ICS 311 Topic #3: Growth of Functions and Asymptotic Concepts

Outline

1. Intro to Asymptotic Analysis
2. Big-O
3. Ω (Omega)
4. Θ (Theta)
5. Asymptotic Notation in Equations
6. Asymptotic Inequality
7. Properties of Asymptotic Sets
8. Common Functions

Readings and Screencasts

- Chapter 3 of CLRS
- Screencasts: 3A, 3B, 3C, and 3D (also available in Laulima and iTunesU)
  These screencasts have some audio problems, but they are usable. The audio problems were resolved (and other aspects of the production improved) in a few weeks.

Intro to Asymptotic Analysis

The notations discussed today are ways to describe behaviors of *functions*, particularly *in the limit*, or *asymptotic* behavior.

The functions need not necessarily be about algorithms, and indeed asymptotic analysis is used for many other applications.

Asymptotic analysis of algorithms requires:

1. Identifying **what aspect of an algorithm we care about**, such as:
   - runtime;
   - use of space;
   - possibly other attributes such as communication bandwidth;
2. Identifying **a function that characterizes that aspect;** and
3. Identifying **the asymptotic class of functions that this function belongs to**, where classes are defined in terms of bounds on growth rate.

The different asymptotic bounds we use are analogous to equality and inequality relations:

- $O \approx \leq$
- $\Omega \approx \geq$
- $\Theta \approx =$

- o ≈ <
- ω ≈ >

In practice, most of our analyses will be concerned with run time. Analyses may examine:

- Worst case
- Best case
- Average case (according to some probability distribution across all possible inputs)

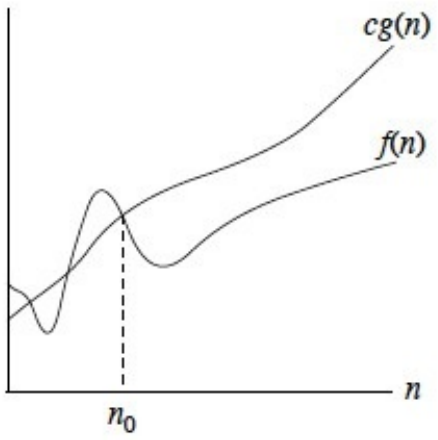
Big-O (asymptotic ≤)

Our first question about an algorithm's run time is often "how bad can it get?" We want a guarantee that a given algorithm will complete within a reasonable amount of time for typical n expected. This requires an **asymptotic upper bound**: the "worst case".

Big-O is commonly used for worst case analyses, because it gives an upper bound on growth rate. Its definition in terms of set notation is:

$$O(g(n)) = \{f(n) : \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le c\,g(n)\ \forall n \ge n_0\}.$$

This definition means that as n increases, afer a point $f(n)$ grows no faster than $g(n)$ (as illustrated in the figure): $g(n)$ is an *asymptotic upper bound* for $f(n)$.



Since $O(g(n))$ is a set, it would be natural to write $f(n) \in O(g(n))$ for any given $f(n)$ and $g(n)$ meeting the definition above, for example, $f \in O(n^2)$.

But the algorithms literature has adopted the convention of using = instead of $\in$, for example, writing $f(n) = O(g(n))$. This "abuse of notation" makes some manipulations possible that would be more tedious if done strictly in terms of set notation. (We do *not* write $O(g(n))=f(n)$; will return to this point).

Using the = notation, we often see definitions of big-O in in terms of truth conditions as follows:

$f(n) = O(g(n))$ iff $\exists$ positive constants c and n_0 such that $0 \le f(n) \le c\,g(n)\ \forall n \ge n_0$.

We assume that all functions involved are asymptotically non-negative. Other authors don't make this assumption, so may use $|f(n)|$ etc. to cover negative values. This assumption is reflected in the condition $0 \le f(n)$.

Examples

Show that $2n^2$ is $O(n^2)$.

To do this we need to show that there exists some c and n_0 such that (letting $2n^2$ play the role of $f(n)$ and n^2 play the role of $g(n)$ in the definition):

$0 \le 2n^2 \le cn^2$ for all $n \ge n_0$.

It works with $c = 2$, since this makes the f and g terms equivalent for all $n \geq n_0 = 0$. (We'll do a harder example under Θ.)

What's in and what's out

These are all $O(n^2)$:

- n^2
- $n^2 + 1000n$
- $1000n^2 + 1000n$
- $n^{1.99999}$
- n

These are not:

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

Insertion Sort Example

Recall that we did a tedious analysis of the worst case of insertion sort, ending with this formula:

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n \\ &\quad - (c_2 + c_4 + c_5 + c_8)\,. \end{aligned}$$

$T(n)$ can be expressed as $pn^2 + qn - r$ for suitable p, q, r ($p = (c_5/2 + c_6/2 + c_7/2)$, etc.).

The textbook (page 46) sketches a proof that $f(n) = an^2 + bn + c$ is $\Theta(n^2)$, and we'll see shortly that $\Theta(n^2) \rightarrow O(n^2)$. This is generalized to all polynomials in Problem 3-1. So, any polynomial with highest order term an^d (i.e., a polynomial in n of degree d) will be $O(n^d)$.

This suggests that the worst case for insertion sort $T(n) \in O(n^2)$. An upper bound on the worst case is also an upper bound on all other cases, so we have already covered those cases.

Notice that the definition of big-O would also work for $g(n) = n^3$, $g(n) = 2^n$, etc., so we can also say that $T(n)$ (the worst case for insertion sort) is $O(n^3)$, $O(2^n)$, etc. However, these loose bounds are not very useful! We'll deal with this when we get to Θ (Theta).

Ω (Omega, asymptotic ≥)

We might also want to know what the best we can expect is. In the last lecture we derived this formula for insertion sort:

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)\,. \end{aligned}$$

We could prove that this best-case version of T(n) is big-O of something, but that would only tell us that the best case is no worse than that something. What if we want to know what is "as good as it gets": a lower bound below which the algorithm will never be any faster?

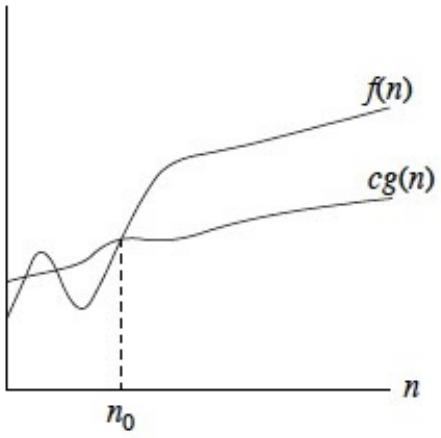
We must both pick an appropriate function to measure the property of interest, and pick an appropriate asymptotic class or comparison to match it to. We've done the former with $T(n)$, but what should it be compared to?

It makes more sense to determine the **asymptotic lower bound** of growth for a function describing the best case run-time. In other words, what's the fastest we can ever expect, in the best case?

Ω (Omega) provides what we are looking for. Its set and truth condition definitions are simple revisions of those for big-O:

$\Omega(g(n)) = \{f(n) : \exists$ positive constants c and n_0 such that $0 \le cg(n) \le f(n) \forall n \ge n_0\}$.
[The $f(n)$ and $cg(n)$ have swapped places.]

$f(n) = \Omega(g(n))$ iff $\exists$ positive constants c and n_0 such that $f(n) \ge cg(n) \forall n \ge n_0$.
[$\le$ has been replaced with $\ge$.]



The semantics of Ω is: as n increases after a point, $f(n)$ grows no slower than $g(n)$ (see illustration).

Examples

Sqrt(n) is $\Omega(\lg n)$ with c=1 and n_0=16.
(At n=16 the two functions are equal; try at n=64 to see the growth, or graph it.)

What's In and What's Out

These are all $\Omega(n^2)$:

- n^2
- $n^2 + 1000n$ *(It's also $O(n^2)$!)*
- $1000n^2 + 1000n$
- $1000n^2 - 1000n$
- n^3
- $n^{2.00001}$

These are not:

- $n^{1.99999}$
- n
- $\lg n$

Insertion Sort Example

We can show that insertion will take at least $\Omega(n)$ time in the best case (i.e., it won't get any better than this) using the above formula and definition.

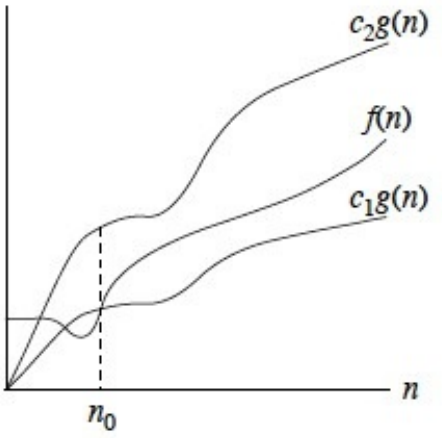
$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

$T(n)$ can be expressed as $pn - q$ for suitable p, q (e.g., $q = c_2 + c_4 + c_5 + c_8$, etc.). (In this case, p and q are positive.) This suggests that $T(n) \in \Omega(n)$, that is, $\exists\, c, n_0$ s.t. $pn - q \geq cn, \forall n \geq n_0$. This follows from the generalized proof for polynomials.

Θ (Theta, asymptotic =)

We noted that there are *loose* bounds, such as $f(n) = n^2$ is $O(n^3)$, etc., but this is an overly pessimistic assessment. It is more useful to have an **asymptotically tight bound** on the growth of a function. In terms of algorithms, we would like to be able to say (when it's true) that a given characteristic such as run time grows *no better and no worse* than a given function. That is, we want to simultaneoulsy bound from above and below. Combining the definitions for O and Ω:

> $\Theta(g(n)) = \{f(n)$: ∃ positive constants $\boldsymbol{c_1}$, $\boldsymbol{c_2}$, and n_0 such that $0 \leq \boldsymbol{c_1 g(n) \leq f(n) \leq c_2 g(n)}, \forall n \geq n_0\}$.



As illustrated, $g(n)$ is an asymptotically tight bound for $f(n)$: after a point, $f(n)$ grows no faster and no slower than $g(n)$.

The book suggests the proof of this theorem as an easy exercise (just combine the two definitions):

> $f(n) = \Theta(g(n))$ iff $f(n) = \Omega(g(n)) \ \wedge \ f(n) = O(g(n))$.

You may have noticed that some of the functions in the list of examples for big-O are also in the list for Ω. This indicates that the set Θ is not empty.

Examples

> *Reminder:* $f(n) = \Theta(g(n))$ iff ∃ positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \forall n \geq n_0$.

$n^2 - 2n$ is $\Theta(n^2)$, with $c_1 = 1/2$; $c_2 = 1$, and $n_0 = 4$, since:

> $n^2/2 \ \leq \ n^2 - 2n \ \leq \ n^2$ for $n \geq n_0 = 4$.

Find an asymptotically tight bound (Θ) for

- $4n^3$
- $4n^3 + 2n$.

Please try these before you find solutions here.

What's in and what's out

These are all $\Theta(n^2)$:

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

These are not

- n^3

- $1000n^2 + 1000n + 32{,}700$
- $1000n^2 - 1000n - 1{,}048{,}315$

- $n^{2.00001}$
- $n^{1.99999}$
- $n \lg n$

Asymptotic Inequality

The O and Ω bounds may or may not be asymptotically tight. The next two notations are for upper bounds that are strictly *not* asymptotically tight. There is an *analogy* to inequality relationships:

- "≤" is to "<" as big-O (may or may not be tight) is to little-o (strictly not equal)
- "≥" is to ">" as Ω (may or may not be tight) is to little-ω (strictly not equal).

o-notation ("little o", asymptotic <)

$$o(g(n)) = \{f(n) : \forall \text{constants } c > 0, \exists \text{ constant } n_0 > 0 \text{ such that } 0 \le f(n) < c\,g(n)\ \forall n \ge n_0\}.$$

Alternatively, $f(n)$ becomes *insignificant* relative to $g(n)$ as n approaches infinity (see box):

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

We say that $f(n)$ is **asymptotically smaller** than $g(n)$ if $f(n) = o(g(n))$

- $n^{1.99999} \in o(n^2)$
- $n^2/\lg n \in o(n^2)$
- $n^2 \notin o(n^2)$ (similarly, 2 is not less than 2)
- $n^2/1000 \notin o(n^2)$

ω-notation ("little omega", asymptotic >)

$$\omega(g(n)) = \{f(n) : \forall \text{constants } c > 0, \exists \text{ constant } n_0 > 0 \text{ such that } 0 \le c\,g(n) < f(n)\ \forall n \ge n_0\}.$$

Alternatively, $f(n)$ becomes *arbitrarily large* relative to $g(n)$ as n approaches infinity (see box):

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

We say that $f(n)$ is **asymptotically larger** than $g(n)$ if $f(n) = \omega(g(n))$

- $n^{2.00001} \in \omega(n^2)$
- $n^2 \lg n \in \omega(n^2)$
- $n^2 \notin \omega(n^2)$

The two are related: $\boldsymbol{f(n) \in \omega(g(n))}$ **iff** $\boldsymbol{g(n) \in o(f(n))}$.

Asymptotic Notation in Equations

We already noted that while asymptotic categories such as $\Theta(n^2)$ are sets, we usually use "=" instead of " ∈ " and write (for example) $f(n) = \Theta(n^2)$ to indicate that f is in this set.

Putting asymptotic notation in equations lets us do shorthand manipulations during analysis.

Asymptotic Notation on Right Hand Side: ∃

$O(g(x))$ on the right hand side stands for *some* anonymous function in the set $O(g(x))$.

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

Asymptotic Notation on Left Hand Side: ∀

The notation is only used on the left hand side when it is also on the right hand side.

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

$2n^2 + \Theta(n) = \Theta(n^2)$ means for *all* $f(n) \in \Theta(n)$, there *exists* $g(n) \in \Theta(n^2)$ such that $2n^2 + f(n) = g(n)$.

Combining Terms

We can do basic algebra such as:

$2n^2 + 3n + 1 = 2n^2 + \Theta(n) = \Theta(n^2)$

Properties

If we keep in mind the analogy to inequality, many of these make sense, but see the end for a caution concerning this analogy.

Relational Properties

Transitivity:

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

Reflexivity:

- $f(n) = \Theta(f(n))$
- $f(n) = O(f(n))$
- $f(n) = \Omega(f(n))$
- *What about o and ω?*

Symmetry:

- $f(n) = \Theta(g(n))$ iff $g(n) = \Theta(f(n))$
- *Should any others be here? Why or why not?*

Transpose Symmetry:

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

Incomparability

Here is where the analogy to numeric (in)equality breaks down: There is no trichotomy. Unlike with constant numbers, we can't assume that one of <, =, > hold. Some functions may be incomparable.

Example: $n^{1+\sin n}$ is incomparable to n since $\sin n$ oscillates between -1 and 1, so $1 + \sin n$ oscillates between 0 and 2. *(Try graphing it.)*

Common Functions and Useful Facts

Various classes of functions and their associated notations and identities are reviewed in the end of the chapter: please review the chapter and refer to ICS 241 if needed. Here we highlight some useful facts:

Monotonicity

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

Polynomials

- $p(n) = \Theta(n^d)$, for asymptoptically positive polynomials in n of degree d

Exponentials

- $n^b = o(a^n)$ for all real constants a and b such that $a > 1$: ***Any exponential function with a base greater than 1 grows faster than any polynomial function.***

- Useful identities:
  - $a^{-1} = 1/a$
  - $(a^m)^n = a^{mn}$
  - $a^m a^n = a^{m+n}$

Logarithms

- $(\lg n)^b = \lg^b n = o(n^a)$, for a > 0: ***any positive polynomial function grows faster than any polylogarithmic function.***

- Useful identities:
  - $a = b^{\log_b a}$ *(Definition of logs.)*
  - $\log_a n = \log_b n / \log_b a$
    (Base change. If n is variable and a and b are constant, the denominator is constant: this is why asymptotic analysis can ignore the base.)

- $\log_c(ab) = \log_c a + \log_c b$ *(Ask your slide rule!)*
- $\log_b a^n = n \log_b a$
- $\log_b(1/a) = -\log_b a$
- $\log_b a = 1 / \log_a b$
- $a^{\log_b c} = c^{\log_b a}$ *(Useful for getting the variable where you want it.)*

Factorials

- $n! = \omega(2^n)$: ***factorials grow faster than exponentials*** *(but it could be worse):*
- $n! = o(n^n)$
- $\lg(n!) = \Theta(n \lg n)$
- See also the more complex **Stirling's approximation** from which these are derived.

Iterated Functions

- Definition: $f^{(i)}(n)$ is f applied i times to the initial value n.
- Iterated Logarithm: $\lg^* n = \min\{i \geq 0: \lg^{(i)} n \leq 1\}$ *(The iteration at which $\lg^{(i)} n$ is less than 1: a very slowly growing function.)*

Fibonacci Numbers

- Definition: $F_0 = 0$; $F_1 = 1$; and for $i > 1$ $F_i = F_{i-1} + F_{i-2}$.
- ***Fibonacci numbers grow exponentially.***

Dan Suthers
Last modified: Sat Jan 25 03:51:57 HST 2014
Images are from the instructor's manual for Cormen et al.