Algorithms 演算法

Foundations

— 1.2 Growth of Functions —

Professor James Chien-Mo Li 李建模
Electrical Engineering Department
National Taiwan University

Outline

- **Introduction**
- **Getting Started**
- **Growth of Functions**
- **Recurrence**

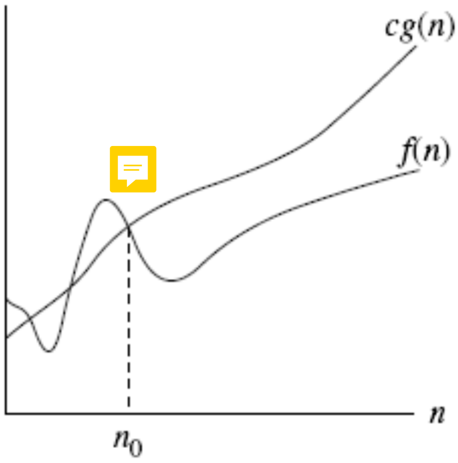
Define Functions: O Θ Ω
In Asymptotic Analysis

(Big) O-notation

- $f(n) = O(g(n))$
  - there exist positive constants c and n_0 such that
  - $0 \le f(n) \le cg(n)$ for all $n \ge n_0$
- $g(n)$ is ***asymptotic upper bound*** for $f(n)$
- **Examples:**
  - $4n^2 = O(n^3)$; $c=2$, $n_0=4$
  - $2n^2 = O(n^2)$;

$\lg n = O(\sqrt{n})$ $\quad \lg n = O(n^{0.1})$

$n^3 = O(2^n)$ $\quad 2^n = O(n!)$



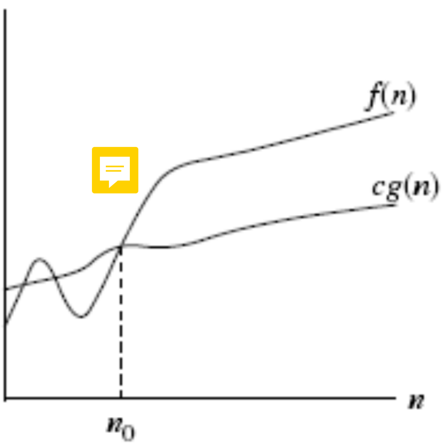
how to prove the big-O relationship?

$$f(n) = O(g(n)) \text{ iff } \lim_{n\to\infty} \frac{f(n)}{g(n)} = c \quad , \textit{ for some } c \ge 0$$

(Big) Ω-notation

- $f(n) = \Omega(g(n))$
  - there exist positive constants c and n_0 such that
  - $0 \le cg(n) \le f(n)$ for all $n \ge n_0$
- $g(n)$ is ***asymptotic lower bound*** for $f(n)$
- **Examples:**
  - $n^3 = \Omega(n^2)$; with $c=1$, $n_0=1$.
  - $\sqrt{n} = \Omega(\lg n)$, with $c = 1$ and $n_0 = 16$

$100n^2 = \Omega(n^2)$

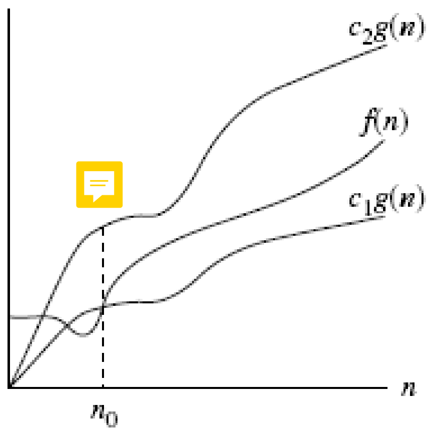


how to prove the big-Omega relationship?

$$f(n) = \Omega(g(n)) \text{ iff } \lim_{n\to\infty} \frac{g(n)}{f(n)} = c \quad , \textit{ for some } c \ge 0$$

Θ-notation

- $f(n) = \Theta(g(n))$
  - there exist positive constants c_1, c_2, and n_0 such that
  - $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$
- $g(n)$ is ***asymptotically tight bound*** for $f(n)$
- **Examples:**
  - $n^2/2 - 2n = \Theta(n^2)$, with $c_1 = 1/4$, $c_2 = 1/2$, and $n_0 = 8$
- <u>(Theorem3.1) $f(n) = \Theta(g(n))$ if and only if $f = O(g(n))$ and $f = \Omega(g(n))$</u>



How to prove the Θ relationship?

$$f(n) = \Theta(g(n)) \ \ iff \ \ \lim_{n\to\infty} \frac{f(n)}{g(n)} = c \ , \ \ for \ some \ c > 0$$

(small) o-notation

- $f(n) = o(g(n))$
  - for all constants $c > 0$, there exists constant $n_0 > 0$ such that
  - $0 \le f(n) < cg(n)$ for all $n \ge n_0$
  - $f(n)$ is ***asymptotically smaller than*** $g(n)$
- **Examples**
  - $n^{1.9999} = o(n^2)$
  - $n^2 / \lg n = o(n^2)$
  - $n^2 \ne o(n^2)$ (just like $2 \not< 2$)

How to Prove the o - relationship?

$$f(n) = o(g(n)) \ \ iff \ \ \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

(small) ω-notation

- $f(n) = \omega(g(n))$
  - for all constants $c > 0$, there exists constant $n_0 > 0$ such that
  - $0 \le cg(n) < f(n)$ for all $n \ge n_0$
  - $f(n)$ is ***asymptotically larger than*** $g(n)$
- **Examples**
  - $n^{2.0001} = \omega(n^2)$
  - $n^2 \lg n = \omega(n^2)$
  - $n^2 \ne \omega(n^2)$

How to prove ω – relatinship?

$$f(n) = \omega(g(n)) \;\; iff \;\; \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Analogy

$$f(n) = O(g(n)) \approx f \le g$$
$$f(n) = \Omega(g(n)) \approx f \ge g$$
$$f(n) = \Theta(g(n)) \approx f = g$$
$$f(n) = o(g(n)) \approx f < g$$
$$f(n) = \omega(g(n)) \approx f > g$$

- **An algorithm has worst-case run time $O(g(n))$ means that**
  - **exists constant c s.t. for every large *n*, *every execution* on an input of size n takes *at most* $cg(n)$ time.**
  - **example: insertion sort *worst-case* running time is $O(n^2)$**
- **An algorithm has best-case run time $\Omega(g(n))$ means that**
  - **exists constant c s.t. for every large *n* , *at least one* execution on an input of size n takes *at least* $cg(n)$ time.**
  - **example: insertion sort *best-case* running time is $\Omega(n)$**

Properties

- ***Transitivity***:
  - If $f(n) = \Pi[g(n)]$ and $g(n) = \Pi[h(n)]$, then $f(n) = \Pi[h(n)]$
  - where $\Pi = \mathrm{O}, \mathrm{o}, \Omega, \omega$, or Θ
- ***Rule of sums***:
  - $f(n) + g(n) = \Pi[\max\{f(n), g(n)\}]$, where $\Pi = \mathrm{O}, \Omega$, or Θ
- ***Rule of products***:
  - If $f_1(n) = \Pi[g_1(n)]$ and $f_2(n) = \Pi[g_2(n)]$, then $f_1(n)\, f_2(n) = \Pi[g_1(n)\, g_2(n)]$, where $\Pi = \mathrm{O}, \mathrm{o}, \Omega, \omega$, or Θ
- ***Transpose symmetry***:
  - $f(n) = \mathrm{O}[g(n)]$ iff $g(n) = \Omega(f(n))$
  - $f(n) = \mathrm{o}[g(n)]$ iff $g(n) = \omega(f(n))$.
- ***Reflexivity***: $f(n) = \Pi[f(n)]$, where $\Pi = \mathrm{O}, \Omega$, or Θ
- ***Symmetry***: $f(n) = \Theta[g(n)]$ iff $g(n) = \Theta[f(n)]$

Comparison of Running Time

- 1,000 Million Instruction Per Second (MIPS)
- Different asymptotic function makes a huge difference

Order	Name	n = 10	n = 100	$n = 10^3$	$n = 10^6$
1	constant	1×10^{-9} sec	1×10^{-9} sec	1×10^{-9} sec	1×10^{-9} sec
$\lg n$	logarithmic	3×10^{-9} sec	7×10^{-9} sec	1×10^{-8} sec	2×10^{-8} sec
$\sqrt{n}$	square root	3×10^{-9} sec	1×10^{-8} sec	3×10^{-8} sec	1×10^{-6} sec
n	Linear	1×10^{-8} sec	1×10^{-7} sec	1×10^{-6} sec	0.001 sec
$n \lg n$	linearithmic	3×10^{-8} sec	2×10^{-7} sec	3×10^{-6} sec	0.006 sec
n^2	quadratic	1×10^{-7} sec	1×10^{-5} sec	0.001 sec	16.7 min
n^3	cubic	1×10^{-6} sec	0.001 sec	1 sec	3×10^{5} cent.
2^n	exponential	1×10^{-6} sec	3×10^{17} cent.	∞	∞
$n!$	factorial	0.003 sec	∞	∞	∞

Food for Thoughts

- Q1: merge sort is $O(n \lg n)$, merge sort is also $\Theta(n \lg n)$
  - which one should I write in exam?
- Q2: Suppose f=n^2 , g=n^3
  - f= $O(n^3)$ = g
  - f=g ? What is wrong?