CSC311 – Spring 2017
Designand Analysis of Algorithms
2. Growth of Functions

(Chap. 3 – Introduction to Algorithms (3rd edition)
by Cormen, Leiserson, Rivest & Stein)

Prof. Mohamed Menai
Department of Computer Science
King Saud University

Outline

- Asymptotic notation
- The O-notation
- The Θ-notation
- The Ω-notation
- The o-notation
- The ω-notation

2

Overview

- Order of growth of functions provides a simple characterization of efficiency
- Allows for comparison of relative performance between alternative algorithms
- Concerned with ***asymptotic*** efficiency of algorithms
- Best asymptotic efficiency usually is best choice except for smaller inputs
- Several standard methods to simplify asymptotic analysis of algorithms

3

Asymptotic Notation

- Applies to functions whose domains are the set of natural numbers $\mathrm{N} = \{0,1,2,\ldots\}$
- If time resource $T(n)$ is being analyzed, the function's range is usually the set of non-negative real numbers: $T(n) \in \mathrm{R}^{+}$
- If space resource $S(n)$ is being analyzed, the function's range is usually also the set of natural numbers: $S(n) \in \mathrm{N}$

4

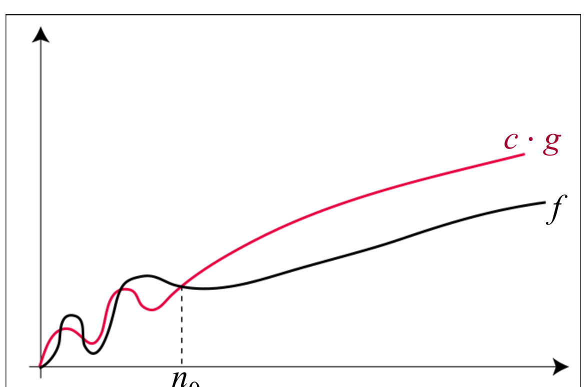
The O-Notation

- The O-notation is an asymptotic upper bound.
- $f(n) = O(g(n))$ pronounced "f of n is big-oh of g of n":

  $O(g(n)) = \{ f(n) : \exists c > 0, n_0 > 0$ so that $\forall n \geq n_0: 0 \leq f(n) \leq c \cdot g(n)\}$

The O-Notation

$O(g(n)) = \{ f(n) : \exists c > 0, n_0 > 0$ so that $\forall n \geq n_0: 0 \leq f(n) \leq c \cdot g(n)\}$



Using the Definition of the *O*-Notation

Example: Show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$.
Solution: Since when $x > 1$, $x < x^2$ and $1 < x^2$

$$0 \leq x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2$$

- Can take $c = 4$ and $n_0 = 1$ as witnesses to show that $f(x)$ is $O(x^2)$

- Alternatively, when $x > 2$, we have $2x \leq x^2$ and $1 < x^2$. Hence, $0 \leq x^2 + 2x + 1 \leq x^2 + x^2 + x^2 = 3x^2$ when $x > 2$.
  - Can take $c = 3$ and $n_0 = 2$ as witnesses instead.

7

Combinations of Functions

- If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$ then $(f_1 + f_2)(x)$ is $O(\max(|g_1(x)|,|g_2(x)|))$.
  - See next slide for proof
- If $f_1(x)$ and $f_2(x)$ are both $O(g(x))$ then $(f_1 + f_2)(x)$ is $O(g(x))$.
- If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$ then $(f_1 f_2)(x)$ is $O(g_1(x)g_2(x))$.
- $f(n) = O(g(n)) \Rightarrow f(n) + g(n) = O(g(n))$

8

Combinations of Functions

- If $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$ then
  $(f_1 + f_2)(x)$ is $O(\max(g_1(x), g_2(x)))$.

  – By the definition of O-notation, there are constants c_1, c_2, k_1, k_2 such that $f_1(x) \le c_1.g_1(x)$ when $x > k_1$ and $f_2(x) \le c_2.g_2(x)$ when $x > k_2$.
  – $(f_1 + f_2)(x) = f_1(x) + f_2(x)$
  – $f_1(x) + f_2(x) \le c_1.g_1(x) + c_2.g_2(x) \le c_1.g(x) + c_2.g(x)$
  where $g(x) = \max(g_1(x), g_2(x))$
  $= (c_1 + c_2).g(x)$
  $= c.g(x)$ where $c = c_1 + c_2$
  – Therefore $(f_1 + f_2)(x) \le c.g(x)$ whenever $x > k$, where $k = \max(k_1, k_2)$

9

The Θ-Notation

- The Θ-notation is an asymptotically tight bound on $f(n)$.
- Θ-notation is a stronger notion than O-notation. $\Theta(g(n))$ is a sub-set of $O(g(n))$

10

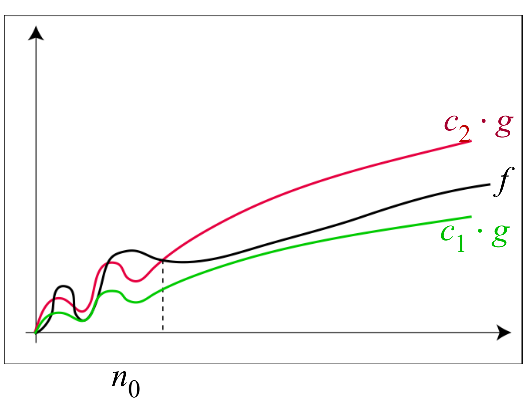
The Θ-Notation

- $\Theta(g(n))$ is the set of functions:
  $\Theta\ (g(n)) = \{ f(n) : \exists c_1, c_2 > 0, n_0 > 0$ so that
  $\forall n \geq n_0: c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$
- A function $f(n)$ belongs to the set $\Theta(g(n))$ if there exist positive constants c_1 and c_2 such that it can be "sandwiched" between $c_1 \cdot g(n)$ and $c_2 \cdot g(n)$, for sufficiently large n.
- Notation: $f(n) = \Theta(g(n))$ even if $f(n)$ is a member of $\Theta(g(n))$.

11

The Θ-Notation

$f(n) = \Theta(g(n))$: $g(n)$ asymptotically bounds a function from above and below.



12

The Θ-Notation Estimates for Polynomials

Theorem: Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_o$ where $a_0, a_1, \ldots, a_n$ are real numbers with $a_n \neq 0$. Then $f(x)$ is of order x^n (or $\Theta(x^n)$).

Example:
The polynomial $f(x) = 8x^5 + 5x^2 + 10$ is order of x^5 (or $\Theta(x^5)$).
The polynomial $f(x) = 8x^{199} + 7x^{100} + x^{99} + 5x^2 + 25$ is order of x^{199} (or $\Theta(x^{199})$).

13

The Ω-Notation

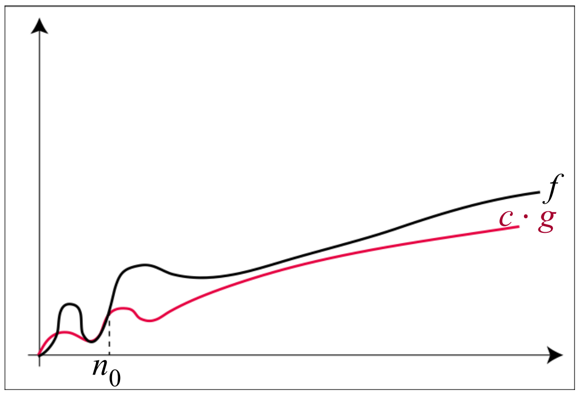
- The O-notation provides an asymptotic upper bound on a function.
- The Ω-notation provides an asymptotic lower bound on a function.
- $f(n) = \Omega(g(n))$ pronounced "f of n is big-omega of g of n":

  $\Omega(g(n)) = \{ f(n) : \exists c > 0, n_0 > 0$ so that $\forall n \geq n_0 : f(n) \geq c \cdot g(n) \geq 0\}$

14

The Ω-Notation

$\Omega(g(n)) = \{ f(n) : \exists c > 0, n_0 > 0$ so that
$\forall n \geq n_0 : f(n) \geq c \cdot g(n) \geq 0\}$



The Ω-Notation

Example: Show that $f(x) = 8x^3 + 5x^2 + 7$ is $\Omega(g(x))$ where $g(x) = x^3$

.

Solution: $f(x) = 8x^3 + 5x^2 + 7 \geq 8x^3$ for all positive real numbers x.

Is it also the case that $g(x) = x^3$ is $O(8x^3 + 5x^2 + 7)$?

Theorem

For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

17

Example

Show that the sum of the first n positive integers is $\Theta(n^2)$.

Solution: Let $f(n) = 1 + 2 + \cdots + n$.

- We easily show that $f(n)$ is $O(n^2)$.
- To show that $f(n)$ is $\Omega(n^2)$, we need a positive constant c such that $f(n) > cn^2$ for sufficiently large n. Summing only the terms greater than $n/2$ we obtain the inequality

$$\begin{aligned} 1 + 2 + \cdots + n &\geq \lceil n/2\rceil + (\lceil n/2\rceil + 1) + \cdots + n \\ &\geq \lceil n/2\rceil + \lceil n/2\rceil + \cdots + \lceil n/2\rceil \\ &= (\mathrm{n} - \lceil n/2\rceil + 1)\lceil n/2\rceil \\ &\geq (\mathrm{n}/2)(\mathrm{n}/2) = n^2/4 \end{aligned}$$

- Taking $c = ¼$, $f(n) > cn^2$ for all positive integers n. Hence, $f(n)$ is $\Omega(n^2)$, and we can conclude that $f(n)$ is $\Theta(n^2)$.

18

The o-Notation

- The asymptotic upper bound provided by the O-notation may or may not be asymptotically tight:
  - The bound $2n^2 = O(n^2)$ is asymptotically tight.
  - The bound $2n = O(n^2)$ is not.
- The o-notation is used to denote an upper bound that is not asymptotically tight.

  $f(n) = o(g(n))$ pronounced "f of n is little-oh of g of n": $o(g(n)) = \{ f(n) : \forall c > 0\ \exists n_0 > 0$ so that $\forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$
- For example, $2n = o(n^2)$, but $2n^2 \neq o(n^2)$

19

The o-Notation

$o(g(n)) = \{ f(n) : \forall c > 0\ \exists n_0 > 0$ so that

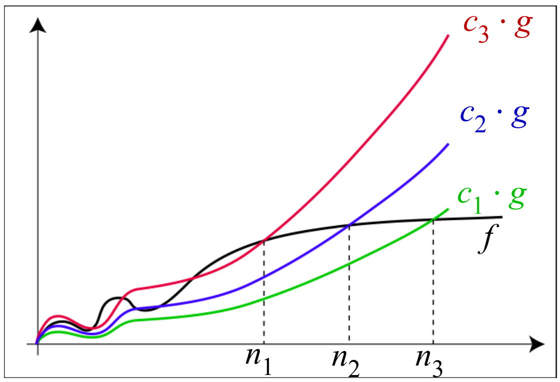
$\forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$

- In $f(n) = O(g(n))$, the bound $f(n) \leq c \cdot g(n)$ holds for some constant $c > 0$.
- In $f(n) = o(g(n))$, the bound $f(n) \leq c \cdot g(n)$ holds for all constants $c > 0$.
- Intuitively, the function $f(n)$ becomes insignificant relative to $g(n)$, as n approaches infinity: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

20

The o-Notation

$o(g(n)) = \{ f(n) : \forall c > 0\ \exists n_0 > 0$ so that $\forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$

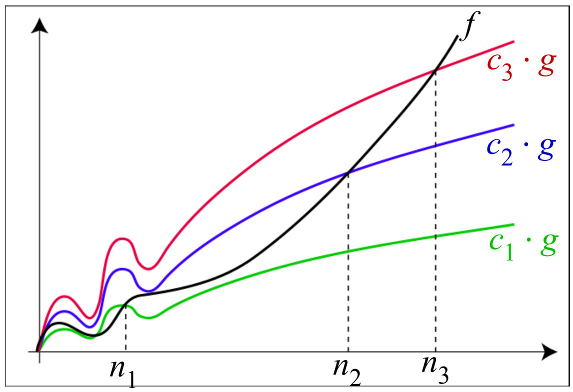


The ω-Notation

- The ω-notation is to Ω-notation, as the o-notation is to O-notation.
- The ω-notation is used to denote a lower bound that is not asymptotically tight.
- $f(n) \in \omega(g(n))$ if and only if $g(n) \in o(f(n))$
- pronounced "f of n is little-omega of g of n".
- $f(n) \in \omega(g(n))$ implies that: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

The ω-Notation

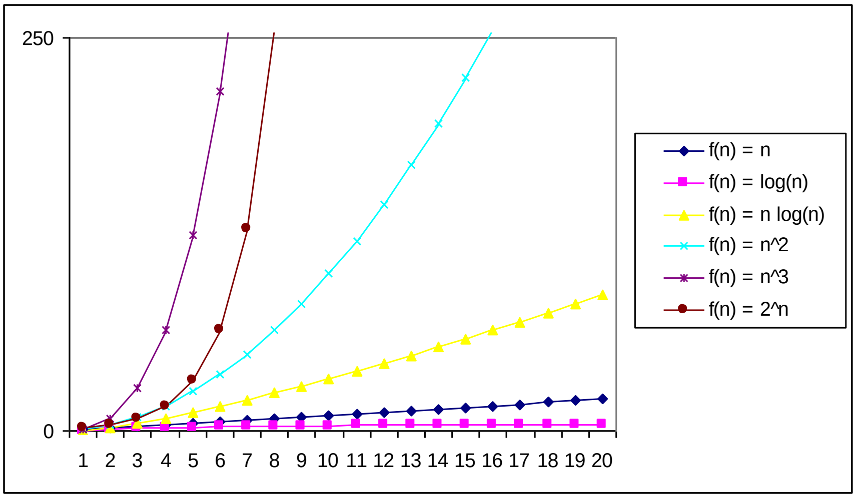
$\omega(g(n)) = \{ f(n) : \forall c > 0\ \exists n_0 > 0$ so that $\forall n \geq n_0 : f(n) \geq c \cdot g(n) \geq 0\}$



Running Times

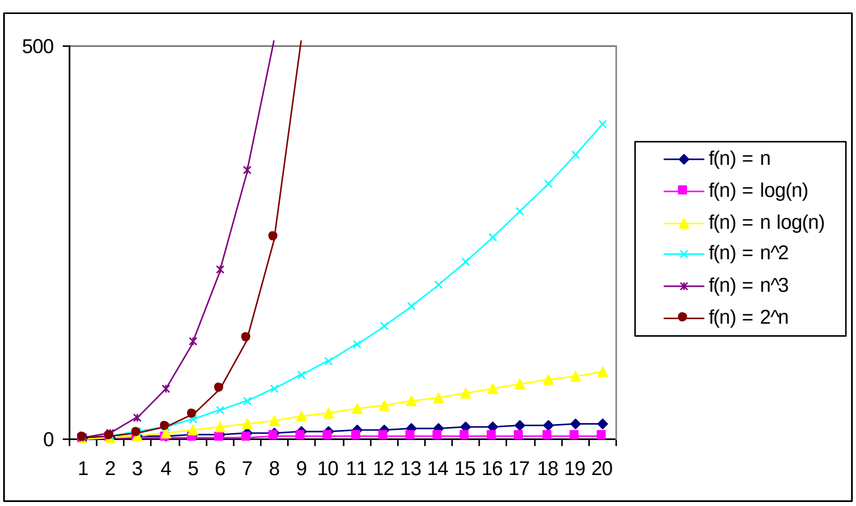
- "Running time is $O(f(n))$" $\Rightarrow$ Worst case is $O(f(n))$
- $O(f(n))$ bound on the worst-case running time $\Rightarrow$ $O(f(n))$ bound on the running time of every input.
- $\Theta(f(n))$ bound on the worst-case running time $\not\Rightarrow$ $\Theta(f(n))$ bound on the running time of every input.
- "Running time is $\Omega(f(n))$" $\Rightarrow$ Best case is $\Omega(f(n))$

Practical Complexity



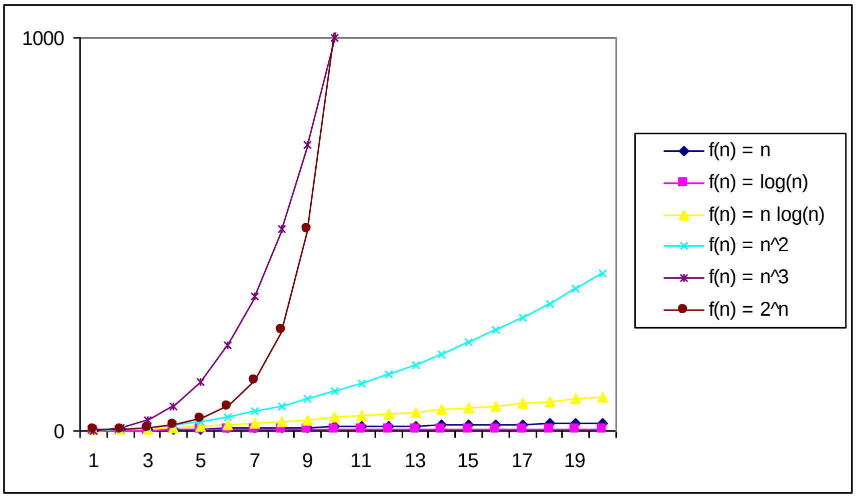
25

Practical Complexity



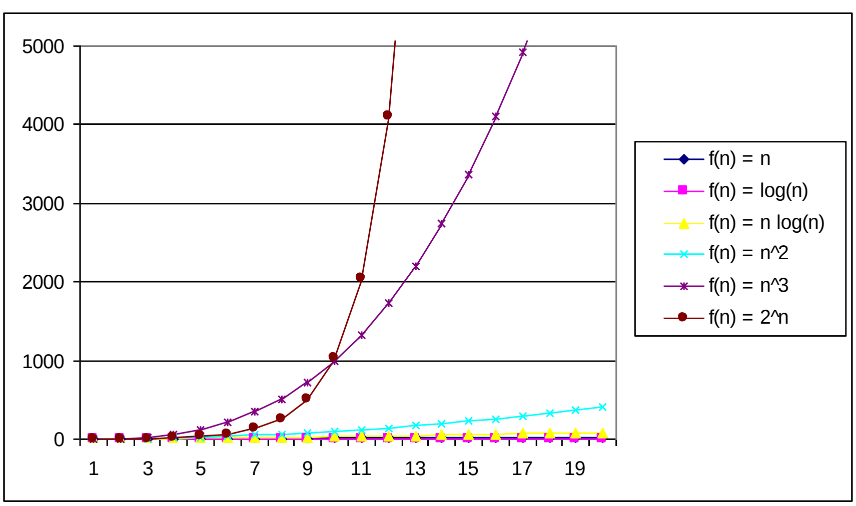
26

Practical Complexity



27

Practical Complexity



28

Comparison of Functions

Transitivity

- $f(n) = O(g(n))$ and
  $g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$
- $f(n) = \Omega(g(n))$ and
  $g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$
- $f(n) = \Theta(g(n))$ and
  $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

Reflexivity

- $f(n) = O(f(n))$
  $f(n) = \Omega(f(n))$
  $f(n) = \Theta(f(n))$

29

Comparison of Functions

Symmetry

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

Transpose Symmetry

- $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$
- $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$

30

Asymptotic Analysis and Limits

$$\text{if } \lim_{n\to\infty}\frac{f(n)}{g(n)}=0, \text{ then } f(n)=o(g(n)).$$

$$\text{if } \lim_{n\to\infty}\frac{f(n)}{g(n)}=c, \text{ for some constant } c>0, \text{ then } f(n)=\Theta(g(n)).$$

$$\text{if } \lim_{n\to\infty}\frac{f(n)}{g(n)}=0, \text{ then } a^{f(n)}=o\left(a^{g(n)}\right), \text{ for any } a>1.$$

$$f(n)=o(g(n))\Rightarrow a^{f(n)}=o\left(a^{g(n)}\right), \text{ for any } a>1.$$

$$f(n)=\Theta(g(n))\not\Rightarrow a^{f(n)}=\Theta\left(a^{g(n)}\right).$$

31

Standard Notation and Common Functions

- Important relationships
  - For all real constants a and b such that $a > 1$,
    $n^b = o(a^n)$

    that is, any exponential function with a base strictly greater than unity grows faster than any polynomial function.
  - For all real constants a and b such that $a > 0$,
    $\log^b n = o(n^a)$

    that is, any positive polynomial function grows faster than any polylogarithmic function.

32

Standard Notation and Common Functions

- Factorials
  - For all n the function $n!$ or "n factorial" is given by
    $n! = n \times (n-1) \times (n-2) \times (n-3) \times \ldots \times 2 \times 1$
  - It can be established that
    $n! = o(n^n)$
    $n! = \omega(2^n)$
    $\log(n!) = \Theta(n \log n)$

33

Asymptotic Running Time of Algorithms

- We consider algorithm A better than algorithm B if:
  $$T_A(n) = o(T_B(n))$$
- Why is it acceptable to ignore the behavior of algorithms for small inputs?
- Why is it acceptable to ignore the constants?
- What do we gain by using asymptotic notation?

34

Things to Remember

- Asymptotic analysis studies how the values of functions compare as their arguments grow without bounds.
- Ignores constants and the behavior of the function for small arguments.
- Acceptable because all algorithms are fast for small inputs and growth of running time is more important than constant factors.

Things to Remember

- Ignoring the usually unimportant details, we obtain a representation that succinctly describes the growth of a function as its argument grows and thus allows us to make comparisons between algorithms in terms of their efficiency.

Reading

Chapter 3

Cormen, Leiserson, Rivest, & Stein, Introduction to Algorithms (Third Edition), The MIT Press, 2009.