Design & Analysis of Algorithms
COMP 482 / ELEC 420

John Greiner

Math Background: Review & Beyond

1. Asymptotic notation
2. Math used in asymptotics
3. Recurrences
4. Probabilistic analysis

To do:
[CLRS] 2, 3.1
#1

Asymptotes

Want to describe an algorithm's running time.
(or space, …)

Described by $O(n^2)$, $\Omega(n \log n)$, $\Theta(n)$, …

¿ What do these mean?
Why do we use these notations? ?

Upper Bounds

Creating an algorithm proves we can solve the problem within a given bound.

But another algorithm might be faster.

E.g., sorting an array.
Insertion sort → $O(n^2)$

Lower Bounds

Sometimes can prove that we cannot compute something without a sufficient amount of time.

That doesn't necessarily mean we know how to compute it in this lower bound.

E.g., sorting an array.
comparisons needed in worst case → $\Omega(n \log n)$

Will prove this soon…

Upper & Lower Bounds: Informal Summary

Upper bounds:
$\leq$ O() $<$ o()

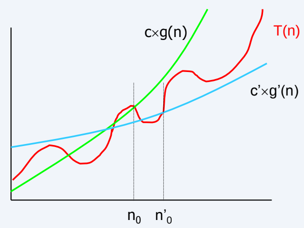
Lower bounds:
$\geq$ Ω() $>$ ω()

Upper & lower ("tight") bounds:
$=$ Θ()

Definitions: O, Ω

$T(n) \in O(g(n))$
$\leftrightarrow$
$\exists$ constants $c, n_0 > 0$
such that
$\forall n \geq n_0, T(n) \leq c \times g(n)$

$T(n) \in \Omega(g'(n))$
$\leftrightarrow$
$\exists$ constants $c', n'_0 > 0$
such that
$\forall n \geq n'_0, T(n) \geq c' \times g'(n)$



Bounding allows abstractions of details

$2n+13 \in O($? $)$	O(n)	Also, $O(n^2)$, $O(5n)$, … Can always weaken the bound.
$2n+13 \in \Omega($? $)$	Ω(n), also Ω(log n), Ω(1), …	
$2^n \in O(n)$? Ω(n) ?	Given a c, $2^n \geq c \times n$, for all but small n. Ω(n), not O(n).	
$n^{\log n} \in O(n^5)$?	No. Given a c, $\log n \geq c \times 5$, for all large enough n. Thus, $\Omega(n^5)$.	

Definitions: o, ω

Might know that our upper & lower bounds aren't tight.

$T(n) \in o(g(n))$
$\leftrightarrow$
$\forall$ constants $c>0$, $\exists$ constant $n_0>0$,
such that
$\forall n \geq n_0, T(n) < c \times g(n)$

Also, $T(n) \in o(g(n))$
$\leftrightarrow$
$$\lim_{n\to\infty} \frac{T(n)}{g(n)} = 0$$

Never equal,
for any constant.

$T(n) \in \omega(g'(n))$
$\leftrightarrow$
$\forall$ constants $c'>0$, $\exists$ constant $n'_0>0$,
such that
$\forall n \geq n'_0, T(n) > c \times g'(n)$

Examples: o, ω

$2n+13 \in o(n)$? ω(n) ?	No to both! $2n+13 < c \times n$ fails for many c. $2n+13 > c \times n$ fails for many c.
$2n+13 \in o($? $)$ ω(?)	o(n log n), $o(n^2)$, … ω(log n), ω(1), …
$1 / \log n \in o(1)$?	Yes.

$$\lim_{n\to\infty} \frac{T(n)}{g(n)} = \lim_{n\to\infty} \frac{1/\log n}{1} = \lim_{n\to\infty} \frac{1}{\log n} = 0$$

Definition: Θ

$T(n) \in \Theta(g(n))$
$\leftrightarrow$
$T(n) \in O(g(n))$ and $T(n) \in \Omega(g(n))$

Ideally, find algorithms that are asymptotically as good as possible.

Standard notational abuse

O(), Ω(), … are sets of functions.

$T(n) = O(\ldots)$ = $T(n) \in O(\ldots)$

$T(n) = f(n) + O(\ldots)$ = $T(n) = f(n) + g(n)$, $\exists g(n) \in O(\ldots)$

Three types of measures

Most common.

Worst case	T(n) = max {T(x) | x is an instance of size n}
Best case	T(n) = min {T(x) | x is an instance of size n}
Average case	$T(n) = \Sigma_{\lvert x\rvert=n} \Pr\{x\} \times T(x)$

Determining the input probability distribution can be difficult.