Algorithms
Chapter 3 Growth of Functions

Associate Professor: Ching-Chi Lin
林清池 副教授
chingchi.lin@gmail.com

Department of Computer Science and Engineering
National Taiwan Ocean University

Outline

- **Asymptotic notation**
- Standard notations and common functions

The purpose of this chapter$_{1/3}$

- The order of growth of the running time of an algorithm gives us some information about:
  - the algorithm's efficiency
  - the relative performance of alternative algorithms
- The merge sort, with its $\Theta(n \lg n)$ worst-case running time, beats insertion sort, whose worst-case running time is $\Theta(n^2)$.
- For large enough inputs, the following are dominated by the effects of the input size itself.
  - multiplicative constants
  - lower-order terms of an exact running time

The purpose of this chapter$_{2/3}$

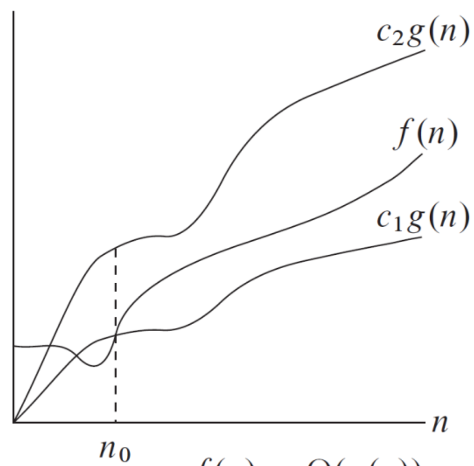
- When the input size *n* becomes large enough, we are studying the **asymptotic** efficiency of algorithms.
- That is, we are concerned with
  - how the running time of an algorithm increases with the size of the input **in the limit**, as the size of the input increases without bound.
- Usually, an algorithm that is asymptotically more efficient will be the best choice for all but very small inputs.

The purpose of this chapter$_{3/3}$

- We will study how to **measure** and **analyze** an algorithm's efficiency for large inputs.
- The next section begins by defining asymptotic notations,
  - Θ-notation
  - O-notation
  - Ω-notation

Θ-notation

- For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions
  - $\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$.
- For $n \geq n_0$, the function $f(n)$ is equal to $g(n)$ to within a constant factor.
- Here, $g(n)$ is an **asymptotically tight bound** for $f(n)$.
- Because $\Theta(g(n))$ is a set, we could write "$f(n) \in \Theta(g(n))$".
- Usually, we write "$f(n) = \Theta(g(n))$".



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

An example

- To show that $n^2/2 - 3n = \Theta(n^2)$, we must determine positive constants c_1, c_2, and n_0 such that

$$c_1 n^2 \le n^2/2 - 3n \le c_2 n^2 \text{ for all } n \ge n_0.$$

- Dividing by n^2 yields

$$c_1 \le 1/2 - 3/n \le c_2.$$

  - $c_1 \le 1/2 - 3/n$ holds for $n \ge 7$ by $c_1 \le 1/14$
  - $1/2 - 3/n \le c_2$ holds for $n \ge 1$ by $c_2 \ge 1/2$

- Thus, choosing $c_1 = 1/14$, $c_2 = 1/2$, and $n_0 = 7$, we can verify that $n^2/2 - 3n = \Theta(n^2)$.

- Show that $3n^3 - 2 = \Theta(n^3)$.

Another example

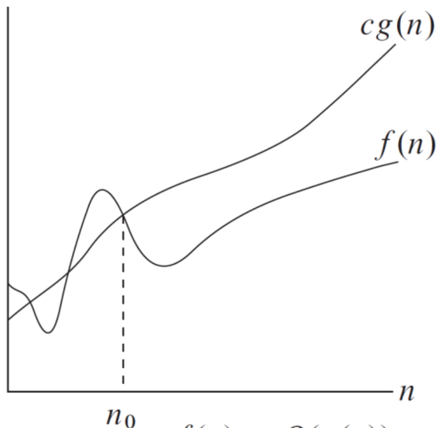
- We show that $6n^3 \neq \Theta(n^2)$ by contradiction.
  - Suppose c_2 and n_0 exist such that $6n^3 \leq c_2 n^2$ for all $n \geq n_0$.
  - Then $n \leq c_2/6$, a contradiction.
  - Since c_2 is constant, it cannot possibly hold for arbitrary large n.

Summary

- The lower-order terms can be ignored
  - because they are insignificant for large n.
- The coefficient of the highest-order term can likewise be ignored
  - since it only changes c_1 and c_2 by a constant factor equal to the coefficient.
- In general, for any polynomial $p(n) = a_d n^d + \ldots + a_1 n + a_0$, where a_i are constants and $a_d > 0$, we have $p(n) = \Theta(n^d)$.
- For example, $f(n) = an^2 + bn + c$, where a, b, and c are constants and $a > 0$. Then, we have $f(n) = \Theta(n^2)$.

O-notation

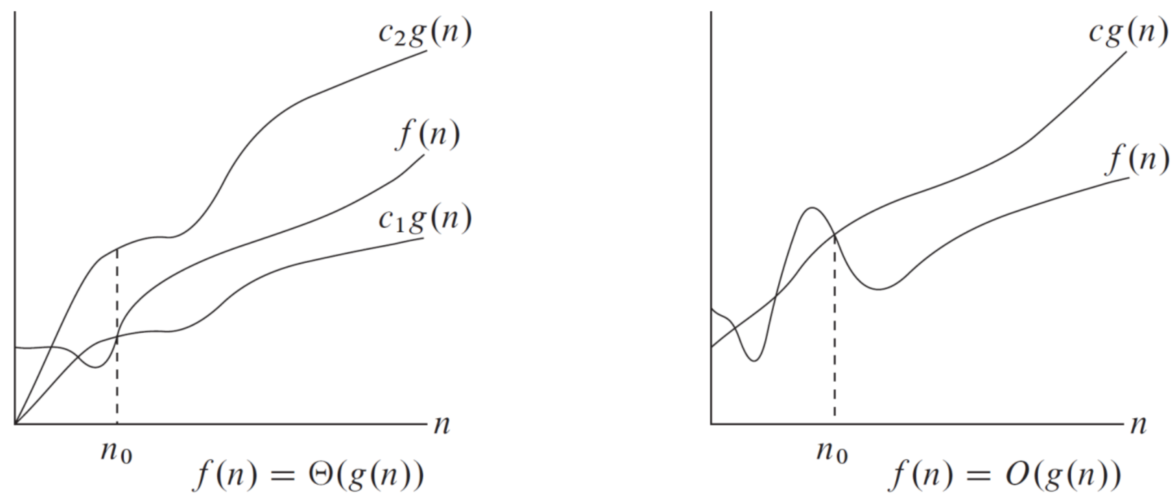
- For a given function $g(n)$, we denote by $O(g(n))$ the set of functions
  - $O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0 \}$.
- We write $f(n) = O(g(n))$ implies $f(n)$ is a member of the set $O(g(n))$.
- Note that $f(n) = \Theta(g(n))$ implies $f(n) = O(g(n))$.
  - any proof showing that $f(n) = \Theta(g(n))$ also shows that $f(n) = O(g(n))$.
  - $\Theta(g(n)) \subseteq O(g(n))$.
- Show that $3n^2 - 2 = O(n^2)$.



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

The meaning of O-notation$_{1/2}$

- The Θ-notation asymptotically bounds a function from above and below.
- When we have only an **asymptotic upper bound**, we use O-notation.
- Hence, Θ-notation is a stronger notation than O-notation.

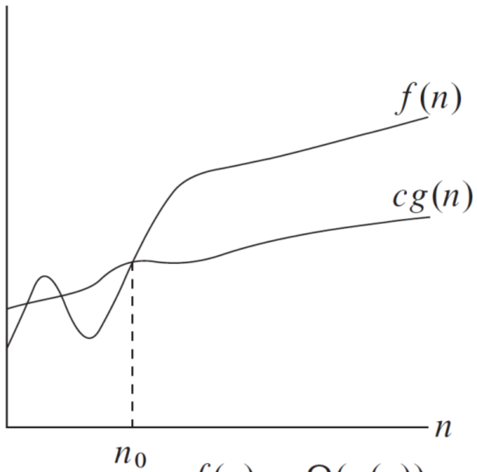


The meaning of O-notation$_{2/2}$

- Any linear function $an + b$ is in $O(n^2)$, which is easily verified by taking $c = a + |b|$ and $n_0 = 1$.
  - $an + b \leq (a + |b|)\, n^2$ for $n \geq 1$
- $f(n) = O(g(n))$ merely claims that
  - $g(n)$ is an asymptotic **upper** bound on $f(n)$
  - does not claim about how tight an upper bound it is
- In practical, O-notation is used to describe the **worst-case** running time of an algorithm.
- "an algorithm is $O(g(n))$" means that
  - the running time is at most constant times $g(n)$, for sufficiently large n
  - no matter what particular input of size n is chosen for each value of n

Ω-notation

- For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions
  - $\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0\}$.
- We write $f(n) = \Omega(g(n))$ implies $f(n)$ is a member of the set $\Omega(g(n))$.
- Ω-notation provides **asymptotic lower bound**.



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

The relationship between Θ, O, and Ω

- Theorem 3.1
  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 $f(n) = \Omega(g(n))$.
- For example:
  - $n^2/2 - 3n = \Theta(n^2) \Rightarrow n^2/2 - 3n = O(n^2)$ and $n^2/2 - 3n = \Omega(n^2)$
  - $n^2/2 - 3n = O(n^2)$ and $n^2/2 - 3n = \Omega(n^2) \Rightarrow n^2/2 - 3n = \Theta(n^2)$

The meaning of Ω-notation

- The Ω-notation is used to bound the **best-case** running time of an algorithm.
- "an algorithm is $\Omega(g(n))$" means that
  - the running time is at least constant times $g(n)$, for sufficiently large n
  - no matter what particular input of size n is chosen for each value of n

o-notation

- For a given function $g(n)$, we denote by $o(g(n))$ the set of functions
  - $o(g(n)) = \{f(n)$: for **any** positive constant $c>0$, there exists a constant $n_0>0$ such that $0 \le f(n) < cg(n)$ for all $n \ge n_0\}$.
- We use o-notation to denote an upper bound that is **not** asymptotically tight.
- For example, $2n=o(n^2)$, but $2n^2 \neq o(n^2)$.
- Intuitively, the function $f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity; that is,

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

ω-notation

- For a given function $g(n)$, we denote by $\omega(g(n))$ the set of functions
  - $\omega(g(n))=\{f(n)$: for **any** positive constant $c>0$, there exists a constant $n_0>0$ such that $0\leq cg(n)<f(n)$ for all $n\geq n_0\}$.
- We use ω-notation to denote a lower bound that is **not** asymptotically tight.
- For example, $n^2/2=\omega(n)$, but $n^2/2\neq\omega(n^2)$.
- The relation $f(n)=\omega(g(n))$ implies that

  $$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\infty.$$

  if the limit exists.

Comparison of functions$_{1/4}$

- 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 $f(n) = \Omega(h(n))$,
  - $f(n) = o(g(n))$ and $g(n) = o(h(n))$ imply $f(n) = o(h(n))$,
  - $f(n) = \omega(g(n))$ and $g(n) = \omega(h(n))$ imply $f(n) = \omega(h(n))$.

Comparison of functions$_{2/4}$

- 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 $g(n) = \Omega(f(n))$,
  - $f(n) = o(g(n))$ if and only if $g(n) = \omega(f(n))$.

Comparison of functions$_{3/4}$

- Analogy between the asymptotic comparison and the real number comparison:
  - $f(n) = \Theta(g(n)) \approx a = b$.
  - $f(n) = O(g(n)) \approx a \leq b$.
  - $f(n) = \Omega(g(n)) \approx a \geq b$.
  - $f(n) = o(g(n)) \approx a < b$.
  - $f(n) = \omega(g(n)) \approx a > b$.

Comparison of functions$_{4/4}$

- Trichotomy property of real numbers does not carry over to asymptotic notation:
  - **Trichotomy:** For any two real numbers a and b, exactly one of the following must hold: $a < b$, $a = b$, or $a > b$.
- Not all functions are asymptotically comparable.
  - For two functions $f(n)$ and $g(n)$, it may be the case that neither $f(n)=O(g(n))$ nor $f(n)=\Omega(g(n))$.
  - For example, the function n and $n^{1+\sin n}$ cannot be compared, since the value of $n^{1+\sin n}$ oscillates between 0 and 2.