Chapter 3: Characterizing Running Times

Chapter 3 overview

- A way to describe behavior of functions *in the limit*. We're studying ***asymptotic*** efficiency.
- Describe *growth* of functions.
- Focus on what's important by abstracting away low-order terms and constant factors.
- How we indicate running times of algorithms.
- A way to compare "sizes" of functions:

  $O \approx \leq$
  $\Omega \approx \geq$
  $\Theta \approx =$
  $o \approx <$
  $\omega \approx >$

Orders of Growth

n	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10	$3.3*10$	10^2	10^3	$\sim 10^3$	$\sim 3.6*10^6$
10^2	6.6	10^2	$6.6*10^2$	10^4	10^6	$\sim 1.3*10^{30}$	$\sim 9*10^{157}$
10^3	10	10^3	$1.0*10^4$	10^6	10^9	…	…
10^4	13	10^4	$1.3*10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7*10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0*10^7$	10^{12}	10^{18}		

Efficient

O-notation, Ω-notation, and Θ-notation

[Section 3.1 does not go over formal definitions of O-notation, Ω-notation, and Θ-notation, but is intended to informally introduce the three most commonly used types of asymptotic notation and show how to use these notations to reason about the worst-case running time of insertion sort.]

O-notation

O-notation characterizes an *upper bound* on the asymptotic behavior of a function: it says that a function grows *no faster* than a certain rate. This rate is based on the highest order term.

For example, $f(n) = 7n^3 + 100n^2 - 20n + 6$ is $O(n^3)$, since the highest order term is $7n^3$, and therefore the function grows no faster than n^3.

Te function $f(n)$ is also $O(n^5)$, $O(n^6)$, and $O(n^c)$ for any constant $c \geq 3$.

Ω-notation

Ω-notation characterizes a *lower bound* on the asymptotic behavior of a function: it says that a function grows *at least as fast* as a certain rate. This rate is again based on the highest-order term.

For example, $f(n) = 7n^3 + 100n^2 - 20n + 6$ is $\Omega(n^3)$, since the highest-order term, n^3, grows at least as fast as n^3.
The function $f(n)$ is also $\Omega(n^2)$, $\Omega(n)$, and $\Omega(n^c)$ for any constant $c \leq 3$.

Θ-notation

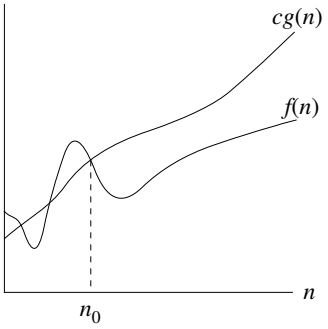
Θ-notation characterizes a *tight bound* on the asymptotic behavior of a function: it says that a function grows *precisely* at a certain rate, again based on the highest-order term.

If a function is both $O(f(n))$ and $\Omega(f(n))$, then a function is $\Theta(f(n))$.

Asymptotic notation: formal definitions

O-notation

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



$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

If $f(n) \in O(g(n))$, we write $f(n) = O(g(n))$ (will precisely explain this soon).

Example

$2n^2 = O(n^3)$, with $c = 1$ and $n_0 = 2$.

Examples of functions in $O(n^2)$:

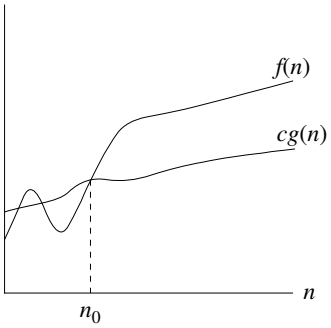
n^2
$n^2 + n$
$n^2 + 1000n$
$1000n^2 + 1000n$

Also,

n
$n/1000$
$n^{1.99999}$
$n^2/\lg\lg\lg n$

Ω-notation

$\Omega(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that
$0 \le cg(n) \le f(n)$ for all $n \ge n_0\}$.



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

Example

$\sqrt{n} = \Omega(\lg n)$, with $c = 1$ and $n_0 = 16$.

Examples of functions in $\Omega(n^2)$:

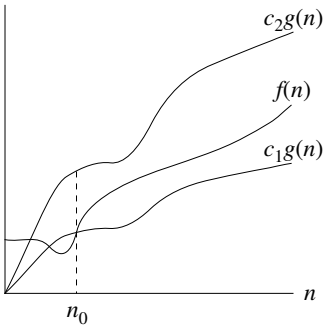
n^2
$n^2 + n$
$n^2 - n$
$1000n^2 + 1000n$
$1000n^2 - 1000n$

Also,

n^3
$n^{2.00001}$
$n^2 \lg\lg\lg n$
2^{2^n}

Θ-notation

$\Theta(g(n)) = \{f(n) :$ 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 $n \ge n_0\}$.



$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

Example

$n^2/2 - 2n = \Theta(n^2)$, with $c_1 = 1/4$, $c_2 = 1/2$, and $n_0 = 8$.

Theorem

$f(n) = \Theta(g(n))$ if and only if $f = O(g(n))$ and $f = \Omega(g(n))$.

Leading constants and low-order terms don't matter.

Can express a constant factor as $O(1)$ or $\Theta(1)$, since it's within a constant factor of 1.

Asymptotic notation and running times

Need to be careful to use asymptotic notation correctly when characterizing a running time. Asymptotic notation describes functions, which in turn describe running times. Must be careful to specify *which* running time.

For example, the worst-case running time for insertion sort is $O(n^2)$, $\Omega(n^2)$, and $\Theta(n^2)$; all are correct. Prefer to use $\Theta(n^2)$ here, since it's the most precise. The best-case running time for insertion sort is $O(n)$, $\Omega(n)$, and $\Theta(n)$; prefer $\Theta(n)$.

But *cannot* say that the running time for insertion sort is $\Theta(n^2)$, with "worst-case" omitted. Omitting the case means making a blanket statement that covers *all* cases, and insertion sort does *not* run in $\Theta(n^2)$ time in all cases.

Can make the blanket statement that the running time for insertion sort is $O(n^2)$, or that it's $\Omega(n)$, because these asymptotic running times are true for all cases.

For merge sort, its running time is $\Theta(n \lg n)$ in all cases, so it's OK to omit which case.

Common error: conflating O-notation with Θ-notation by using O-notation to indicate an asymptotically tight bound. O-notation gives only an asymptotic upper

bound. Saying "an $O(n \lg n)$-time algorithm runs faster than an $O(n^2)$-time algorithm" is not necessarily true. An algorithm that runs in $\Theta(n)$ time also runs in $O(n^2)$ time. If you really mean an asymptotically tight bound, then use Θ-notation.

Use the simplest and most precise asymptotic notation that applies. Suppose that an algorithm's running time is $3n^2 + 20n$. Best to say that it's $\Theta(n^2)$. Could say that it's $O(n^3)$, but that's less precise. Could say that it's $\Theta(3n^2 + 20n)$, but that obscures the order of growth.

Asymptotic notation in equations

When on right-hand side

$O(n^2)$ stands for some anonymous function in the set $O(n^2)$.

$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means $2n^2 + 3n + 1 = 2n^2 + f(n)$ *for some* $f(n) \in \Theta(n)$. In particular, $f(n) = 3n + 1$.

Interpret the number of anonymous functions as equaling the number of times the asymptotic notation appears:

$\sum_{i=1}^{n} O(i)$ OK: 1 anonymous function

$O(1) + O(2) + \cdots + O(n)$ not OK: n hidden constants
$\Rightarrow$ no clean interpretation

When on left-hand side

No matter how the anonymous functions are chosen on the left-hand side, there is a way to choose the anonymous functions on the right-hand side to make the equation valid.

Interpret $2n^2 + \Theta(n) = \Theta(n^2)$ as meaning *for all* functions $f(n) \in \Theta(n)$, there exists a function $g(n) \in \Theta(n^2)$ such that $2n^2 + f(n) = g(n)$.

Can chain together:

$$\begin{aligned} 2n^2 + 3n + 1 &= 2n^2 + \Theta(n) \\ &= \Theta(n^2) \ . \end{aligned}$$

Interpretation:

- First equation: There exists $f(n) \in \Theta(n)$ such that $2n^2 + 3n + 1 = 2n^2 + f(n)$.
- Second equation: For all $g(n) \in \Theta(n)$ (such as the $f(n)$ used to make the first equation hold), there exists $h(n) \in \Theta(n^2)$ such that $2n^2 + g(n) = h(n)$.

Proper abuses of asymptotic notation

It's usually clear what variable in asymptotic notation is tending toward ∞: in $O(g(n))$, looking at the growth of $g(n)$ as n grows.

What about $O(1)$? There's no variable appearing in the asymptotic notation. Use the context to disambiguate: in $f(n) = O(1)$, the variable is n, even though it does not appear in the right-hand side of the equation.

Subtle point: asymptotic notation in recurrences

Often abuse asymptotic notation when writing recurrences: $T(n) = O(1)$ for $n < 3$. Strictly speaking, this statement is meaningless. Definition of O-notation says that $T(n)$ is bounded above by a constant $c > 0$ for $n \geq n_0$, for some $n_0 > 0$. The value of $T(n)$ for $n < n_0$ might not be bounded. So when we say $T(n) = O(1)$ for $n < 3$, cannot determine any constraint on $T(n)$ when $n < 3$ because could have $n_0 > 3$.

What we really mean is that there exists a constant $c > 0$ such that $T(n) \leq c$ for $n < 3$. This convention allows us to avoid naming the bounding constant so that we can focus on the more important part of the recurrence.

Asymptotic notation defined for only subsets

Suppose that an algorithm assumes that its input size is a power of 2. Can still use asymptotic notation to describe the growth of its running time. In general, can use asymptotic notation for $f(n)$ defined on only a subset of $\mathbb{N}$ or $\mathbb{R}$.

o-notation

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

Another view, probably easier to use: $\lim_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$.

$n^{1.9999} = o(n^2)$
$n^2/\lg n = o(n^2)$
$n^2 \neq o(n^2)$ (just like $2 \not< 2$)
$n^2/1000 \neq o(n^2)$

ω-notation

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

Another view, again, probably easier to use: $\lim_{n\to\infty} \dfrac{f(n)}{g(n)} = \infty$.

$n^{2.0001} = \omega(n^2)$
$n^2 \lg n = \omega(n^2)$
$n^2 \neq \omega(n^2)$

Comparisons of functions

Relational properties: **Transitivity:**

$f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$.
Same for O, Ω, o, and ω.

Reflexivity:

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

Same for O and Ω.

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

Comparisons:

- $f(n)$ is ***asymptotically smaller*** than $g(n)$ if $f(n) = o(g(n))$.
- $f(n)$ is ***asymptotically larger*** than $g(n)$ if $f(n) = \omega(g(n))$.

No trichotomy. Although intuitively, we can liken O to $\leq$, Ω to $\geq$, etc., unlike real numbers, where $a < b$, $a = b$, or $a > b$, we might not be able to compare functions.

Example: $n^{1+\sin n}$ and n, since $1 + \sin n$ oscillates between 0 and 2.

Standard notations and common functions

Monotonicity

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

Exponentials

Useful identities:

$$a^{-1} = 1/a ,$$
$$(a^m)^n = a^{mn} ,$$
$$a^m a^n = a^{m+n} .$$

Can relate rates of growth of polynomials and exponentials: for all real constants a and b such that $a > 1$,

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0 ,$$

which implies that $n^b = o(a^n)$.

A suprisingly useful inequality: for all real x,

$$e^x \geq 1 + x .$$

As x gets closer to 0, e^x gets closer to $1 + x$.

Logarithms

Notations:

$$\begin{aligned} \lg n &= \log_2 n && \text{(binary logarithm)} , \\ \ln n &= \log_e n && \text{(natural logarithm)} , \\ \lg^k n &= (\lg n)^k && \text{(exponentiation)} , \\ \lg \lg n &= \lg(\lg n) && \text{(composition)} . \end{aligned}$$

Logarithm functions apply only to the next term in the formula, so that $\lg n + k$ means $(\lg n) + k$, and *not* $\lg(n + k)$.

In the expression $\log_b a$:

- Hold b constant $\Rightarrow$ the expression is strictly increasing as a increases.
- Hold a constant $\Rightarrow$ the expression is strictly decreasing as b increases.

Useful identities for all real $a > 0$, $b > 0$, $c > 0$, and n, and where logarithm bases are not 1:

$$\begin{aligned} a &= b^{\log_b a} , \\ \log_c(ab) &= \log_c a + \log_c b , \\ \log_b a^n &= n \log_b a , \\ \log_b a &= \frac{\log_c a}{\log_c b} , \\ \log_b(1/a) &= -\log_b a , \\ \log_b a &= \frac{1}{\log_a b} , \\ a^{\log_b c} &= c^{\log_b a} . \end{aligned}$$

[For the last equality, can show by taking $\log_b$ *of both sides:*

$$\begin{aligned} \log_b a^{\log_b c} &= (\log_b c)(\log_b a) , \\ \log_b c^{\log_b a} &= (\log_b a)(\log_b c) . \ ] \end{aligned}$$

Changing the base of a logarithm from one constant to another only changes the value by a constant factor, so we usually don't worry about logarithm bases in asymptotic notation. Convention is to use lg within asymptotic notation, unless the base actually matters.

Just as polynomials grow more slowly than exponentials, logarithms grow more slowly than polynomials. In $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$, substitute $\lg n$ for n and 2^a for a:

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

implying that $\lg^b n = o(n^a)$.

Factorials

$n! = 1 \cdot 2 \cdot 3 \cdot n$. Special case: $0! = 1$.

Can use ***Stirling's approximation***,

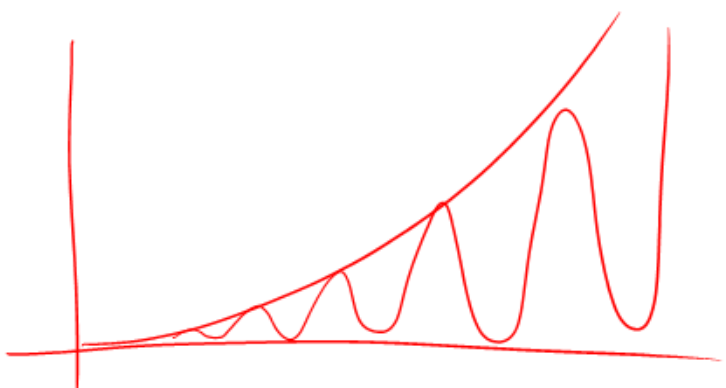
$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right) ,$$

to derive that $\lg(n!) = \Theta(n \lg n)$.

Some additional observations

Distinguishing O from Θ

- Can you draw a function that is in $O(n^2)$ but not $\Theta(n^2)$?



Example

- Show that $\frac{1}{2}n^2 - 3n \in \Theta(n^2)$
- Must find positive constants c_1, c_2, n_0 such that

$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2 \quad \forall n \geq n_0$$

- Divide through by n^2 to get $c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2 \quad \forall n \geq n_0$
- Hint: consider one side of the inequality at a time:
  - for n_0=7 we have $0 < c_1 \leq \frac{1}{14}$
  - for n_0=7 we can chose $c_2 \geq \frac{1}{2}$
- This proof is *constructive*

Another Example

Show that $6n^3 \notin \Theta(n^2)$

- Use proof by contradiction: assume that $6n^3 \in \Theta(n^2)$
- Suppose positive constants c_2, n_0 exist such that
$$6n^3 \leq c_2 n^2 \ \forall n \geq n_0$$
- But this implies that $n \leq \frac{c_2}{6} \ \ \forall n \geq n_0$
- i.e., n is bounded by $\frac{c_2}{6}$, a constant
- But n is unbounded; hence we have a contradiction.
- Thus, our assumption was false; hence we have shown that $6n^3 \notin \Theta(n^2)$

Duality

$$f(n) \in \Omega(g(n)) \text{ if and only if } g(n) \in O(f(n))$$

g(n) in O(f(n)) means *all* instances of size *n* are solvable within $c_1 f(n)$ time.

f(n) in Ω(g(n)) means *at least one* instance of size n is solvable in $c_2 g(n)$ time.

The Limit Rule

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} \begin{cases} \in R^+ & \text{then } f(n) \in O(g(n)) \wedge g(n) \in O(f(n)) \\ = 0 & \text{then } f(n) \in O(g(n)) \wedge g(n) \notin O(f(n)) \\ = +\infty & \text{then } f(n) \notin O(g(n)) \wedge g(n) \in O(f(n)) \end{cases}$$

BY DUALITY

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

HENCE, $f(n) \in \Theta(g(n))$

Example

$n^2 \in O(n^3)$

$f(n)$ $g(n)$

Apply Limit Rule

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{1}{n} = 0$$

$f(n) \in O(g(n))$ By The Limit Rule

Useful Identity: L'Hopital's Rule

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

Applicable when:

- $\lim f(n) = \lim g(n) = 0$
- $\lim f(n) = \lim g(n) = \infty$

As $n \to \infty$

Review: More Logarithms

IDENTITY:

$$\frac{d}{dn} \log(n) = \frac{1}{n}$$

EXAMPLE:

$$\lim_{n \to \infty} \frac{\log n}{n} = \lim_{n \to \infty} \frac{1/n}{1} = \lim_{n \to \infty} \frac{1}{n} = 0$$

Functional Iteration

$f^{(i)}(n)$ denotes: The function *f(n)* iteratively applied i times to an initial value of n.

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

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

$f^{(0)}(n) = n$

$f^{(1)}(n) = f(n) = 2n$

$f^{(2)}(n) = f(f^{(1)}n) = f(2n) = 2(2n) = 2^2 n$

.

.

$f^{(i)}(n) = 2^i n$

The iterated logarithm function

$\log^*n$ (log star of n) denotes the iterated logarithm, defined as follows. Let $\lg^{(i)}n$ be as denoted on the previous page, with $f(n) = \lg n$. $\lg^{(i)}n$ is defined only if $\lg^{(i-1)}n > 0$.
Distinguish the $\lg^{(i)}n$ (the logarithm function applied i times in succession, starting with argument n) from the $\lg^i n$ (the logarithm of n raised to the *ith* power).
Then

$$\lg^*n = \min\{i \geq 0 : \lg^{(i)}n \leq 1\} \quad \textit{minimum } i \textit{ such that } \lg^{(i)}n \leq 1$$

The iterated logarithm is a very slowly growing function:

$\lg^*2 = 1$,	Since $\lg^{(1)}2 = 1$	=> i =1
$\lg^*4 = 2$,	Since $\lg^{(2)}4 = \lg\lg^{(1)}4 = \lg 2 = 1$	=> i = 2
$\lg^*16 = 3$,	Since $\lg^{(3)}16 = \lg\lg\lg 16 = \lg\lg 4 = \lg 2 = 1$	=> i = 3
$\lg^*(2^{16}) = 4$,	.	
$\lg^*(2^{65536}) = 5$	.	

Fibonacci Numbers

Remember that Fibonacci numbers are defined as:

$F_0 = 1$,

$F_1 = 1$,

$F_i = F_{i-1} + F_{i-2}$ for $i \geq 2$

Fibonacci numbers are related to the **golden ratio** Φ and to its **conjugate** $\hat{\Phi}$, which are the two roots of the following equation:

$$x^2 = x + 1$$

$\Phi = \frac{1+\sqrt{5}}{2} = 1.61803\ldots$

$\hat{\Phi} = \frac{1-\sqrt{5}}{2} = -0.61803\ldots$

Specifically, we have

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$$

Since $|\hat{\Phi}| < 1$, we have

$$\frac{|\hat{\phi}^i|}{\sqrt{5}} < \frac{1}{\sqrt{5}} < \frac{1}{2}$$

Which implies that

$$F_i = \left\lfloor \frac{\phi^i}{\sqrt{5}} + \frac{1}{2} \right\rfloor$$

Which states that F_i is equal to $\frac{\phi^i}{\sqrt{5}}$ rounded to the nearest integer. Thus, Fibonacci numbers grow exponentially.