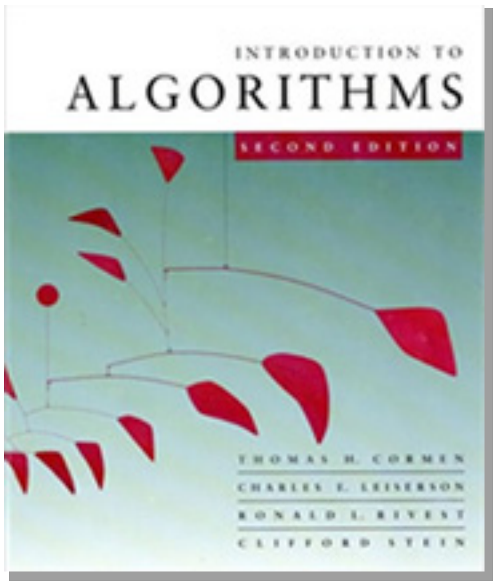
Introduction to Algorithms

6.046J/18.401J



LECTURE 2

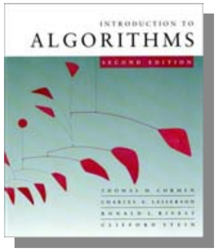
Asymptotic Notation

- O-, Ω-, and Θ-notation

Recurrences

- Substitution method
- Iterating the recurrence
- Recursion tree
- Master method

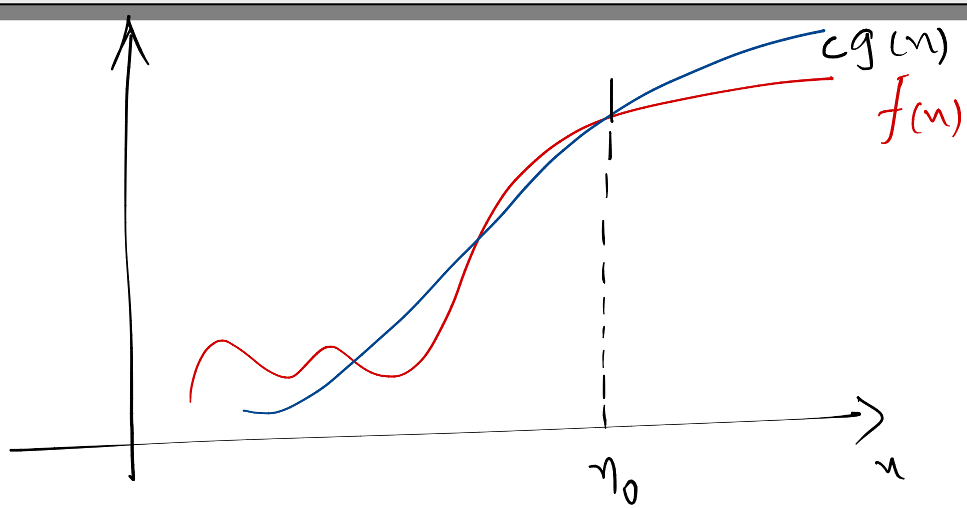
Prof. Erik Demaine

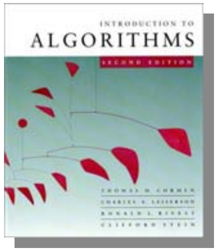


Asymptotic notation

O-notation (upper bounds):

We write $\boxed{f(n) = O(g(n))}$ if there exist constants $c > 0$, $n_0 > 0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$.



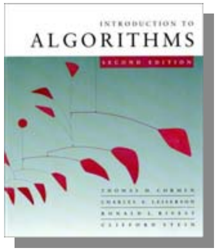


Asymptotic notation

O-notation (upper bounds):

> We write $f(n) = O(g(n))$ if there exist constants $c > 0$, $n_0 > 0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$.

EXAMPLE: $\underbrace{2n^2}_{f(n)} = O(\underbrace{n^3}_{g(n)})$ $\quad (c = 1, n_0 = 2)$



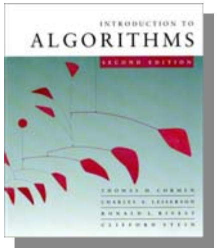
Asymptotic notation

O-notation (upper bounds):

We write $f(n) = O(g(n))$ if there exist constants $c > 0$, $n_0 > 0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$.

Example: $2n^2 = O(n^3)$ $\quad (c = 1, n_0 = 2)$

functions, not values



Asymptotic notation

O-notation (upper bounds):

We write $f(n) = O(g(n))$ if there exist constants $c > 0$, $n_0 > 0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$.

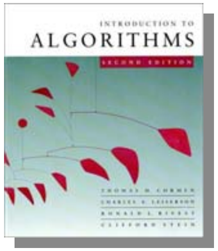
EXAMPLE: $2n^2 = O(n^3)$ $\quad (c = 1, n_0 = 2)$

functions, not values

funny, "one-way" equality

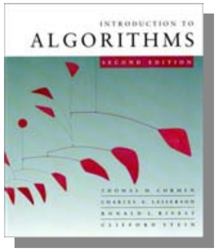
$f(n) = \underline{2n^2} \in O(n^3)$

$\to g(n)$



Set definition of O-notation

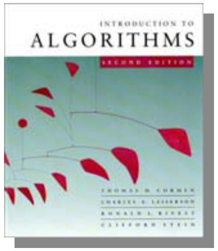
$O(g(n)) = \{ f(n)$: there exist constants $c > 0$, $n_0 > 0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0 \}$



Set definition of O-notation

$O(g(n)) = \{ f(n) :$ there exist constants $c > 0$, $n_0 > 0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$ $\}$

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

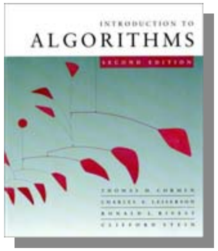


Set definition of O-notation

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

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

(*Logicians:* $\lambda n.2n^2 \in O(\lambda n.n^3)$, but it's convenient to be sloppy, as long as we understand what's *really* going on.)



Macro substitution

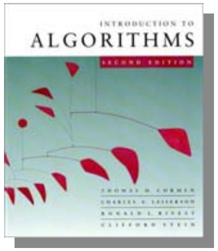
Convention: A set in a formula represents an anonymous function in the set.

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

→ treat like a macro

$\Rightarrow f(n) \in O(g(n))$

→ $h(n)$



Macro substitution

Convention: A set in a formula represents an anonymous function in the set.

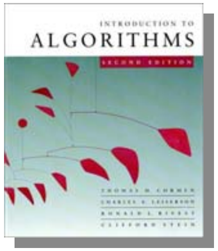
EXAMPLE: $f(n) = n^3 + O(n^2)$

means

Think of $O(n^2)$ like a macro & substitute!

$f(n) = n^3 + h(n)$

for some $h(n) \in O(n^2)$.



Macro substitution

Convention: A set in a formula represents an anonymous function in the set.

EXAMPLE: $n^2 + \overset{f(n)}{O(n)} = \overset{h(n)}{O(n^2)}$

means

for any $f(n) \in O(n)$:

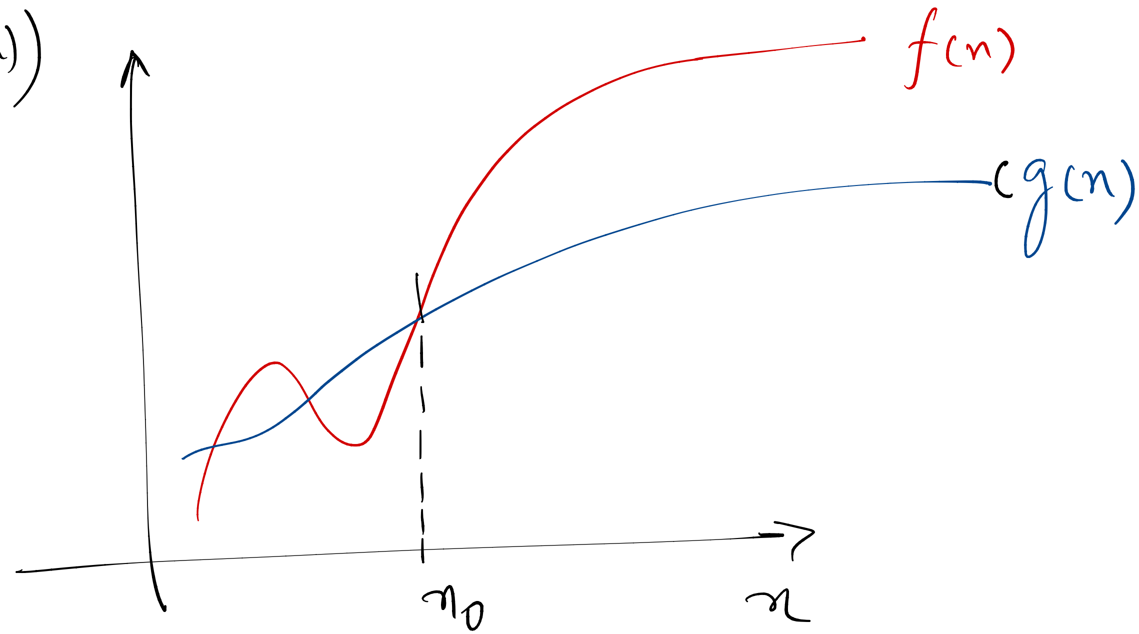
$n^2 + f(n) = h(n)$

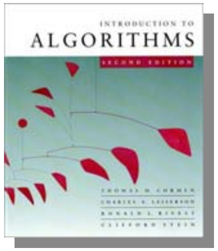
for some $h(n) \in O(n^2)$.

Ω-notation (lower bounds)

O-notation is an *upper-bound* notation. It makes no sense to say $f(n)$ is at least $O(n^2)$.

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

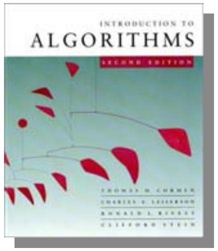




Ω-notation (lower bounds)

O-notation is an *upper-bound* notation. It makes no sense to say $f(n)$ is at least $O(n^2)$.

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



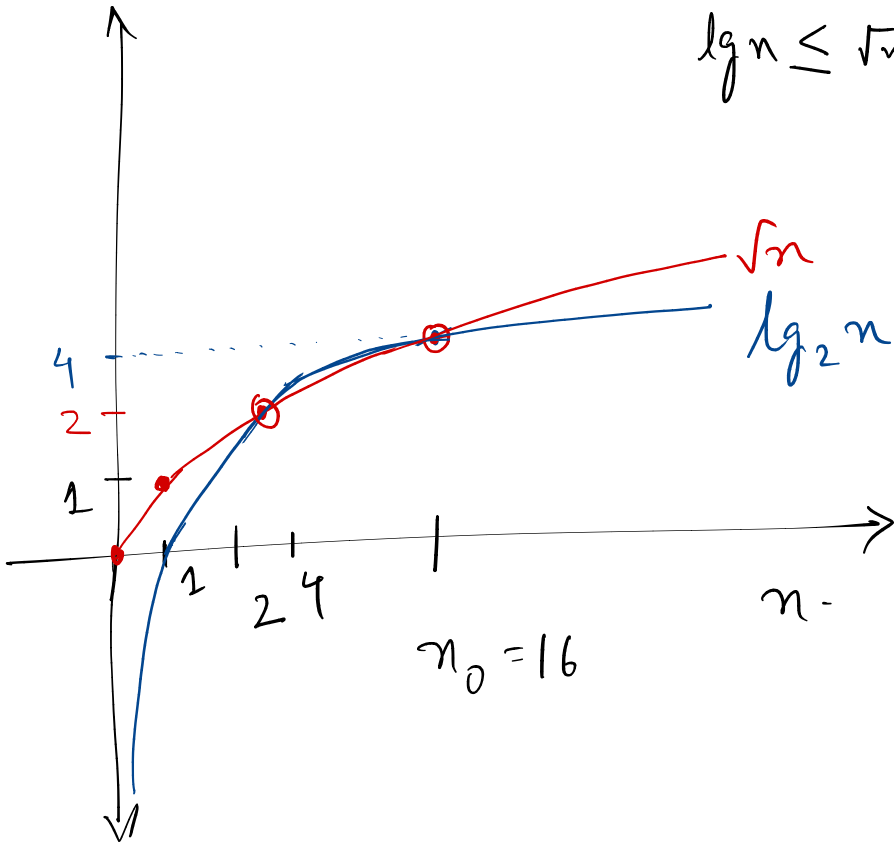
Ω-notation (lower bounds)

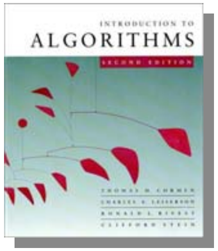
O-notation is an *upper-bound* notation. It makes no sense to say $f(n)$ is at least $O(n^2)$.

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

EXAMPLE: $\sqrt{n} = \Omega(\lg n)$ $\quad (c = 1,\ n_0 = 16)$

$\lg n \leq \sqrt{n}$ for $n_0 \geq 16$



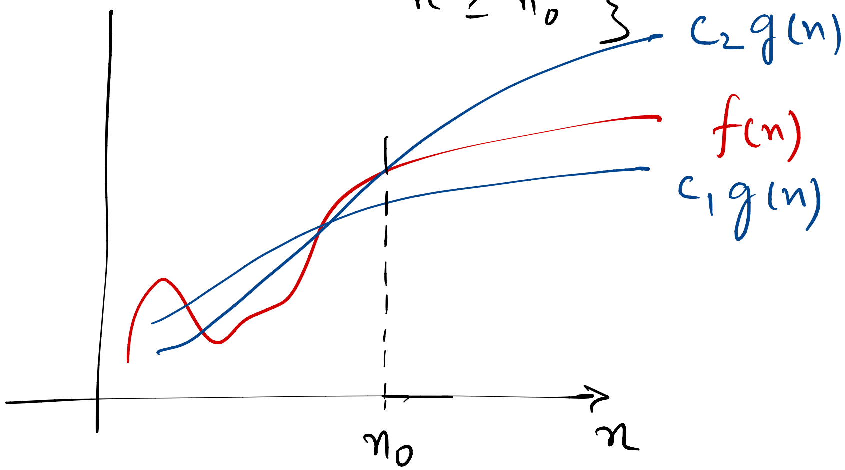


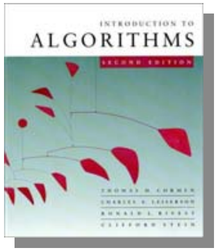
Θ-notation (tight bounds)

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

$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

$\Theta(g(n)) = \{ f(n) :$ There exist constants c_1, c_2 and n_0 for which $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for $n \ge n_0 \}$





Θ-notation (tight bounds)

$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

EXAMPLE: $\frac{1}{2}n^2 - 2n = \Theta(n^2)$

Mnemonic:

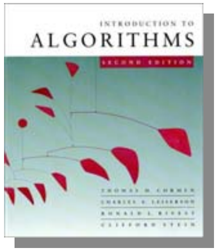
O : $\leq$

Θ : $=$

Ω : $\geq$

o : $<$

ω : $>$



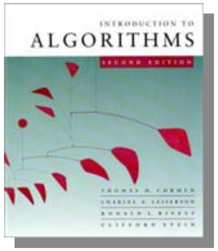
o-notation and ω-notation

O-notation and Ω-notation are like $\leq$ and $\geq$.
o-notation and ω-notation are like $<$ and $>$.

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

EXAMPLE: $2n^2 = o(n^3)$ $\qquad \underbrace{(n_0 = 2/c)}_{\text{Check for correctness}}$ Handle boundary case

$\Rightarrow 2n^2 \in o(n^3)$

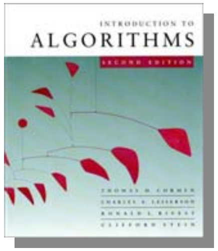


o-notation and ω-notation

O-notation and Ω-notation are like $\le$ and $\ge$.
o-notation and ω-notation are like $<$ and $>$.

$\omega(g(n)) = \{\, f(n) :$ for any constant $c > 0$,
there is a constant $n_0 > 0$
such that $0 \le cg(n) < f(n)$
for all $n \ge n_0$ $\}$

Example: $\sqrt{n} = \omega(\lg n)$ $\quad (n_0 = 1 + 1/c)$

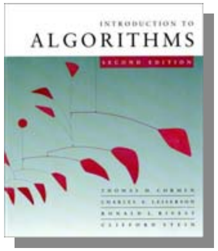


Solving recurrences

- The analysis of merge sort from ***Lecture 1*** required us to solve a recurrence.
- Recurrences are like solving integrals, differential equations, etc. Ex.: Merge Sort:
  - Learn a few tricks.

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

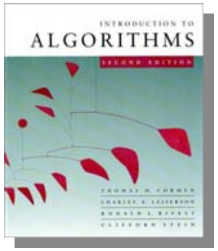
- ***Lecture 3***: Applications of recurrences to divide-and-conquer algorithms.



Substitution method

The most general method:

1. ***Guess*** the form of the solution.
2. ***Verify*** by induction.
3. ***Solve*** for constants.



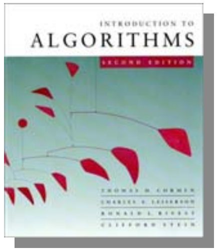
Substitution method

The most general method:

1. ***Guess*** the form of the solution.
2. ***Verify*** by induction.
3. ***Solve*** for constants.

EXAMPLE: $T(n) = 4T(n/2) + n$

- [Assume that $T(1) = \Theta(1)$.]
- Guess $O(n^3)$. (Prove O and Ω separately.)
- Assume that $T(k) \le ck^3$ for $k < n$. I.H. : Inductive Hypothesi
- Prove $T(n) \le cn^3$ by induction.



Example of substitution

I.H: $T(k) \le ck^3$ for $k < n$

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4c(n/2)^3 + n \\
&= (c/2)n^3 + n \\
&= cn^3 - ((c/2)n^3 - n) \quad \longleftarrow \boxed{\textit{desired} - \textit{residual}} \\
&\le cn^3 \quad \longleftarrow \textit{desired}
\end{aligned}
$$

whenever $(c/2)n^3 - n \ge 0$, for example, if $c \ge 2$ and $n \ge 1$.

residual