Algorithms
Chapter 3 Growth of Functions

Instructor: Ching-Chi Lin
林清池 助理教授
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$_{2/3}$

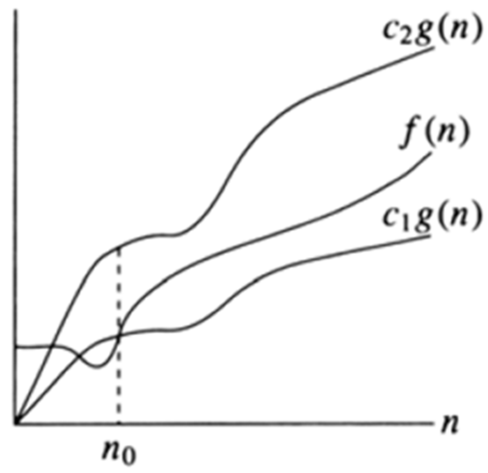
- When the input size *n* becomes large enough, we are studying the **asymptotic** efficiency of algorithms.
- That is, we are concerned with
  - how the running time of an algorithm increases with the size of the input in the limit, as the size of the input increases without bound.
- Usually, an algorithm 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
- Then, we review
  - the commonly used functions in the analysis of algorithms.

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 $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0\}$.
- For $n \ge n_0$, 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 "$f(n) = \Theta(g(n))$".



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

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

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

Another example

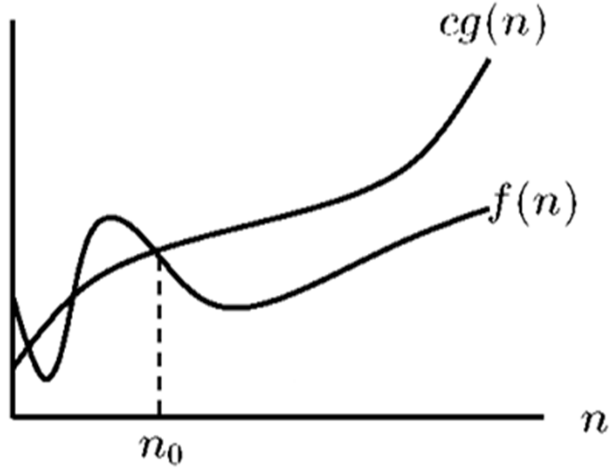
- We show that $6n^3 \neq \Theta(n^2)$ by contradiction.
  - Suppose c_2 and n_0 exist such that $6n^3 \leq c_2 n^2$ for all $n \geq n_0$.
  - Then $n \leq c_2/6$, a contradiction.
  - Since c_2 is constant, it cannot possibly hold for arbitrary large n,

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 $p(n) = \Sigma_{i=0\sim d}\, a_i n^i$, 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 $f(n) = \Theta(n^2)$.

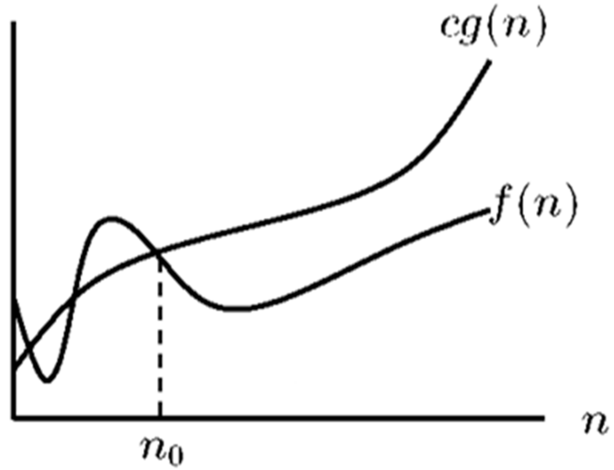
O-notation

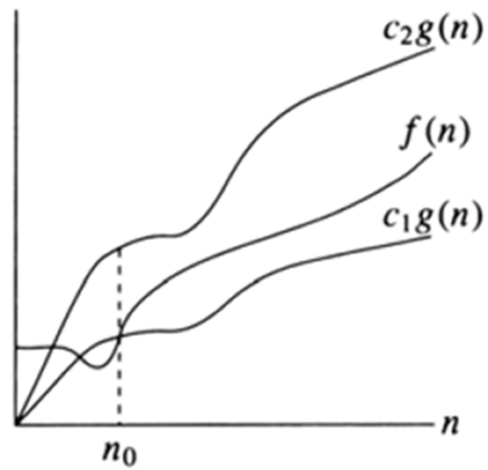
- For a given function $g(n)$, we denote by $O(g(n))$ the set of functions
  - $O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0 \}$.
- We write $f(n) = O(g(n))$ implies $f(n)$ is a member of the set $O(g(n))$.
- Note that $f(n) = \Theta(g(n))$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



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.



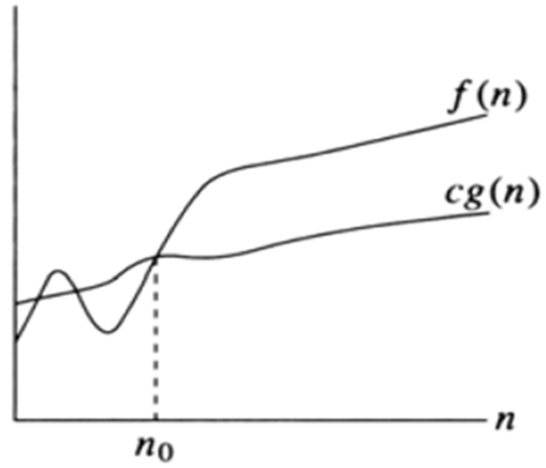


The meaning of O-notation$_{2/2}$

- 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 $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$.
- 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**.



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

Asymptotic notation in equations and inequalities$_{1/2}$

- On the right-hand side of an equation (or inequality)
  - the equal sign means set membership
  - $n = O(n^2)$ means that $n \in O(n^2)$
- In a formula
  - it is interpreted as some anonymous function that we do not care to name
  - $2n^2+3n+1=2n^2+\Theta(n)$ means that $2n^2+3n+1=2n^2+f(n)$, where $f(n) \in \Theta(n)$
- On the left-hand side of an equation
  - No matter how the anonymous functions are chosen on the left of the equal sign, there is a way to choose the anonymous functions on the right of the equal sign to make the equation valid
  - $2n^2+\Theta(n)=\Theta(n^2)$ means that for **any** function $f(n) \in \Theta(n)$, there is **some** function $g(n) \in \Theta(n^2)$ such that $2n^2+f(n)=g(n)$ for **all** n

Asymptotic notation in equations and inequalities$_{2/2}$

- A number of such relationships can be chained together, as in
  - $2n^2+3n+1=2n^2+\Theta(n)$
    $=\Theta(n^2)$
  - The first equation says that there is **some** function $f(n)\in\Theta(n)$ such that $2n^2+3n+1=2n^2+f(n)$ for all n.
  - The second equation says that for **any** function $g(n)\in\Theta(n)$, there is **some** function $h(n)\in\Theta(n^2)$ such that $2n^2+g(n)=h(n)$ for all n.
  - Note that the interpretation implies $2n^2+3n+1\ )=\Theta(n^2)$, which is what the chaining of equations intuitively gives us.

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 $2n^2 \ne o(n^2)$.
- 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.

Outline

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

Monotonicity

- A function $f(n)$ is **monotonically increasing** if $m \leq n$ implies $f(m) \leq f(n)$.
- A function $f(n)$ is **monotonically decreasing** if $m \leq n$ implies $f(m) \geq f(n)$.
- A function $f(n)$ is **strictly increasing** if $m < n$ implies $f(m) < f(n)$.
- A function $f(n)$ is **strictly decreasing** if $m < n$ implies $f(m) > f(n)$.

Floors and ceilings

- For any real number x, we denote the **greatest** integer less than or equal to x by $\lfloor x \rfloor$ and the **least** integer greater than or equal to x by $\lceil x \rceil$.
- For all real x,
  - $x-1 < \lfloor x \rfloor \le x \le \lceil x \rceil < x+1$

  For any integer n,

  $\lceil n/2 \rceil + \lfloor n/2 \rfloor = n$
- For any real number $n \ge 0$ and integers a, $b > 0$
  - $\lceil \lceil n/a \rceil / b \rceil = \lceil n/ab \rceil$, and $\lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/ab \rfloor$
  - $\lceil a/b \rceil \le (a+(b-1))/b$, and $\lfloor a/b \rfloor \ge (a-(b-1))/b$
- The floor and ceiling functions are monotonically increasing.

Modular arithmetic

- For any integer a and any positive integer n, the value $a \bmod n$ is the **remainder** (or **residue**) of the quotient a/n:
  - $a \bmod n = a - \lfloor a/n \rfloor n$
- If $(a \bmod n) = (b \bmod n)$, we write $a \equiv b \pmod{n}$ and say that a is equivalent to b, modulo n.
- $a \equiv b \pmod{n}$
  - **If** a and b have the same remainder when divided by n
  - **If and only if** n is a divisor of $b - a$
- We write $a \not\equiv b \pmod{n}$ if a is not equivalent to b, modulo n.

Polynomials

- **A polynomial in *n* of degree *d*** is a function

  $$P(n) = a_d n^d + a_{d-1} n^{d-1} + \ldots + a_2 n^2 + a_1 n + a_0$$

  - d is a nonnegative integer
  - $a_d, \ldots, a_0$ are constants called the coefficients of the polynomial
  - $a_d \neq 0$
- An **asymptotically positive function** is one that is positive for all sufficiently large n.
- A polynomial is asymptotically positive if and only if $a_d > 0$.
- For any real constant $a \geq 0$ (respectively, $a \leq 0$), the function n^a is monotonically increasing (respectively, decreasing).
- A function $f(n)$ is **polynomially bounded** if $f(n)=O(n^k)$ for some constant k.

Exponentials$_{1/2}$

- For all real a>0, m, and n, we have the following identities:
  - $a^0 = 1$, $a^1 = a$, $a^{-1} = 1/a$
  - $(a^m)^n = (a^n)^m = a^{mn}$
  - $a^m a^n = a^{m+n}$
  - $0^0 = 1$ (for convenient)
- For all real constants a and b such that $a > 1$, $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$, from which we conclude that $n^b = o(a^n)$.
- Thus, any exponential function with a base strictly greater than 1 grows faster than any polynomial function.

Exponentials$_{2/2}$

- The natural logarithm function for all real *x*,

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

  - $e = 2.71828$
- For all real *x*, we have $e^x \geq 1+x$
  - equality holds only when $x = 0$
- When $|x| \leq 1$, we have $1+x \leq e^x \leq 1+x+x^2$
- When $x \to 0$, we have $e^x = 1+x+\Theta(x^2)$
- For all *x*, $\lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n = e^x$

Logarithms$_{1/4}$

- We shall use the following notations:
  - $\lg n \;=\; \log_2 n$ (binary logarithm)
  - $\ln n \;=\; \log_e n$ (natural logarithm)
  - $\lg^k n \;=\; (\lg n)^k$ (exponentiation)
  - $\lg\lg n = \lg(\lg n)$ (composition)
- Note that $\lg n+k$ means $(\lg n)+k$, not $\lg(n+k)$.
- If we hold $b > 1$ constant, then for $n > 0$, the function $\log_b n$ is strictly increasing.

Logarithms$_{2/4}$

- For all real *a*, *b*, *c* > 0, and *n*, if the logarithm bases are not 1, then, we have
  - $\log_b a^n = n\log_b$ $\qquad$ $\log_b a = \dfrac{\log_c a}{\log_c b}$
  - $a^{\log_b c} = c^{\log_b a}$ $\qquad$ $a = b^{\log_b a}$
  - $\log_b \dfrac{1}{a} = -\log_b a$ $\qquad$ $\log_b a = \dfrac{1}{\log_a b}$
  - $\log_c (ab) = \log_c a + \log_c b$

Logarithms$_{3/4}$

- If $|x| < 1$, then
  - $$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \cdots$$
- We also have the following inequalities for $x > -1$:
  - $$\frac{x}{1+x} \leq \ln(1+x) \leq x$$
  - the equality holds only for $x = 0$

Logarithms$_{4/4}$

- f(n) is called **polylogarithmically bounded** if f(n) = $O(\lg^k n)$ for some constant k.
- By substituting lg n for n and 2^a for a in $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$
  - $$\lim_{n\to\infty} \frac{\lg^b n}{(2^a)^{\lg n}} = \lim_{n\to\infty} \frac{\lg^b n}{n^a} = 0$$
- So, for any constant $a > 0$
  - $\lg^b n = o(n^a)$
  - any positive function grows faster than any polylogarithmic function

Factorials

- $n! = \begin{cases} 1 & \text{if } n = 0, \\ n \cdot (n-1)! & \text{if } n > 0. \end{cases}$
- Stirling's approximation: $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \theta\left(\frac{1}{n}\right)\right)$
  - e is the base of the natural logarithm
  - give a tighter upper bound, and a tighter low bound
- One can prove
  - $n! = o(n^n) \qquad \lg(n!) = \theta(n \lg n)$
  - $n! = \omega(2^n)$
- For all n ≥ 1, we have
  - $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n}, \text{ where } \frac{1}{12n+1} < \alpha_n < \frac{1}{12n}$

Functional iteration

- Let $f(n)$ be a function over the reals. Then, for nonnegative integer i, we recursively define

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

- For example, if $f(n) = 2n$, then $f^{(i)}(n) = 2^i n$

The iterated logarithm function

- Let $\lg^{(i)} n$ be defined as above, with $f(n) = \lg n$
- Note that $\lg^i n = (\lg n)^i \neq \lg^{(i)} n$
- Because the logarithm of a nonpositive number is undefined, $\lg^{(i)} n$ is defined only if $\lg^{(i-1)} n > 0$
- The iterated logarithm function, is defined as

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

  - $\lg^* 2 = 1$ $\lg^* 4 = 2$ $\lg^* 16 = 3$
  - $\lg^* 65536 = 4$ $\lg^* 2^{65536} = 5$
  - a very slowly growing function

Fibonacci numbers$_{1/2}$

- The **Fibonacci numbers** are defined by the recurrence relation

$$\begin{aligned} F_0 &= 0, \\ F_1 &= 1, \\ F_i &= F_{i-1} + F_{i-2} \quad \text{for} \quad i \geq 2. \end{aligned}$$

- The Fibonacci numbers are : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …
- Fibonacci numbers are related to the **golden ratio** ϕ and to its conjugate $\hat{\phi}$.
- One can prove that
  - $F_i = \dfrac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$, where $\phi = \dfrac{1+\sqrt{5}}{2} = 1.61803\ldots$ and $\hat{\phi} = \dfrac{1-\sqrt{5}}{2} = -.61803\ldots$

Fibonacci numbers$_{2/2}$

- Since $|\hat{\phi}| < 1$, we have $\frac{\hat{\phi}^i}{\sqrt{5}} < \frac{1}{\sqrt{5}} < \frac{1}{2}$.
- So that the *i*th Fibonacci number $F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$ is equal to $\frac{\phi^i}{\sqrt{5}}$ round to the nearest integer.
- Thus, Fibonacci number grow exponentially.