

# Algorithms

Dr. Khalil Yousef

Lecture 2:

Reading Assignment:  
Read Chapter 3 of *the book*

# Course Learning Outcomes

**Analyze numerical computations algorithms**  
(e.g. matrix multiplication).

# Outline

- Big-Oh and Other Notations in Algorithm Analysis
  - Classifying Functions by Their Asymptotic Growth
  - Theta, Little oh, Little omega
  - Big Oh, Big Omega
  - Rules to manipulate Big-Oh expressions
  - Typical Growth Rates

# Classifying Functions by Their Asymptotic Growth

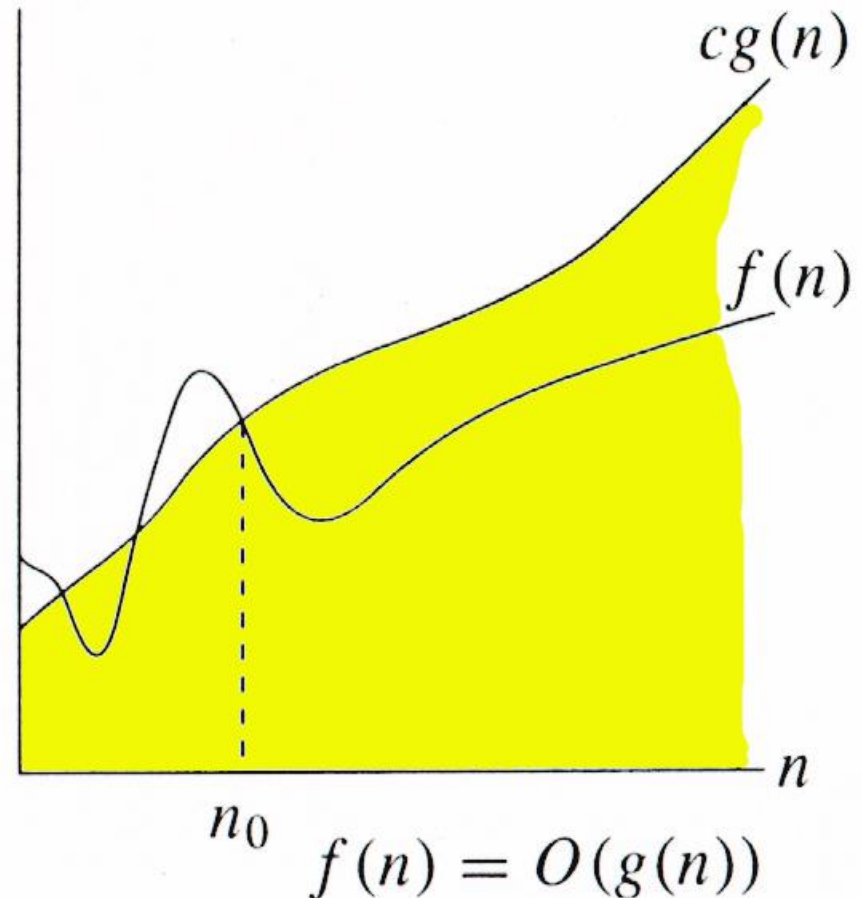
- Asymptotic growth:
  - The rate of growth of a function
- Given a particular differentiable function  $f(n)$ , all other differentiable functions fall into three classes:
  - Growing with the same rate
  - Growing faster
  - Growing slower

# The Big-Oh Notation: O-notation

- For a given function  $g(n)$ , we denote by  $O(g(n))$  the set of functions
- $O(g(n)) = \{f(n): \text{there exist positive constants } c \text{ and } n_0 \text{ such that}$

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

for all  $n \geq n_0$



We say  $g(n)$  is an *asymptotic upper bound* for  $f(n)$   
Or we say  $f(n)$  grows with **same rate or slower** than  $g(n)$ .

# Important Notes

- A very useful aspect of this asymptotic notation (*Big-Oh* and others) is that constants and lower-order terms can be ignored.
  - Example
    - $5n^2+100n+22 = O(n^2)$  and  $n = O(n^2)$
- The worst-case running time of INSERTION-SORT is  $O(n^2)$ , but this does not imply that there is a  $O(n^2)$  bound on every input. For example, on sorted input INSERTION-SORT runs in linear time.
- Though we write  $f(n) = O(g(n))$ , technically this means  $f(n) \in O(g(n))$ .

# Asymptotic Notation in Equations

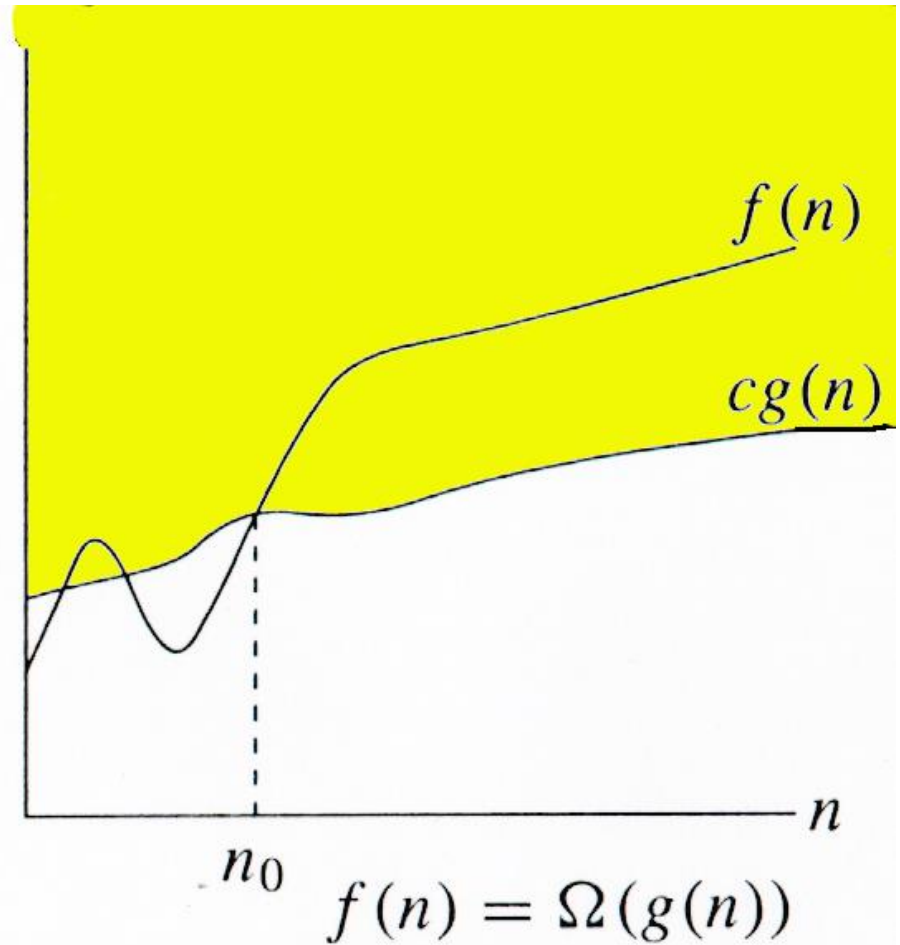
- When asymptotic notation appears alone on the right-hand side of an equation, as in  $f(n) = O(n^2)$ , we are indicating that  $f(n) \in O(n^2)$ .
- When it appears in a formula, we interpret it as standing for *some anonymous function that shall remain nameless*.
- For example,  $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$  means:  $2n^2 + 3n + 1 = 2n^2 + f(n)$  where  $f(n)$  is some function in the set  $\Theta(n)$ .
- By using this mechanism, we can eliminate clutter in equations.
- What if it occurs on both sides? As in  $2n^2 + 3n + \Theta(1) = 2n^2 + \Theta(n)$ . We take this to mean for any  $f(n) \in \Theta(1)$  there is some  $g(n) \in \Theta(n)$  such that  $2n^2 + 3n + f(n) = 2n^2 + g(n)$ .

# The Big-Omega Notation: $\Omega$ -notation

- For a given function  $g(n)$ , we denote by  $\Omega(g(n))$  the set of functions
- $\Omega(g(n)) = \{f(n): \text{there exist positive constants } c \text{ and } n_0 \text{ such that}$

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

for all  $n \geq n_0$  }



We say  $g(n)$  is an *asymptotic lower bound* for  $f(n)$

Or we say  $f(n)$  grows with **same rate or faster** than  $g(n)$ .



# The Big-Omega Notation: $\Omega$ -notation

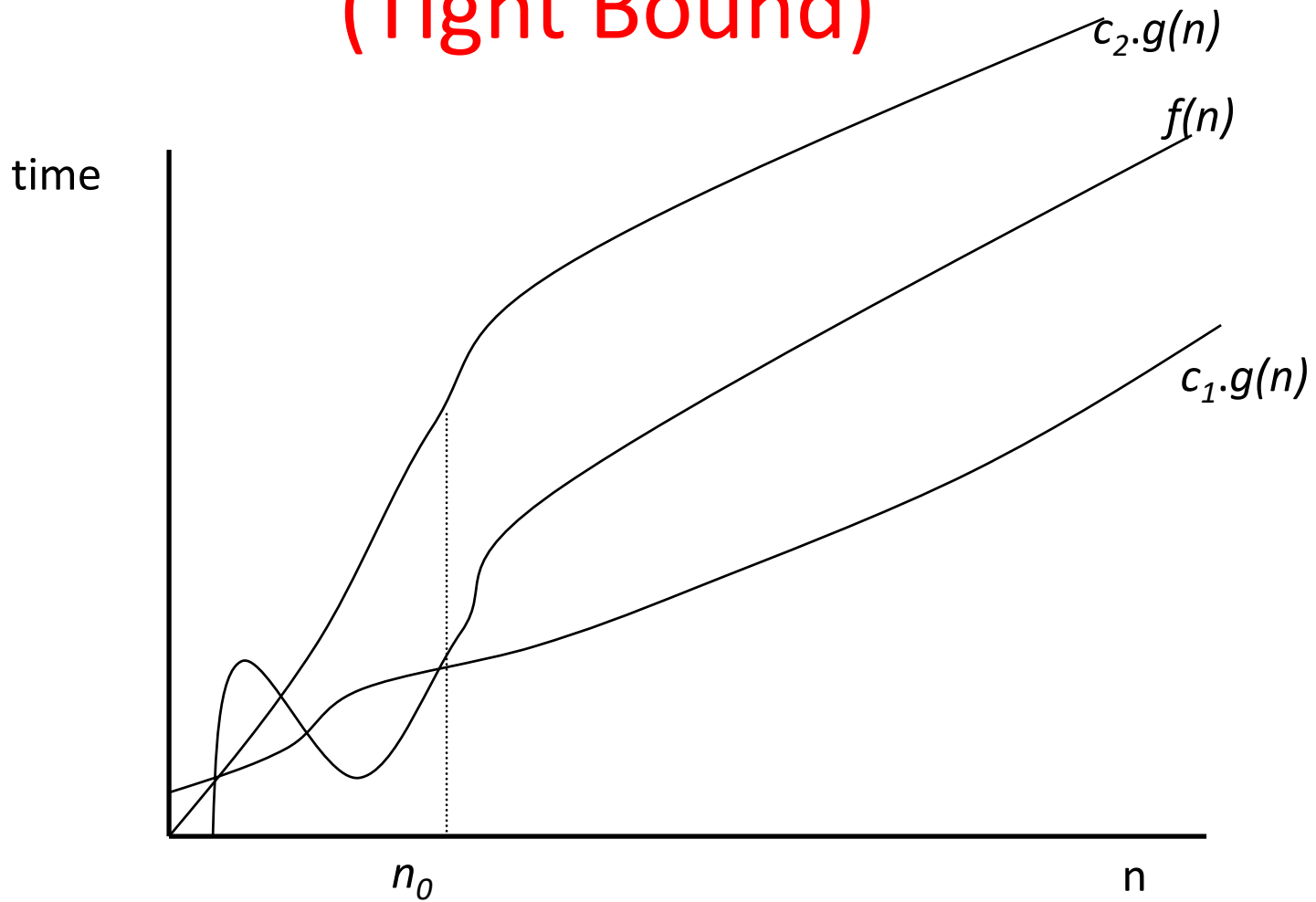
- Example:

- $5n^2+100n+22 = \Omega(n^2)$  and  $n^2 = \Omega(n)$ .

# $\Theta$ notation (Theta) (Tight Bound)

- In some cases,
  - If  $f(n) = O(g(n))$  and  $f(n) = \Omega(g(n))$
  - This means, that the worst and best cases require the same amount of time  $t$  within a constant factor
  - In this case we use a new notation called “theta  $\Theta$ ”
  - “theta  $\Theta$ ” represents an asymptotically tight bound
- For a given function  $g(n)$ , we denote by  $\Theta(g(n))$  the set of functions
  - $\Theta(g(n)) = \{f(n): \text{there exist positive constants } c_1 > 0, c_2 > 0 \text{ and } n_0 > 0 \text{ such that}$ 
    - $c_1 g(n) \leq f(n) \leq c_2 g(n) \forall n \geq n_0\}$

# $\Theta$ notation (Theta) (Tight Bound)



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

# Discussion of the Asymptotic Notation

- **Example 1:** prove that  $6n^3 \neq \Theta(n^2)$ .

Proof by contradiction: i.e., assume  $6n^3 = \Theta(n^2)$ .

$$0 \leq c_1 n^2 \leq 6n^3 \leq c_2 n^2, \forall n \geq n_0$$

$$0 \leq c_1 \leq 6n \leq c_2, \forall n \geq n_0$$

This implies that  $n \leq \frac{c_2}{6}, \forall n \geq n_0$ , a contradiction.

# Discussion of the Asymptotic Notation

- Example 2:

$$f(n) = 5n^2 + 1000n$$

$$\text{Claim: } f(n) = \Theta(n^2)$$

Needed:  $c_1, c_2$ , and  $n_0$ , such that:

$$0 \leq c_1 n^2 \leq 5n^2 + 1000n \leq c_2 n^2$$

$$0 \leq c_1 \leq 5 + \frac{1000}{n} \leq c_2$$

One choice:  $n_0 = 1000, c_1 = 5, c_2 = 6$

# Discussion of the Asymptotic Notation

- **Example 3:** Let us try to prove that  $n \neq \Theta(n^2)$ .

Proof by contradiction: i.e., assume  $n = \Theta(n^2)$ .

$$0 \leq c_1 n^2 \leq n \leq c_2 n^2, \forall n \geq n_0$$

$$0 \leq c_1 \leq \frac{1}{n} \leq c_2, \forall n \geq n_0$$

This implies that  $n \leq \frac{1}{c_1}, \forall n \geq n_0$ , a contradiction.

# Little oh Notation: o-notation

- o-notation denotes an upper bound that is not asymptotically tight (it is certain to grow faster). In contrast,  $O$  may or may not be asymptotically tight.
- For a given function  $g(n)$ , we denote by  $o(g(n))$  the set of functions:
- $o(g(n)) = \{f(n): \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0\}$
- $f(n)$  becomes insignificant relative to  $g(n)$  as  $n$  approaches infinity:  
i.e.  $f(n) = o(g(n))$  iff  $\lim_{n \rightarrow \infty} [f(n) / g(n)] = 0$
- We say  $g(n)$  is an *upper bound* for  $f(n)$  that is *not asymptotically tight*.
  - For example,  $2n = o(n^2)$ , but  $2n^2 \neq o(n^2)$ .

# $O(*)$ versus $o(*)$

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

$o(g(n)) = \{f(n): \text{for **any** positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0\}.$

Thus  $o(f(n))$  is a weakened  $O(f(n))$ .

For example:  $n^2 = O(n^2)$

$$n^2 \neq o(n^2)$$

$$n^2 = O(n^3)$$

$$n^2 = o(n^3)$$



# Little omega Notation: $\omega$ -notation

- For a given function  $g(n)$ , we denote by  $w(g(n))$  the set of functions:
- $\omega(g(n)) = \{f(n): \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq cg(n) < f(n) \text{ for all } n \geq n_0\}$
- $f(n)$  becomes insignificant relative to  $g(n)$  as  $n$  approaches infinity:  $f(n) = \omega(g(n))$  iff  $\lim_{n \rightarrow \infty} [f(n) / g(n)] = \infty$
- We say  $g(n)$  is an *lower bound* for  $f(n)$  *that is not asymptotically tight*.

For example,  $\frac{n^2}{2} = \omega(n)$ , but  $\frac{n^2}{2} \neq \omega(n^2)$

# Comparison of Asymptotic Functions

**Transitivity:**

Means AND

$$f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) \rightarrow f(n) = \Theta(h(n))$$

$$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \rightarrow f(n) = O(h(n))$$

$$f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n)) \rightarrow f(n) = \Omega(h(n))$$

$$f(n) = o(g(n)) \wedge g(n) = o(h(n)) \rightarrow f(n) = o(h(n))$$

$$f(n) = \omega(g(n)) \wedge g(n) = \omega(h(n)) \rightarrow f(n) = \omega(h(n))$$

**Reflexivity:**

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

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

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

**Symmetry:**

$$f(n) = \Theta(g(n)) \text{ iff } g(n) = \Theta(f(n))$$

# Comparison of Asymptotic Functions (Cont...)

## Transpose Symmetry:

$$f(n) = O(g(n)) \text{ iff } g(n) = \Omega(f(n))$$

$$f(n) = o(g(n)) \text{ iff } g(n) = \omega(f(n))$$

These properties allow us to draw an analogy between the asymptotic comparison of functions  $f$  and  $g$  and the comparison of real numbers  $a$  and  $b$ :

$$f(n) = O(g(n)) \approx a \leq b$$

$$f(n) = \Omega(g(n)) \approx a \geq b$$

$$f(n) = \Theta(g(n)) \approx a = b$$

$$f(n) = o(g(n)) \approx a < b$$

$$f(n) = \omega(g(n)) \approx a > b$$

Although two real numbers can be compared (using  $<$ ,  $=$ , or  $>$ ), not all functions are asymptotically comparable (e.g.,  $n$  and  $n^{1+\sin n}$  cannot be compared; the exponent of the second oscillates between 0 and 2).

# **SUMMARY**

# *The Big-Oh Notation*

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

if  $f(n)$  grows with **same rate** or **slower** than  $g(n)$ .

Means  $f(n) = \Theta(g(n))$  or  
 $f(n) = o(g(n))$

# *The Big-Omega Notation*

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

if  $f(n)$  grows with **same rate** or **faster** than  $g(n)$ .

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

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

# The Big-Omega Notation

- **The inverse of Big-Oh is  $\Omega$**
- If  $g(n) = O(f(n))$ ,
- then  $f(n) = \Omega(g(n))$

# Theta $\Theta$

- $f(n)$  and  $g(n)$  have same rate of growth, if
  - $\lim( f(n) / g(n) ) = c,$
  - $0 < c < \infty, \quad n \rightarrow \infty$
- Notation:  $f(n) = \Theta( g(n) )$
- Pronounced "theta"



# Theta: Relation of Equivalence

- **$\Theta$ : "having the same rate of growth":**
  - **Relation of equivalence**
  - Gives a partition over the set of all differentiable functions - classes of equivalence.
- **Functions in one and the same class are equivalent with respect to their growth.**

# *Little oh*

$f(n)$  grows slower than  $g(n)$   
(or  $g(n)$  grows faster than  $f(n)$ )

if

$$\lim( f(n) / g(n) ) = 0, \quad n \rightarrow \infty$$

Notation:  $f(n) = o( g(n) )$  pronounced "little oh"

# *Little omega*

$f(n)$  grows faster than  $g(n)$   
(or  $g(n)$  grows slower than  $f(n)$ )  
if

$$\lim( f(n) / g(n) ) = \infty, \quad n \rightarrow \infty$$

Notation:  $f(n) = \omega(g(n))$  pronounced "little omega"

# Little omega and Little oh

- if  $g(n) = \mathbf{o}(f(n))$
- then  $f(n) = \mathbf{\omega}(g(n))$
- **Examples:** Compare  $n$  and  $n^2$ 
  - $\lim(n/n^2) = 0, n \rightarrow \infty, n = \mathbf{o}(n^2)$
  - $\lim(n^2/n) = \infty, n \rightarrow \infty, n^2 = \mathbf{\omega}(n)$

# Examples

# Recall: Algorithms with Same Complexity

- Two algorithms have **same complexity**, if the functions representing the number of basic operations have **same rate of growth**.
- Among all functions with same rate of growth we **choose the simplest** one to represent the complexity.

# Example

- Compare  $n$  and  $(n+1)/2$ 
  - $\lim( n / ((n+1)/2 ) ) = 2,$
  - same rate of growth
- $(n+1)/2 = \Theta(n)$ 
  - Rate of growth of a linear function

# Example

- Compare  $n^2$  and  $n^2 + 6n$ 
  - $\lim( n^2 / (n^2 + 6n) ) = 1$
  - same rate of growth.
  - $n^2 + 6n = \Theta(n^2)$
  - Rate of growth of a quadratic function



# Example

- Compare  **$\log n$**  and  **$\log n^2$** 
  - $\lim( \log n / \log n^2 ) = \frac{1}{2}$
  - Same rate of growth.
  - **$\log n^2 = \Theta(\log n)$**
  - **logarithmic** rate of growth

# Example

**$\Theta(n^3)$ :**  $n^3$   
 $5n^3 + 4n$   
 $105n^3 + 4n^2 + 6n$

**$\Theta(n^2)$ :**  $n^2$   
 $5n^2 + 4n + 6$   
 $n^2 + 5$

**$\Theta(\log n)$ :**  $\log n$   
 $\log n^2$   
 $\log (n + n^3)$

# Example

$$\begin{aligned}n+5 &= \Theta(n) = O(n) = O(n^2) \\ &= O(n^3) = O(n^5)\end{aligned}$$

The closest estimation:  $n+5 = \Theta(n)$

**The general practice is to use**

**The Big-Oh notation:**

$$n+5 = O(n)$$

# Techniques to show that $f(n)$ is (not) $\Theta(n)$ , $O(n)$ , or $\Omega(n)$

1. Use the definition. For example, to show  $f(n) = O(n)$ , find positive constants  $c, n_0$  to solve  $0 \leq f(n) \leq cn$ .
2. Proof by contradiction (e.g., to show  $f(n) \neq O(n)$ ).
- 3a. If  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$  and  $f(n), g(n)$  are asymptotically non-negative ( $\geq 0$  for large  $n$ ), then  $f(n) = o(g(n))$ . This is a special case for  $o$ :  $g(n)$  is growing *much faster than*  $f(n)$ .
- 3b. If  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c, c \neq 0$  and  $f(n), g(n)$  are asymptotically non-negative ( $\geq 0$  for large  $n$ ), then  $f(n) = \Theta(g(n))$ .

# Techniques to show that $f(n)$ is (not) $\Theta(n)$ , $O(n)$ , or $\Omega(n)$

**3c.** If  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$  and  $f(n), g(n)$  are asymptotically non-negative ( $\geq 0$  for large  $n$ ), then  $f(n) = \omega(g(n))$ . This is a special case for  $\omega$ :  $f(n)$  is growing *much faster than*  $g(n)$ .

**4.** L'Hôpital's Rule: If  $\lim_{n \rightarrow \infty} f(n) = \lim_{n \rightarrow \infty} g(n) = \infty$ , then  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)}$ .

# Rules to manipulate Big-Oh expressions

## Rule 1:

a. If

$$T_1(N) = O(f(N))$$

and

$$T_2(N) = O(g(N))$$

then

$$T_1(N) + T_2(N) = \max( O( f(N) ), O( g(N) ) )$$

# Rules to manipulate Big-Oh expressions

**b.**  $f(n) + g(n) \neq \Theta(\min(f(n), g(n)))$

$$n^4 + n^2 \neq \Theta(n^2)$$

$$f(n) \neq O((f(n))^2)$$

$$f(n) = \frac{1}{n^2} \neq O\left(\frac{1}{n^4}\right)$$

$$f(n) \neq \Theta(f(n/2))$$

$$f(n) = 4^n \neq \Theta(4^{n/2})$$

# Rules to manipulate Big-Oh expressions

**C.** If

$$T_1(N) = O( f(N) )$$

and

$$T_2(N) = O( g(N) )$$

then

$$T_1(N) * T_2(N) = O( f(N) * g(N) )$$



# Rules to manipulate Big-Oh expressions

## Rule 2:

If  $T(N)$  is a polynomial of degree  $k$ ,  
then

$$T(N) = \Theta(N^k)$$

## Rule 3:

$\log^k N = O(N)$  for any constant  $k$ .

# Examples

- $n^2 + n = O(n^2)$ 
  - we disregard any lower-order term
- $n \log(n) = O(n \log(n))$
- $n^2 + n \log(n) = O(n^2)$

# Standard Notations: Monotonicity

**Monotonicity** of a function:

A function is monotonically increasing if  $m \leq n \rightarrow f(m) \leq f(n)$ .

A function is monotonically decreasing if  $m \leq n \rightarrow f(m) \geq f(n)$ .

A function is strictly increasing if  $m < n \rightarrow f(m) < f(n)$ .

A function is strictly decreasing if  $m < n \rightarrow f(m) > f(n)$ .

# Standard Notations: Floors and Ceilings

For any real number  $x$ , the greatest integer less than or equal to  $x$  is called the floor of  $x$ , denoted  $\lfloor x \rfloor$ .

For any real number  $x$ , the least integer greater than or equal to  $x$  is called the ceiling of  $x$ , denoted  $\lceil x \rceil$ .

For all real  $x$ ,  $x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$ .

For any integer  $n$ ,  $\lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil = n$ .

For any integer  $n$  and integers  $a \neq 0, b \neq 0$ :

$$\begin{aligned}\lceil \frac{\lceil \frac{n}{a} \rceil}{b} \rceil &= \lceil \frac{n}{ab} \rceil \\ \lfloor \frac{\lfloor \frac{n}{a} \rfloor}{b} \rfloor &= \lfloor \frac{n}{ab} \rfloor\end{aligned}$$

# Polynomials

Given a positive integer  $d$ , a polynomial in  $n$  of degree  $d$  is a function  $p(n)$  in the form:

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

where  $a_0, a_1 \dots a_n$  are called coefficients, and  $a_d \neq 0$ .

For an asymptotically positive polynomial of degree  $d$  (i.e.,  $a_d > 0$ ),  $p(n) = \Theta(n^d)$ .

We say that  $f(n)$  is polynomially bounded if  $f(n) = n^{O(1)}$ , which is equivalent to  $f(n) = O(n^k)$  for constant  $k$ .

# Exponentials

$$a^0 = 1$$

$$a^1 = a$$

$$a^{-1} = \frac{1}{a}$$

$$(a^m)^n = a^{mn}$$

$$(a^m)^n = (a^n)^m$$

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

It is useful to know some properties of the special exponential  $e^x$ .

For example, for all real  $x$ ,  $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$ .

For all real  $x$ ,  $e^x \geq 1 + x$ , where equality holds only if  $x = 0$ .

When  $|x| \leq 1$ ,  $1 + x \leq e^x \leq 1 + x + x^2$ .

$$\forall x, \lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x$$

# Logarithms

$$\lg n = \log_2 n \text{ (binary logarithm)}$$

$$\ln n = \log_e n \text{ (natural logarithm)}$$

$$\lg^k n = (\lg n)^k \text{ (exponentiation)}$$

$$\lg \lg n = \lg(\lg n) \text{ (composition)}$$

For all real  $a > 0, b > 0, c > 0$ :

$$a = b^{\log_b a}$$

$$\log_c(ab) = \log_c a + \log_c b$$

$$\log_b a^n = n \log_b a$$

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

$$\log_b\left(\frac{1}{a}\right) = -\log_b a$$

$$\log_b a = \frac{1}{\log_a b}$$

$$a^{\log_b n} = n^{\log_b a}$$

# Logarithms

- Show that  $a^{\log_b n} = n^{\log_b a}$

$$\text{let } a = b^{\log_b a}$$

$$\therefore a^{\log_b n} = \left(b^{\log_b a}\right)^{\log_b n} = b^{(\log_b a)(\log_b n)} = b^{(\log_b n)(\log_b a)} = a^{\log_b n} = \left(b^{\log_b n}\right)^{\log_b a} = n^{\log_b a}$$



# Logarithms (Cont...)

Note that  $\lg n + k = (\lg n) + k$ .

Since changing a log base only changes a value by a constant factor, we will usually use  $\lg n$  when we don't care about constant factors.

Note, when  $|x| < 1$ :

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots$$

For  $x > -1$ ,  $\frac{x}{1+x} \leq \ln(1+x) \leq x$ , where the equality holds if  $x = 0$ .

A function is polylogarithmically bounded if  $f(n) = \lg^{O(1)} n$ .

# Rates of Growth

The rates of growth of any positive exponential function is faster than any polynomial function, as the following shows.

For all real constants  $a, b$ , where  $a > 1$ ,  $\lim_{n \rightarrow \infty} \frac{n^b}{a^n} = 0$ , hence we can conclude  $n^b = o(a^n)$ .

The rates of growth of any polynomial function is faster than any polylogarithmic function, as the following shows.

By substituting  $\lg n$  for  $n$  and  $2^a$  for  $a$  in the above, we get:

$$\lim_{n \rightarrow \infty} \frac{\lg^b n}{(2^a)^{\lg n}} = \lim_{n \rightarrow \infty} \frac{\lg^b n}{n^a} = 0$$

Hence, we can conclude  $\lg^b n = o(n^a)$ , for any  $a > 0$ .

# Factorials

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ n(n-1)! & n > 0. \end{cases}$$

$$n! = \prod_{i=1}^n i$$

Intuition:  $n!$  is the number possible permutations of a given input set with  $n$  members. This is fast-growing!

A weak upper bound on the factorial function is  $n! \leq n^n$ .

Stirling's Approximation provides us with tighter upper and lower bounds.

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

$$\sqrt{2\pi n} \left(\frac{n}{e}\right)^n \leq n! \leq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}, n \geq 8$$

# The Iterated Logarithm Function

The notation  $\lg^* n$  is used to represent the iterated logarithm, which is an extremely slow growing function defined in terms of  $\lg^{(i)} n$ , which is a function defined on non-negative integers that applies the logarithm function  $i$  times in succession.

$$\lg^{(i)} n = \begin{cases} n & \text{if } i = 0, \\ \lg(\lg^{(i-1)} n) & \text{if } i > 0 \text{ and } \lg^{(i-1)} n > 0 \\ \text{undefined} & \text{if } i > 0 \text{ and } \lg^{(i-1)} n \leq 0 \text{ or } \lg^{(i-1)} \text{ is undefined} \end{cases}$$

Then:

$$\lg^* n = \min\{i \geq 0 : \lg^{(i)} n \leq 1\}$$

For example:

$$\lg^* 2 = 1$$

$$\lg^* 4 = 2$$

$$\lg^* 16 = 3$$

$$\lg^* 65536 = 4, \text{ etc.}$$

# Rank Ordering Functions by Order or Growth

To rank order a list of functions into an arrangement  $f_1, f_2, \dots, f_n$  and  $f_1 = \Omega(f_2), f_2 = \Omega(f_3), \dots, f_{n-1} = \Omega(f_n)$  (as well as identify those functions that belong to the same equivalence class, where  $f_1(n)$  and  $f_2(n)$  belong in the same equivalence class if and only if (iff)  $f_1(n) = \Theta(f_2(n))$ ), we can use what we know about the functions themselves and asymptotic notation. Much of the ranking is based on:

- Exponential functions grow faster than polynomial functions, which grow faster than polylogarithmic functions.
- The base of a logarithm does not matter asymptotically (recall  $\log_b a = \frac{\log_c a}{\log_c b}$ ), but the base of an exponential and the degree of a polynomial do matter.
- Identities can help, as can working with approximation formulas such as Stirling's approximation.

# Rank Ordering Functions by Order or Growth: Useful Identities

1.  $2^{\lg n} = n^{\lg 2} = n$

2.  $4^{\lg n} = n^{\lg 4} = n^2$

3.  $(\lg n)^{\lg n} = n^{\lg \lg n}$

4.  $2 = n^{\frac{1}{\lg n}}$  (raising identity 1 to the power  $\frac{1}{\lg n}$ )

5.  $(\sqrt{2})^{\lg n} = 2^{\frac{1}{2} \lg n} = 2^{\lg \sqrt{n}} = \sqrt{n}$

# Rank Ordering Functions by Order or Growth: More Useful Identities

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

2.  $n! = \Theta(n^{n+\frac{1}{2}}e^{-n})$  (drop constants and low order term)

3.  $(\lg n)! = \Theta((\lg n)^{\lg n + \frac{1}{2}}e^{-\lg n}) = \Theta((\lg n)^{\lg n + \frac{1}{2}}n^{-\lg e})$  (substitute  $\lg n$  for  $n$  in 2)

4.  $\lg(n!) = \Theta(n \lg n)$

5.  $n! = o(n^n)$

6.  $n! = \omega(2^n)$

# CLR 3-3: Rank Ordering Functions by Order or Growth: Examples

Ranking by asymptotic growth rate, equivalent classes are enclosed by '[ ]'.

$[1, n^{1/\lg n}]$	$(3/2)^n$
$\lg(\lg^* n)$	$2^n$
$[\lg^*(\lg n), \lg^*(n)]$	$n2^n$
$2^{\lg^* n}$	$e^n$
$\ln \ln n$	$n!$
$\sqrt{\lg n}$	$(n+1)!$
$\ln n$	$2^{2^n}$
$\lg^2 n$	$2^{2^{n+1}}$
$2^{\sqrt{2 \lg n}}$	
$(\sqrt{2})^{\lg n}$	
$2^{\lg n}$	
$[n \lg n, \lg(n!)]$	
$[4^{\lg n}, n^2]$	
$n^3,$	
$[(\lg n)!, n^{\lg \lg n}, (\lg n)^{\lg n}]$	



# Rank Ordering Functions by Order or Growth: Examples

Order the following functions:

1.  $(\lg n)^{\lg n}$  and  $n^3$

2.  $n^{\frac{1}{\lg n}}$  and  $n$

3.  $2^n$  and  $(\frac{3}{2})^n$

4.  $\lg^2 n$  and  $\ln n$

5.  $4^{\lg n}$  and  $8^{\lg n}$

6.  $(\lg n)!$  and  $\lg(n!)$

# Typical Growth Rates

$C$	constant, we write $O(1)$
$\log N$	logarithmic
$\log^2 N$	log-squared
$N$	linear
$N \log N$	
$N^2$	quadratic
$N^3$	cubic
$2^N$	exponential
$N!$	factorial

# Problems

- $N^2 = O(N^2)$   
– true
- $2N = O(N^2)$   
– true
- $N = O(N^2)$   
– true
  
- $N^2 = O(N)$   
– false
- $2N = O(N)$   
– true
- $N = O(N)$   
– true

# Problems

- $N^2 = \Theta(N^2)$ 
  - true
- $2N = \Theta(N^2)$ 
  - false
- $N = \Theta(N^2)$ 
  - false
  
- $N^2 = \Theta(N)$ 
  - false
- $2N = \Theta(N)$ 
  - true
- $N = \Theta(N)$ 
  - true

# Course Learning Outcomes

**Analyze** numerical computations algorithms  
(e.g. **matrix multiplication**).

# Matrix Multiplication

## Brute Force Alg.

### Counting Scalar Multiplications in Matrix Multiplication

---

MATRIX-MULTIPLY( $A, B$ )

1. **if**  $columns[A] \neq rows[B]$
2.   **then error** "incompatible dimensions"
3.   **else for**  $i \leftarrow 1$  to  $rows[A]$
4.       **do for**  $j \leftarrow 1$  to  $columns[B]$
5.           **do**  $C[i, j] \leftarrow 0$
6.               **for**  $k \leftarrow 1$  to  $columns[A]$
7.                   **do**  $C[i, j] = C[i, j] + A[i, k] \cdot B[k, j]$
8.       **return**  $C$

$$\boxed{n \times m} \cdot \boxed{m \times l} = \boxed{n \times l}$$

The number of scalar multiplications is:  $n \times m \times l$

# Matrix Multiplication Brute Force Alg.

- If  $m=l=n$ 
  - Then Time complexity is  $n^3$

# Strassen's Matrix Multiplication

Strassen observed [1969] that the product of two matrices can be computed as follows:

$$\begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} * \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}$$
$$= \begin{pmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 + M_3 - M_2 + M_6 \end{pmatrix}$$



# Formulas for Strassen's Algorithm

$$M_1 = (A_{00} + A_{11}) * (B_{00} + B_{11})$$

$$M_2 = (A_{10} + A_{11}) * B_{00}$$

$$M_3 = A_{00} * (B_{01} - B_{11})$$

$$M_4 = A_{11} * (B_{10} - B_{00})$$

$$M_5 = (A_{00} + A_{01}) * B_{11}$$

$$M_6 = (A_{10} - A_{00}) * (B_{00} + B_{01})$$

$$M_7 = (A_{01} - A_{11}) * (B_{10} + B_{11})$$

$$\begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} * \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}$$

$$\begin{pmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 + M_3 - M_2 + M_6 \end{pmatrix}$$

# Analysis of Strassen's Algorithm

- If  $n$  is not a power of 2, matrices can be padded with zeros.
- Number of multiplications:  
 $M(n) = 7M(n/2), \quad M(1) = 1$
- Solution:  $M(n) = 7^{\log_2 n} = n^{\log_2 7} \approx n^{2.807}$  vs.  $n^3$  of brute-force alg.
- Algorithms with better asymptotic efficiency are known but they are even more complex.

# The End