rate, based on its highest

order term. Example:  $7n^3 + 100n^2 - 20n + 6$  would be  $\Theta(n^3)$ . However, it isn't  $\Theta(n^2)$  or  $\Theta(n^4)$ .

**Note:** If a function is both  $O(f(n))$  and  $\Omega(f(n))$  for some function  $f(n)$ , then we have shown that the function is  $\Theta(f(n))$ .

**Example:** Do an informal asymptotic analysis of insertion sort to show that its worst-case running time is  $(n^2)$  and  $\Omega(n^2)$  and thus is  $\Theta(n^2)$ .

## Asymptotic Notation: formal definition



- **$O$ -notation: asymptotically upper bound**

- *Definition:* For a given function  $g(n)$ , we denote by  $O(g(n))$  to be the *set of functions*:

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c, \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}$$

- *Alternative Definition:* for two given functions  $f(n)$  and  $g(n)$ :

$$f(n) = O(g(n)) \iff \lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c \text{ or } \infty$$

- We can use the alternative definition to show

$$\begin{aligned} f(n) &= \frac{1}{2}n^2 - 4n + 100 = O(n^2) \quad \text{since } \lim_{n \rightarrow \infty} \frac{n^2}{\frac{1}{2}n^2 - 4n + 100} = 2 \\ &= O(n^3) \quad \text{since } \lim_{n \rightarrow \infty} \frac{n^3}{\frac{1}{2}n^2 - 4n + 100} = \infty \\ &= O(\infty) \quad \text{since } \lim_{n \rightarrow \infty} \frac{\infty}{\frac{1}{2}n^2 - 4n + 100} = \infty \end{aligned}$$

- Another example:

$$f(n) = 10 + \frac{1}{n} = O(1) \text{ since } \lim_{n \rightarrow \infty} \left(10 + \frac{1}{n}\right) = 10 \quad (1)$$

(2)

- **$\Omega$ -notation: asymptotically lower bound**

- *Definition:* For a given function  $g(n)$ , we denote by  $\Omega(g(n))$  to be the *set of functions*:

$$\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c, \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}$$

- *Alternative Definition:* for two given functions  $f(n)$  and  $g(n)$ :

$$f(n) = \Omega(g(n)) \iff \lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c \text{ or } 0$$

- We can use the definition to show

$$\begin{aligned} f(n) &= \frac{1}{2}n^2 - 4n + 100 = \Omega(n^2) \quad \text{since } \lim_{n \rightarrow \infty} \frac{n^2}{\frac{1}{2}n^2 - 4n + 100} = 2 \\ &= \Omega(n) \quad \text{since } \lim_{n \rightarrow \infty} \frac{n}{\frac{1}{2}n^2 - 4n + 100} = 0 \\ &= \Omega(1) \quad \text{since } \lim_{n \rightarrow \infty} \frac{1}{\frac{1}{2}n^2 - 4n + 100} = 0 \end{aligned}$$

- For example:

$$\begin{aligned} \text{INSERTION-SORT}(n) &= \Omega(1) \\ &= \Omega(n) \\ &\neq \Omega(n^2), \text{ since best-case takes time proportional to } n \end{aligned}$$

- **$\Theta$ -notation: asymptotically tight bound**

- *Definition:* For a given function  $g(n)$ , we denote by  $\Theta(g(n))$  to be the *set of functions*:

$$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ for all } n \geq n_0\}$$

- *Alternative Definition:* For two given functions  $f(n)$  and  $g(n)$ :

$$f(n) = \Theta(g(n)) \iff \lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c$$

- We can use the definition to show  $f(n) = \frac{1}{2}n^2 - 4n + 100 = \Theta(n^2)$ , since

$$\lim_{n \rightarrow \infty} \frac{n^2}{\frac{1}{2}n^2 - 4n + 100} = 2$$

- Without using definition, we can determine the  $\Theta$ -notation (as well as other notations) of a function  $f(n)$  by throwing away lower-order terms and ignores the leading coefficient of the highest-order term.

For example,

$$f(n) = \frac{1}{2}n^2 - 4n + 100 = \Theta(n^2)$$

- People often use big-O notation to describe the running time of an algorithm rather than using  $\Theta$ -notation (but they often mean to use  $\Theta$ ). However,  $\Theta$ -notation tells people more exact running time of an algorithm. For example:

`INSERTION-SORT(A) =  $O(n^2)$`  for all input  $n$

$$\begin{aligned}\text{INSERTION-SORT(A)} &\neq \Theta(n^2) \\ &\neq \Theta(n)\end{aligned}$$

The worst case of `INSERTION-SORT(A) =  $\Theta(n^2)$`

- *Theorem: for any two functions  $f(n)$  and  $g(n)$ ,*  
 $f(n) = \Theta(g(n)) \implies f(n) = O(g(n))$  and  $f(n) = \Omega(g(n))$

- **Properties of asymptotics**

- **Transitivity**

$f(n) = \Theta(g(n))$  and  $g(n) = \Theta(h(n))$  imply  $f(n) = \Theta(h(n))$

$f(n) = O(g(n))$  and  $g(n) = O(h(n))$  imply  $f(n) = O(h(n))$

$f(n) = \Omega(g(n))$  and  $g(n) = \Omega(h(n))$  imply  $f(n) = \Omega(h(n))$

- **Reflexivity**

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

$f(n) = O(f(n))$

$f(n) = \Omega(f(n))$

- **Symmetry**

$f(n) = \Theta(g(n))$  if and only if  $g(n) = \Theta(f(n))$

- **Transpose Symmetry**

$f(n) = O(g(n))$  if and only if  $g(n) = \Omega(f(n))$

- **Trichotomy:** For any two numbers  $a$  and  $b$ , exactly one of the following must hold:

$a < b, a = b, a > b$ . Not all functions are asymptotically comparable. For example:  $n$  and  $n^{1+\sin n}$  cannot be asymptotically compared.

- Based on the above properties, we can draw an analogy to comparing two numbers  $a$  and  $b$ :

$f(n) = O(g(n))$  is similar to  $a \leq b$

$f(n) = \Omega(g(n))$  is similar to  $a \geq b$

$f(n) = \Theta(g(n))$  is similar to  $a = b$

- **Asymptotic notations in equations**

- $2n^2 + 3n + 1 = 2n^2 + \Theta(n) \implies 2n^2 + 3n + 1 = 2n^2 + f(n)$ , where  $f(n) = \Theta(n)$ . In this case,  $f(n) = 3n + 1$

This example shows that we can mix asymptotics in equations. For example, we replaced lower order terms above by  $\Theta(n)$  so we could focus on the more important terms.

- $T(n) = O(n) + \Theta(n) + \Omega(n) \implies T(n) = \Omega(n)$

This example shows how to combine three asymptotic components in an equation.  $T(n)$  consists of the sum of the following three terms:

- \*  $O(n)$  which is growing at a rate  $\leq n$ ,
- \*  $\Theta(n)$  so growing at the same rate as  $n$ , and
- \*  $\Omega(n)$  so growing at a rate greater than  $n$ .

If we combine those three, the best we can say is  $\Omega(n)$  since that will be the dominant term for  $T(n)$ . This is because something growing faster than  $n$  will be growing bigger than two other terms growing at a rate equal to or less than  $n$ .

## Categories of functions

growth rate	slowest	$\rightarrow$	$\rightarrow$	$\rightarrow$	fastest
run time	fastest	$\leftarrow$	$\leftarrow$	$\leftarrow$	slowest
categories	constant	logarithms	polynomials	exponentials	super exponentials
examples	5 1 10000	$\log_2 n$ $\log_{10} n$ $100\log_e n$	$n^2$ $n^3$ $n^{0.1}$	$2^{n/2}$ $2^n$ $3^n$	$(\log n)^n$ $n!$ $n^n$

- For logarithm functions, the base does not matter.  $\log_{10} n = \Theta(\log_2 n)$  or  $\log_e n = \Theta(\log_2 n)$
- For exponential functions, the base matters.  $2^{n/2} = \sqrt{2^n} = O(2^n)$ , but  $2^{n/2} \neq \Theta(2^n)$

- Comparison between polynomials and exponentials:  $n^a$  and  $b^n$

No matter how big the  $a$  is and how small the  $b > 1$  is, the exponential  $b^n$  will always out-grow the polynomial  $n^a$  if  $n$  approaches  $\infty$ .

For example,  $1.000001^n$  will out-grow  $n^{1,000,000}$  when  $n \rightarrow \infty$

- **Review Section 3.3 from the textbook:** Discrete mathematics that is useful for analysis. Floors and ceilings, Modular arithmetic, Polynomials, Exponentials, Logarithms, Factorials (pay attention to Stirling's Approximation), and Fibonacci numbers.

- **Exponentials:** For all real  $a > 0$ ,  $m$ , and  $n$

$$\begin{aligned}
 a^0 &= 1 \\
 a^1 &= a \\
 a^{-1} &= \frac{1}{a} \\
 (a^m)^n &= a^{mn} \\
 (a^m)^n &= (a^n)^m
 \end{aligned}$$

$$a^m a^n = a^{m+n}$$

- **Logarithms:** Useful properties to remember:

$$\begin{aligned}\log(ab) &= \log a + \log b \\ \log\left(\frac{a}{b}\right) &= \log a - \log b \\ \log(a^b) &= b \log a\end{aligned}$$

- Logarithms and Exponentials are inverse functions:

$$\begin{aligned}b^{\log_b a} &= a \\ \log_b(b^a) &= a\end{aligned}$$

- **Change of base formula:**

$$\log_b a = \frac{\log_k a}{\log_k b}$$

This shows that logarithms in different bases differ only by a constant factor.

- We often use the base-2 logarithm in computer science since computers are binary.

$$\begin{aligned}\lg n &= \log_2 n \\ \ln n &= \log_e n \\ \log n &= \log_{10} n \\ \log^k n &= (\log n)^k \\ \lg \lg n &= \lg(\lg n)\end{aligned}$$

- **Factorials and Stirling's Approximation:**

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

- This allows us to show that  $\lg(n!) = \Theta(n \lg n)$ , which is useful in the analysis of several algorithms.
- **Review Calculus:** The derivative rule and *L'Hôpital's rule* are useful for obtaining limits of more complex functions.

- **Summations:** Arithmetic series and Geometric series sums are commonly found during analysis of algorithms.

- **Simple Arithmetic series:**

$$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$

Here is a common variation:

$$1 + 2 + 3 + \dots + n - 1 = \frac{n(n-1)}{2}$$

- **General Arithmetic Series:** Each term  $a_i$  is separated from the previous term  $a_{i-1}$  by a constant.

$$a_1 + a_2 + \dots + a_n = \frac{(a_1 + a_n)n}{2}$$

- **Simple Geometric Series:**

$$1 + 2 + 2^2 + 2^3 + \dots + 2^n = 2^{n+1} - 1$$

Here is a common variation:

$$1 + 2 + 2^2 + 2^3 + \dots + 2^{n-1} = 2^n - 1$$

- **General Geometric Series:**

For  $x \neq 1$ , we have:

$$1 + x + x^2 + x^3 + \dots + x^n = \frac{x^{n+1} - 1}{x - 1}$$

If  $|x| < 1$ , then the sum is  $\frac{1}{x-1}$ .

## Practice Problems

- Exercise 3.2-3. Is  $2^{n+1} = O(2^n)$ ? Is  $2^{2n} = O(2^n)$ ?
- Order the following functions by growth rate:  $n, n^2, \sqrt{n}, n^{1.5}, 2^n, n!, (\log n)^2, 3^n, \log n^2$ .

## Math review

- Derivative rules: <https://www.mathsisfun.com/calculus/derivatives-rules.html>
- *L'Hôpital's rule*: <https://www.mathsisfun.com/calculus/l-hopitals-rule.html>