



# Functions, Complexity of Algorithms



## **Section 2.2 The Growth of Functions**

# Topics

---

- Big-O Definition
- Big-O by little-O
- Complexity Classes
- Properties and theorems of Big-O
- Big Omega and Big Theta

# Overview

---

- What really matters in comparing the complexity of algorithms?
  - We only care about the behavior for *large* problems.
  - Even bad algorithms can be used to solve small problems.
  - Ignore implementation details such as loop counter increment, etc. We need to measure the order of complexity.
- We quantify the concept that  $g$  *grows at least as fast as*  $f$ .

# Overview

- Example: algorithm Performance Estimating Running Time for Input of Size  $n$

If running time grows like:	then doubling input size does following to run time	For example
$n$	doubles	1 hr. → 2 hrs.
$n^2$	multiplies by 4	1 hr. → 4 hrs.
$\log_2 n$	increase by small constant	1 hr. → 1 hr. 1 min.
$2^n$	squares it	1 hr. → 60 hrs.

# The Big-O Notation

---

- **Purpose:** to describe the growth rate of a function.

e.g. does it grow like  **$\log n$** ? like  **$n$** ? like  **$n^2$** ?

**Examples:**

**$2n^2 + 3n + 1$**  “grows like”  **$n^2$** .

**$0.5(n \log n) - 3n + 7$**  “grows like”  **$n \log n$** .

Big-O is used to denote an upper bound on growth rate:  **$f(n)$  grows no faster than  $g(n)$**

# The Big-O Definition

- **Formal definition**

$f(n)$  is  $O(g(n))$

if there is a constant  $C > 0$  and a constant  $k$ , such that for all  $n > k$ ,

$f(n) \leq Cg(n)$

In a better format:

- **Definition:** Let  $f$  and  $g$  be functions from  $\mathbb{N}$  to  $\mathbb{R}$ . Then ' $f$  is order  $g$ ', denoted  $f$  is  $O(g)$  or ' $f$  is big-O of  $g$ ,' iff

$$\exists k \exists C \forall n [n > k \rightarrow |f(n)| \leq C |g(n)|]$$

Note:

- Choose  $k$
- Choose  $C$ ; it may depend on your choice of  $k$
- Practically, the function  $g$  is chosen to be as small as possible.

# Big-O by Definition

---

- Example 1:  $f(n) = n + 3$

$$0 \leq n + 3 \leq n + n = 2n$$

for  $n > 3$ .

Therefore,  $n + 3$  is  $O(n)$ .

- Example 2:  $f(n) = 4n^2 + n$

$$0 \leq 4n^2 + n \leq 4n^2 + n^2 = 5n^2$$

for  $n > 1$ .

Therefore  $4n^2 + n$  is  $O(n^2)$ .

# Big-O by Definition

---

- Example 3: Show that  $n^3$  is not  $O(100n^2)$ .

If  $n^3$  is  $O(100n^2)$  then there are constants  $C$  and  $k$  such that

$$n^3 \leq 100Cn^2$$

for  $n > k$ . Then

$$n \leq 100C$$

for all  $n > k$ . This is a contradiction since  $n$  grows without bound.

# Big-O by Little-O

■ **Definition:** if  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$

then  $f$  is  $o(g)$  (called *little-o* of  $g$ )

■ **Theorem:** If  $f$  is  $o(g)$  then  $f$  is  $O(g)$ .

It is usually easier to prove  $f$  is  $o(g)$

- using the theory of limits
- using L'Hospital's rule

An alternative for those with a calculus background.

# Examples

- Example 1:  $3n + 5$  is  $O(n^2)$

Proof: It's easy to show

$$\lim_{n \rightarrow \infty} \frac{3n + 5}{n^2} = 0$$

Hence  $3n + 5$  is  $o(n^2)$  and so it is  $O(n^2)$ .

Q. E. D.

- Example 2:  $7n^2$  is  $O(n^2)$
- Example 3:  $7n^2$  is  $O(n^3)$
- Example 4:  $\sqrt{n}$  is  $O(n)$

# Complexity Classes

- Note that  $O(g)$  is a set called a **complexity class**. It contains all the functions which  $g$  dominates.

- Important Complexity Classes

$$O(1) \subseteq O(\log n) \subseteq O(n) \subseteq O(n \log n) \subseteq O(n^2) \\ \subseteq O(n^j) \subseteq O(c^n) \subseteq O(n!)$$

where  $j > 2$  and  $c > 1$ .

# Properties of Big-O

■ Theorem:  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$  is  $O(x^n)$

■ Part 1: the set  $O(g)$  is closed under addition :

If  $f$  is  $O(g)$  and  $h$  is  $O(g)$  then  $f + h$  is  $O(g)$

■ Example:

$$f_1(x) = 2x^2 \quad (O(x^2))$$

$$f_2(x) = x^2 + 2x + 1 \quad (O(x^2))$$

$$f_3(x) = x + 5 \quad (O(x^2))$$

$$f_1(x) + f_2(x) + f_3(x):$$

# Properties of Big-O

- Part 2: the set  $O(g)$  is closed under multiplication by a scalar  $\alpha$  (real number):

If  $f$  is  $O(g)$  then  $\alpha f$  is  $O(g)$

- Example:

$$f_1=2x^2 \text{ (} O(x^2)\text{)}, f_2=6x^2: O(?)$$

$$g_1=x \text{ (} O(x)\text{)}, g_2=100x: O(?)$$

# Big-O Theorems

- Theorem: If  $f_1$  is  $O(g_1)$  and  $f_2$  is  $O(g_2)$  then
  - i)  $f_1 f_2$  is  $O(g_1 g_2)$
  - ii)  $f_1 + f_2$  is  $O(\max\{g_1, g_2\})$
- Examples:
  - 1.  $f_1 = 2n^2$ ,  $f_2 = 3n$ 
    - $f_1 f_2$ :  $O(?)$
    - $f_1 + f_2$ :  $O(?)$
  - 2.  $f_1 = 5 \log n$ ,  $f_2 = 5$ 
    - $f_1 f_2$
    - $f_1 + f_2$

# Big-O Theorems

■ Proof of i): There is a  $k_1$  and  $C_1$  such that

1.  $f_1(n) < C_1 g_1(n)$  when  $n > k_1$ .

There is a  $k_2$  and  $C_2$  such that

2.  $f_2(n) < C_2 g_2(n)$  when  $n > k_2$ .

We must find a  $k_3$  and  $C_3$  such that

3.  $f_1(n)f_2(n) < C_3 g_1(n)g_2(n)$  when  $n > k_3$ .

We use the inequality

if  $0 < a < b$  and  $0 < c < d$  then  $ac < bd$

to conclude that

$$f_1(n)f_2(n) < C_1 C_2 g_1(n)g_2(n)$$

as long as  $k > \max\{k_1, k_2\}$  so that both inequalities 1 and 2 hold at the same time.

Therefore, choose

$$C_3 = C_1 C_2$$

and

$$k_3 = \max\{k_1, k_2\}.$$

Q. E. D.

# Big-O Estimate Examples

- Example 1: Find Big-O (complexity class) of the function

$$f(n) = 3n^2 + 15n$$

- *Example 2:* Find Big-O (complexity class) of the function

$$f(n) = (3n^2 + 15n)(n+1)^3$$

- *Example 3:* Find the complexity class of the function

$$(nn! + 3^{n+2} + 3n^{100})(n^n + n2^n)$$

If a flop takes a nanosecond, how long it takes to solve a problem for  $n=50$ .

# Big Omega and Big Theta

- **Big O, Big Omega and Big Theta**

- Upper bound: Big-O

- Lower bound: Big-Omega

- Both upper and lower bound: Big-Theta

- **The Big-Omega Definition:** Let  $f$  and  $g$  be functions from  $\mathbb{N}$  to  $\mathbb{R}$ . Then  $f$  is Big-Omega of  $g$ , denoted  $\Omega(g)$ , iff  $\exists k \exists C \forall n [n > k \rightarrow |f(n)| \geq C |g(n)|]$

- **The Big-Theta Definition:** Let  $f$  and  $g$  be functions from  $\mathbb{N}$  to  $\mathbb{R}$ . Then  $f$  is Big-Theta of  $g$ , denoted  $\Theta(g)$ , if  $f$  is  $O(g)$  and  $f$  is  $\Omega(g)$ .

# Big Omega and Big Theta

---

- $f_1(n)=7n^2$ ,  $f_2(n)=7n^3$ ,  $g(n)=n^2$ ; Specify if  $f_1(n)$ ,  $f_2(n)$  is Big-O/ $\Omega$ / $\Theta$  of  $g$ .

	$O(n^2)$	$\Omega(n^2)$	$\Theta(n^2)$
$f_1(n)$			
$f_2(n)$			