CS1800
Day 19

Admin:

    hw8 (function growth, sequences & series) released today (tuesday)

        - due Nov 26 (tuesday)

        - includes class 18 & 19 content

    exam2 & hw7 available by Nov 27 (likely the 25th)

        - we'll push updated grade estimates to canvas then too

Content:
- function growth
- big-o, big-theta, big-omega notation

Which gift will produce more value in one's lifetime?
- a magic penny which doubles it value every 3 years
- $10 a day

1. write first impressions (before computing) what do you think?
2. explicitly label your assumptions
3. compute & explain

Assume: 75 year lifespan
magic penny = 25 doublings = .01 * 2^25 = 335,544.32
10 dollars a day = 365 * 75 * 10= 273, 750

Which gift will produce more value over an infinite amount of time?
- a magic penny which doubles its value every 100000 years
- $100000000000000 a second

1. write first impressions (before computing) what do you think?
2. explicitly label your assumptions
3. explain (maybe don't compute ...)

"doubling" (exponential) is eventually larger than "constant" (linear) growth
- no matter how small initial value of doubling is
- no matter how large initial value of linear growth is
- no matter how often the doubling occurs
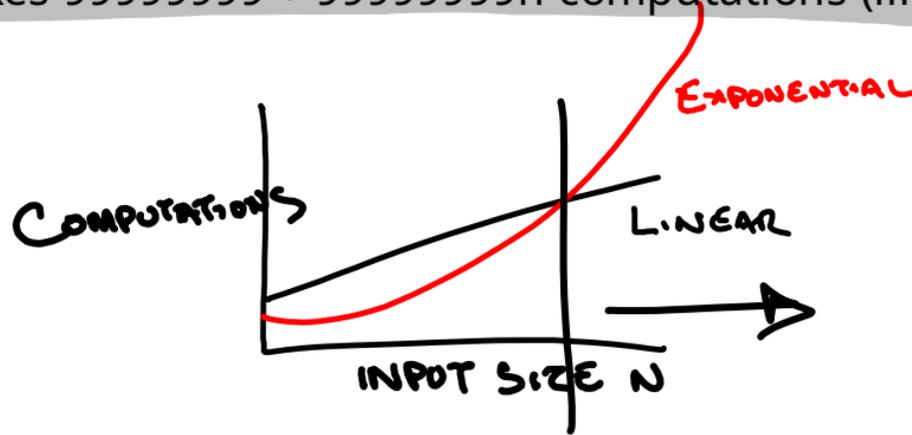- no matter how steep the linear growth occurs
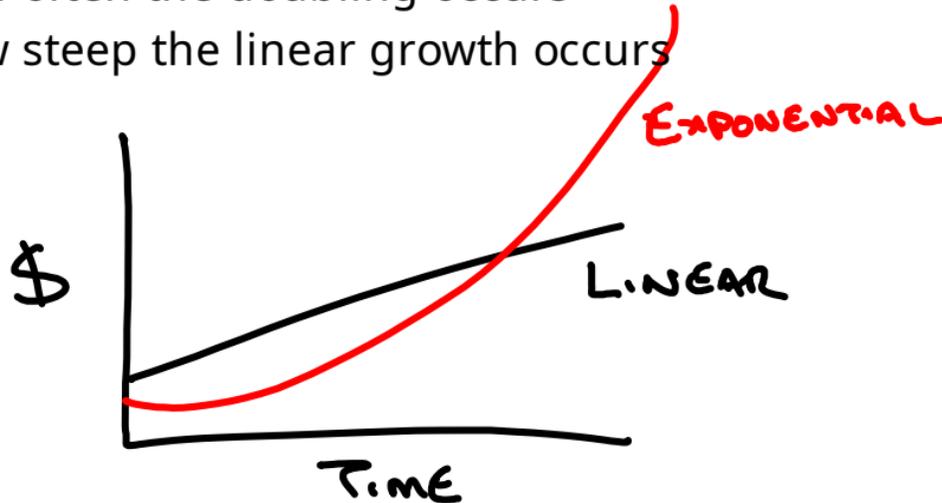
Why do we care that some functions grow faster than others?

Suppose we have two algorithms (i.e. computer programs) which accomplish the same task on an input of size n.

Algorithm 1 takes .00001 * 2^n computations (exponential)

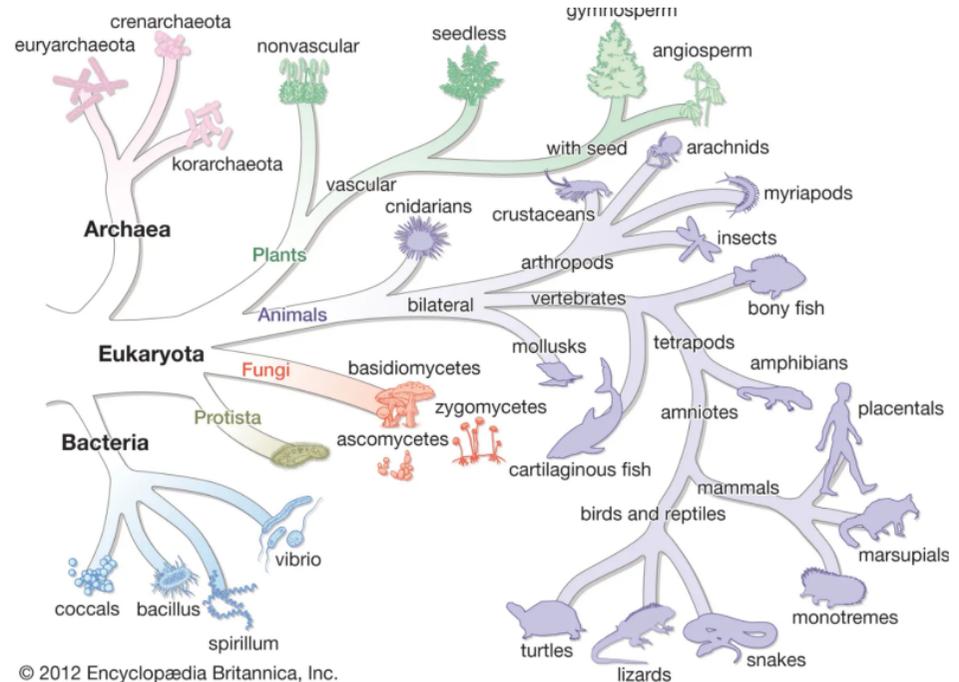Algorithm 2 takes 99999999 + 99999999n computations (linear)

Our previous example shows that for sufficiently large input size (n), algorithm 2 will take fewer computations

Objective:

Create a taxonomy of functions which allows us to organize them based on how quickly they grow.

Taxonomy (organization) of life:

crenarchaeota
euryarchaeota
korarchaeota
Archaea

nonvascular
vascular
Plants

seedless
gymnosperm
angiosperm
with seed

cnidarians
bilateral
Animals

arachnids
myriapods
insects
crustaceans
arthropods
vertebrates
bony fish
tetrapods
amphibians
amniotes
placentals

Eukaryota
Fungi
basidiomycetes
zygomycetes
ascomycetes
mollusks

Protista

Bacteria
cartilaginous fish
mammals
birds and reptiles
marsupials
monotremes

vibrio
coccals bacillus
spirillum

turtles lizards snakes

© 2012 Encyclopædia Britannica, Inc.

$$f(n) = O(g(n))$$ IS KIND OF LIKE $$"f(n) < g(n)"$$

PRONOUNCED
" BIG OH OF G OF N"

g GROWS FASTER THAN F

Big-O Notation (Intution): f(n) = O(g(n)) means g(n) grows faster than f(n)

"g(n)"

f(n)
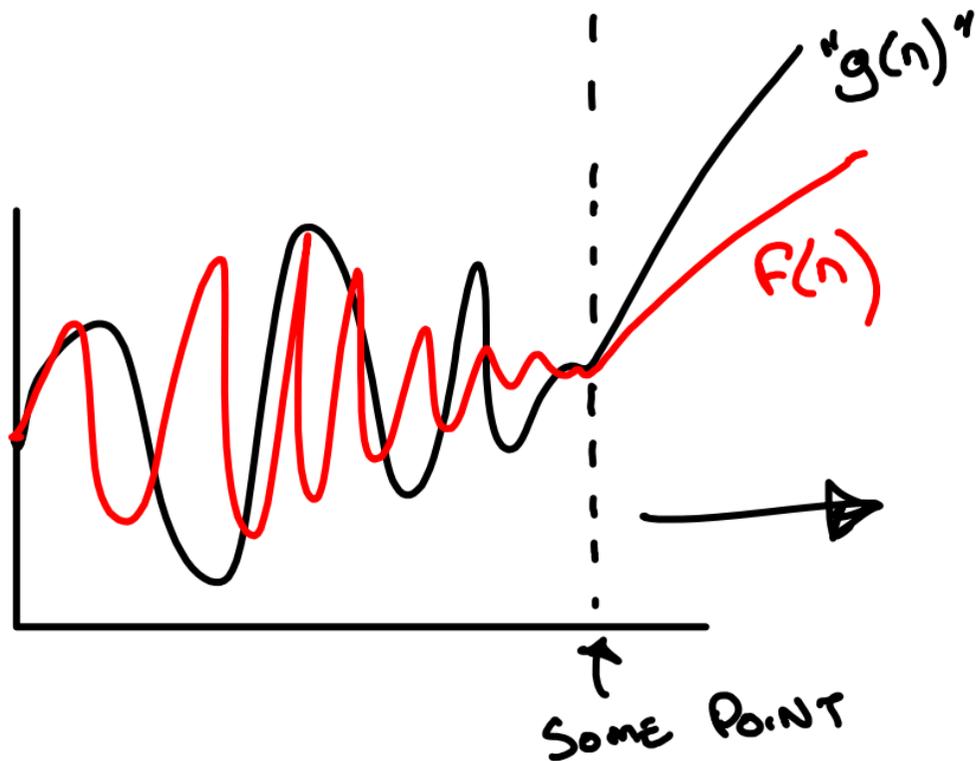
SOME POINT

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

MEANS "g(n)" IS ALWAYS LARGER THAN f(n) BEYOND SOME POINT

Big-O Notation (Intution): f(n) = O(g(n)) means g(n) grows faster than f(n)
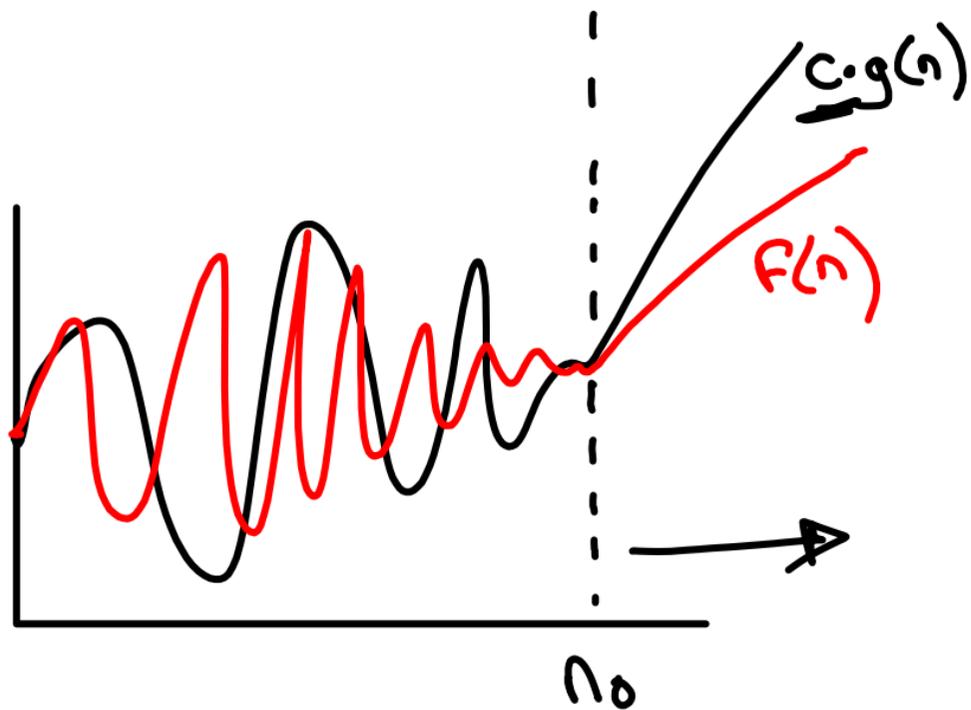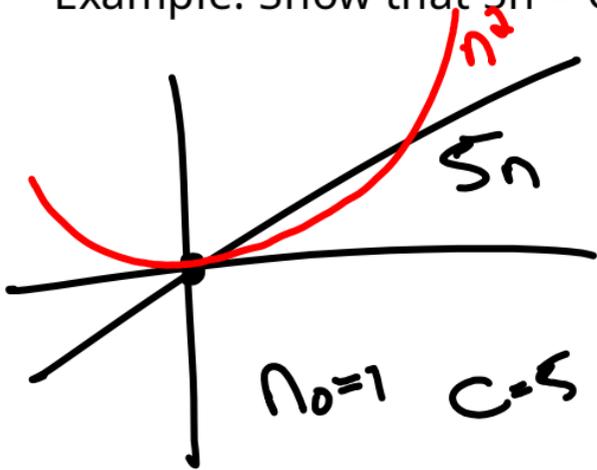

$$f(n) = O(g(n))$$

MEANS
THERE EXISTS VALUES
n_0 AND c WITH

$$n_0 \leq n \longrightarrow 0 \leq f(n) \leq c \cdot g(n)$$

How do we show $f(n) = O(g(n))$? Choose n_0 and c to satisfy the definition

Example: Show that 5n = O(n^2)

n^2

$5n$

$n_0 = 1 \quad c = 5$

$1 \le n \longrightarrow 0 \le 5n \le 5n^2$

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

MEANS
THERE EXISTS VALUES
n_0 AND c WITH

$n_0 \le n \longrightarrow 0 \le f(n) \le c \cdot g(n)$

Proving Big-O notation: FAQ

Aren't there many choices for n_0 and c?

There are!

So why do you choose these particular ones?

Remember, our purpose in writing a proof is to be compelling.
For this reason, choose the n_0 and c which are as simple as possible.

How will I know if my values are the simplest? Will credit be taken if I don't get the absolute simplest values?

There are many n_0, c pairs which are equally compelling. Avoid blindly choosing really large values (even if they "work" they're hard to understand)

Prove each true statement below. If a statement is false, give a justification of why it is false (sketching a graph is often a good idea here).
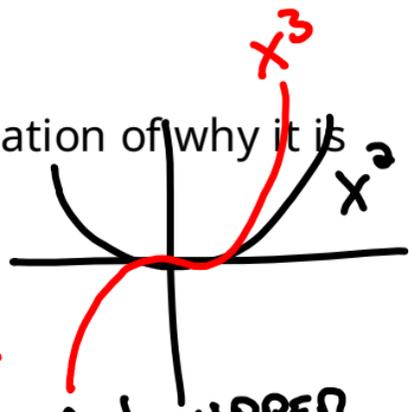
$20 x = O(x)$

$1 \leq x \rightarrow 0 \leq 20x \leq 20x$

$x = O(20x)$

$0 \leq x \rightarrow 0 \leq x \leq 1 \cdot 20x$

$x^3 = O(x^2)$

x^2 IS NOT AN UPPER BOUND FOR x^3

$x^2 = O(x^3)$

$1 \leq x \rightarrow 0 \leq x^2 \leq 1 \cdot x^3$

$$f(x) = O(g(x)) \longleftrightarrow \text{"} f(x) \leq g(x) \text{"}$$

A Final Intuition of Big-O

$f(n) = O(g(n))$ MEANS $g(n)$ GROWS AT LEAST AS QUICKLY AS $f(n)$

$"f(n) \leq g(n)"$

x^3

x^2

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

MEANS
THERE EXISTS VALUES
n_0 AND c WITH

$$n_0 \leq n \longrightarrow 0 \leq f(n) \leq c \cdot g(n)$$

Compute Time

Size of input

n_0

$c \cdot g(n)$

$f(n)$

In our context (n=input size, f(n) = compute time) we don't care about small n, they're easily computed anyways!
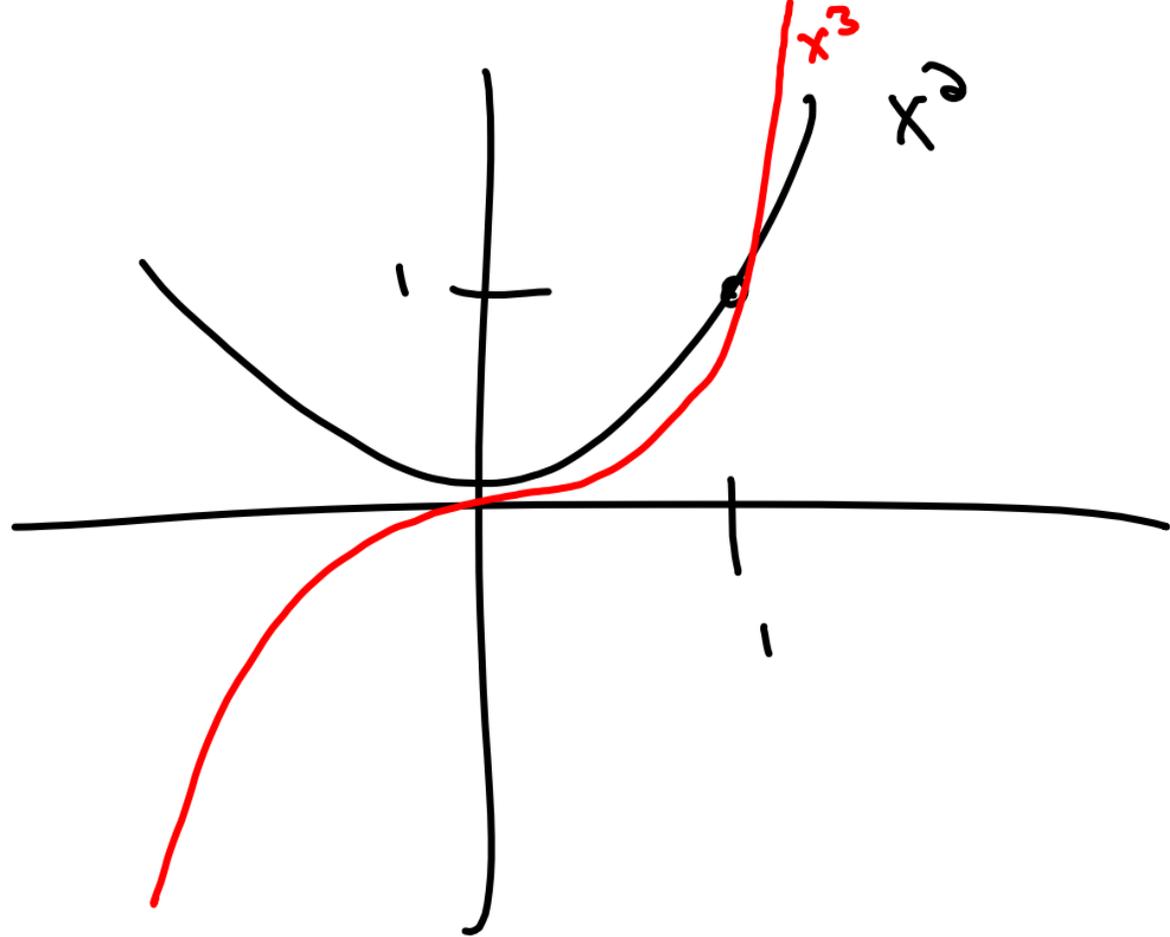
Critiquing the Big-O definition: why allow a multiplicative constant c?

FROM IN CLASS ACTIVITY

$$20_x = O(x) \quad \text{AND} \quad x = O(20x)$$

X AT LEAST
AS FAST AS 20x

20x AT LEAST
AS FAST AS X

SO X AND 20x
GROW EQUALLY QUICKLY

Inclusion of c allows a notion of functions which grow equally quickly.
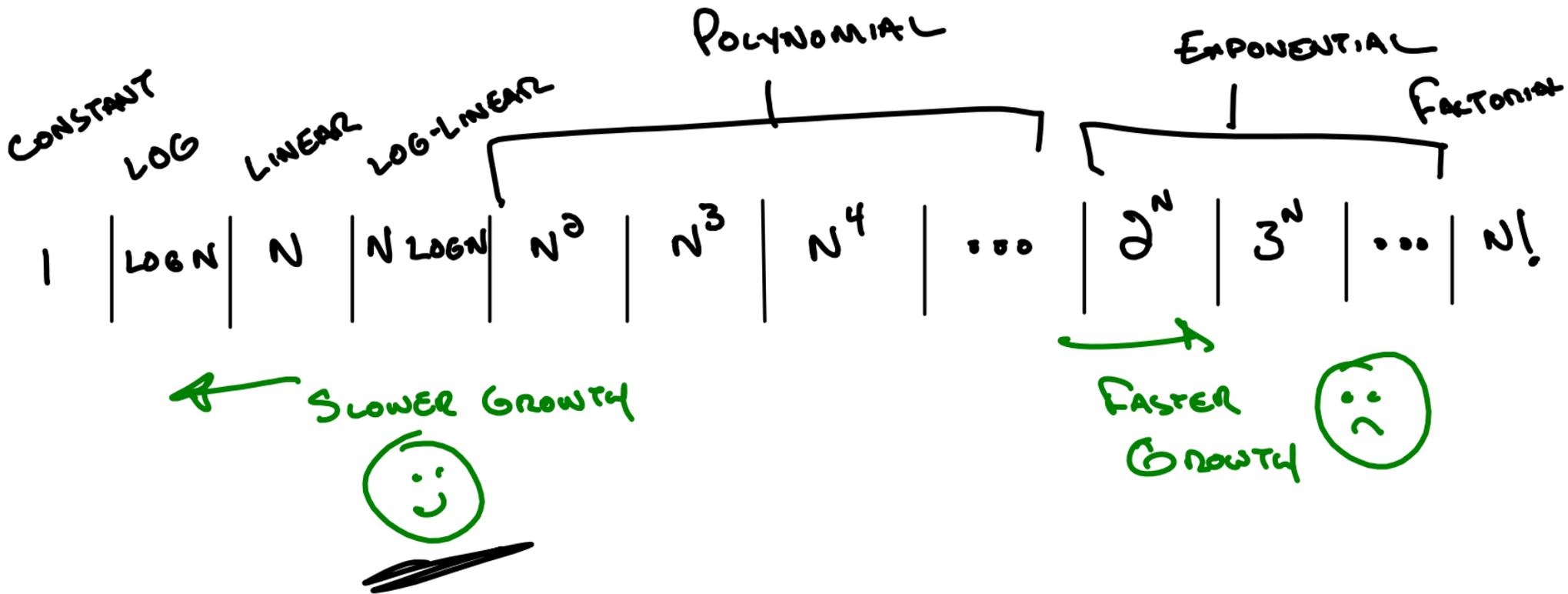
Useful insight 1:
Ignore constant multipliers in a function when considering Big-O

Motivation:
Simplifies how we define function growth (there are many functions in the same "growth bucket", all grow equally quickly)

Function Growth Buckets:

POLYNOMIAL

EXPONENTIAL

CONSTANT

LOG

LINEAR

LOG-LINEAR

FACTORIAL

| 1 | $\log N$ | N | $N \log N$ | N^2 | N^3 | N^4 | $\cdots$ | 2^N | 3^N | $\cdots$ | $N!$ |

← SLOWER GROWTH

FASTER GROWTH →

INPUT SIZE

	n			
	10	50	100	1,000
$\lg n$	0.0003 sec	0.0006 sec	0.0007 sec	0.0010 sec
$n^{1/2}$	0.0003 sec	0.0007 sec	0.0010 sec	0.0032 sec
n	0.0010 sec	0.0050 sec	0.0100 sec	0.1000 sec
$n \lg n$	0.0033 sec	0.0282 sec	0.0664 sec	0.9966 sec
n^2	0.0100 sec	0.2500 sec	1.0000 sec	100.00 sec
n^3	0.1000 sec	12.500 sec	100.00 sec	1.1574 day
n^4	1.0000 sec	10.427 min	2.7778 hrs	3.1710 yrs
n^6	1.6667 min	18.102 day	3.1710 yrs	3171.0 cen
2^n	0.1024 sec	35.702 cen	4×10^{16} cen	1×10^{166} cen
$n!$	362.88 sec	1×10^{51} cen	3×10^{144} cen	1×10^{2554} cen

10 DIFFERENT ALGORITHM

Table 14.1: Time required to process n items at a speed of 10,000 operations/sec using ten different algorithms. *Note:* The units above are seconds (sec), minutes (min), hours (hrs), days (day), years (yrs), and centuries (cen)!

$a(n) + b(n)$

$a(n)$

$b(n)$

n
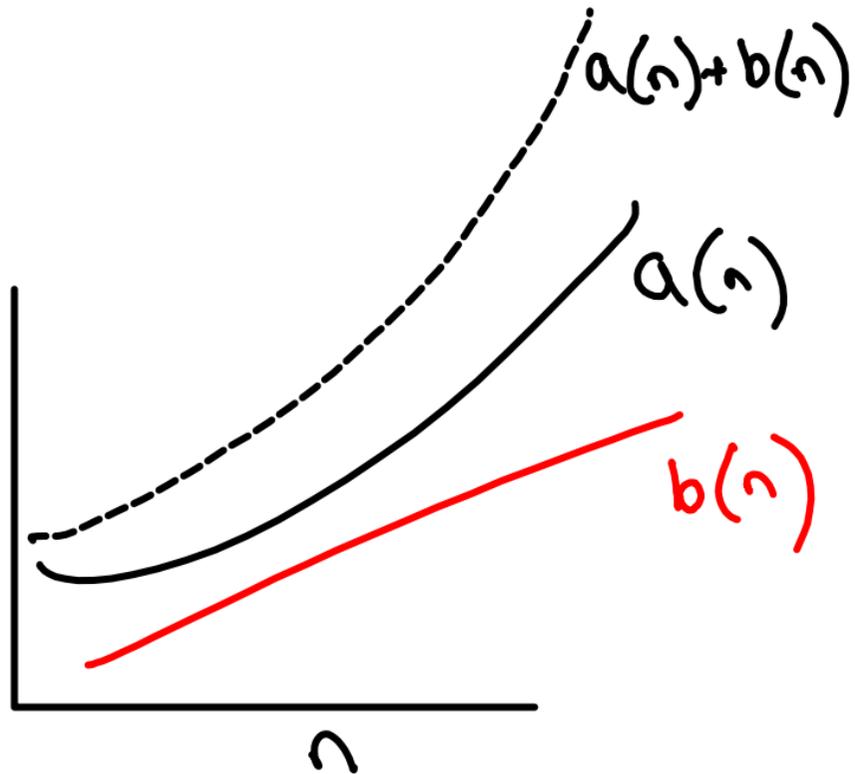
IF a GROWS FASTER THAN b THEN a+b GROWS AS QUICKLY AS a

$=$

IF $b(n) = O(a(n))$

THEN $a(n) + b(n) = O(a(n))$

$$1 \mid LOG\,N \mid N \mid N\,LOG\,N \mid N^2 \mid N^3 \mid N^4 \mid \cdots \mid 2^N \mid 3^N \mid \cdots \mid N!$$
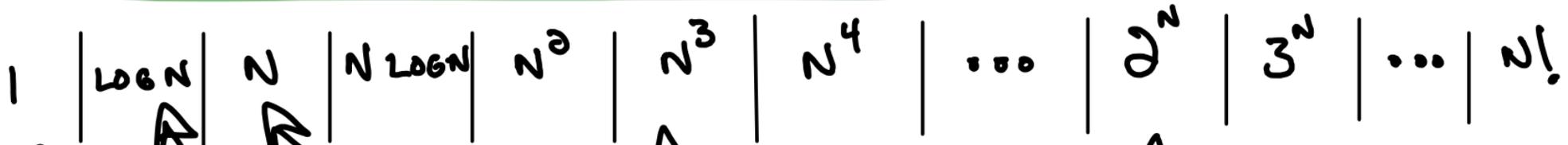
Insight1: ignore constant multipliers

Insight2: discard slower growing terms

$$1 + LOG_{10}\,N + 14\,N + \tfrac{1}{7}\,N^3 + .0001\,2^N = O(2^N)$$

POLYNOMIAL   EXPONENTIAL

1	LOG N	N	N LOG N	N^2	N^3	N^4	...	2^N	3^N	...	N!

Give the simplest, slowest growing function g(n) such that each f(n) = O(g(n))
(see previous slide)

$$f_1(n) = 2n + 3n^2 = O(n^2)$$

$$f_2(n) = 1234 + N \ LOG \ N + 7 + 4 + 3 + n^{1000} + 1.01^n$$
$$= O\left(1.01^N\right)$$

ICE CREAM COST $= .01 N + N^2$

$N = 1$ $\qquad$ $.01 + 1^2 = 1.01$

$N = 100$ $\qquad$ $.01 \cdot 100 + 100^2 = 10001$

Big-Omega is the opposite of Big-O:

$f \geq g$

BIG O

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

$f \leq g$

THERE EXISTS VALUES
n_0 AND c WITH

$$n_0 \leq n \longrightarrow 0 \leq f(n) \leq c \cdot g(n)$$

$g(n)$ IS UPPER BOUND ON $f(n)$

BIG OMEGA

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

THERE EXISTS VALUES
n_0 AND c WITH

$$n_0 \leq n \longrightarrow 0 \leq c \cdot g(n) \leq f(n)$$
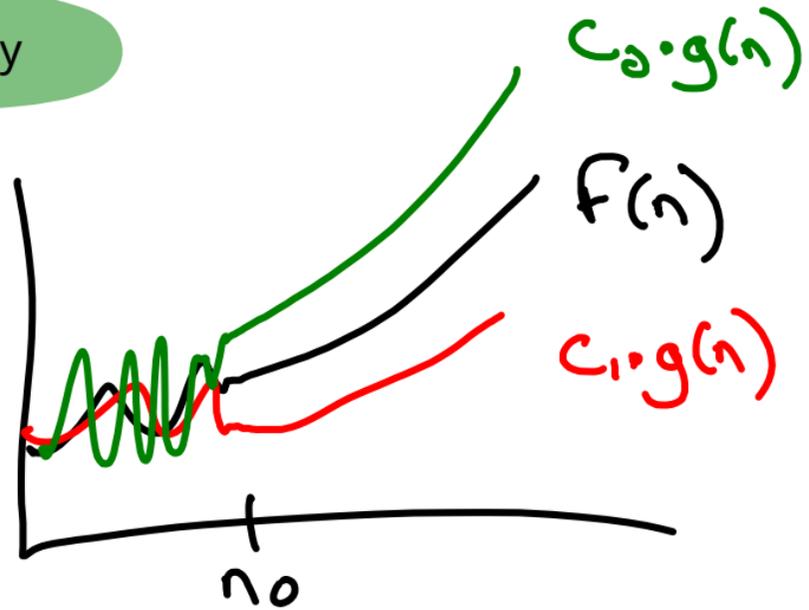
$g(n)$ IS LOWER BOUND ON $f(n)$

$$x = O(x^2)$$

$$x^2 = \Omega(x)$$

BIG THETA

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

THERE EXISTS c_1, c_2, n_0 WITH

$$n_0 \le n \to 0 \le c_1 g(n) \le f(n) \le c_2 g(n)$$

$c_2 \cdot g(n)$

$f(n)$

$c_1 \cdot g(n)$

n_0

$$10N = \Theta(100N)$$

BIG-O AND BIG OMEGA = BIG THETA

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

AND

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

$\longleftrightarrow$

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

Tell whether each of the following statements are true or false

$N = O(N^3)$ "N^3 grows faster / (as fast as) N": True

$N^2 = \Omega(N)$ is the same as $N = O(N^2)$ "N^2 grows faster / as-fast-as N": True

$10N + \log N = O(.1\,N)$ is the same as "$N = O(N)$"   "N grows faster / as fast as N": True

$14 + N \log_2 N = \Theta(15\,N \log_2 N)$ is the same as "$N \log_2 N = \Theta(N \log_2 N)$ " means "$N \log_2 N$ in same growth bucket as $N \log_2 N$": True
$\log_2 N = \Theta(\log_{10} N)$ (hint: $\log_b(x) = \log_a(x) / \log_a(b)$, to change the base of a log we need only multiply by constant)
$\log_2 N = constant * \log_{10} N$, is True

1 | log N | N |NLogN | N^2 | N^3 | ... |2^N | 3^N | ... | N!

$f(n) = O(g(n))$ means   "$f(n) \le g(n)$":   g grows as fast as f (or faster)
$f(n) = \Omega(g(n))$ means   "$f(n) \ge g(n)$":   g grows as slowly as f (or slower)
$f(n) = \Theta(g(n))$ means   "$f(n) = g(n)$":   g grows as quickly as f (in same bucket)

inequalities are analagous and help build intuition,
they're not quite true in a literal sense though (see definitions)