Please download latest notes off of Piazza

Professor Hamlin
Day 19

1) Admin
2) Review
3) Function growth
4) Big-O, Big-Θ, Big-Ω

Review

Summary of Arithmetic, Geometric & Quadratic
(k=0)

	Arithmetic	Geometric	Quadratic
How to identify	2 4 6 8 10 … +2 +2 +2 +2 Difference constant	1 2 4 8 16 … ×2 ×2 ×2 ×2 Constant ratio	1 3 7 13 21 2 4 6 8 +2 +2 +2 constant second difference
Expression of Single term	$a_0 + dk$	$a_0 \cdot r^k$	$ak^2 + bk + c$
Computing Partial Sum	$\sum_{k=0}^{N} a_0 + dk = (a_0 + a_N)\left(\frac{N+1}{2}\right)$	$\sum_{k=0}^{N} a_0 r^k = \frac{a_0(1-r^{N+1})}{1-r}$	Calculus fun (not CS1800)

Exercise: 1) $\sum_{k=0}^{4} 3 + dk = 35$ - What is d?

hint: expand it out, then solve for d

k=0 to 4 (5 terms)

$(3+0d) + (3+1d) + 3+2d + 3+3d + 3+4d = 35$
k=0

$15 + 10d = 35$

$d = 2$

Activity 1 Which gift would be worth more over 75 years?

1) a magic penny whose value doubles every 3 years
2) $10 a day

a) Write 1st impression before math
b) Compute and compare.

a) exponential penny

b) $10 a day

```
10*(75*365)
= 273750
```

exponential penny

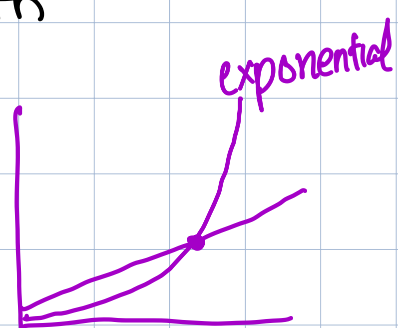
```
(2^(75/3))/100.
= 335544.32
```

Activity 2 Which would produce more value over the life of the universe?

1) A magic penny whose value doubles every 100,000 yrs
2) $1,000,000,000,000 a second

a) what are your 1st impression pre-math
b) explain (maybe don't compute)

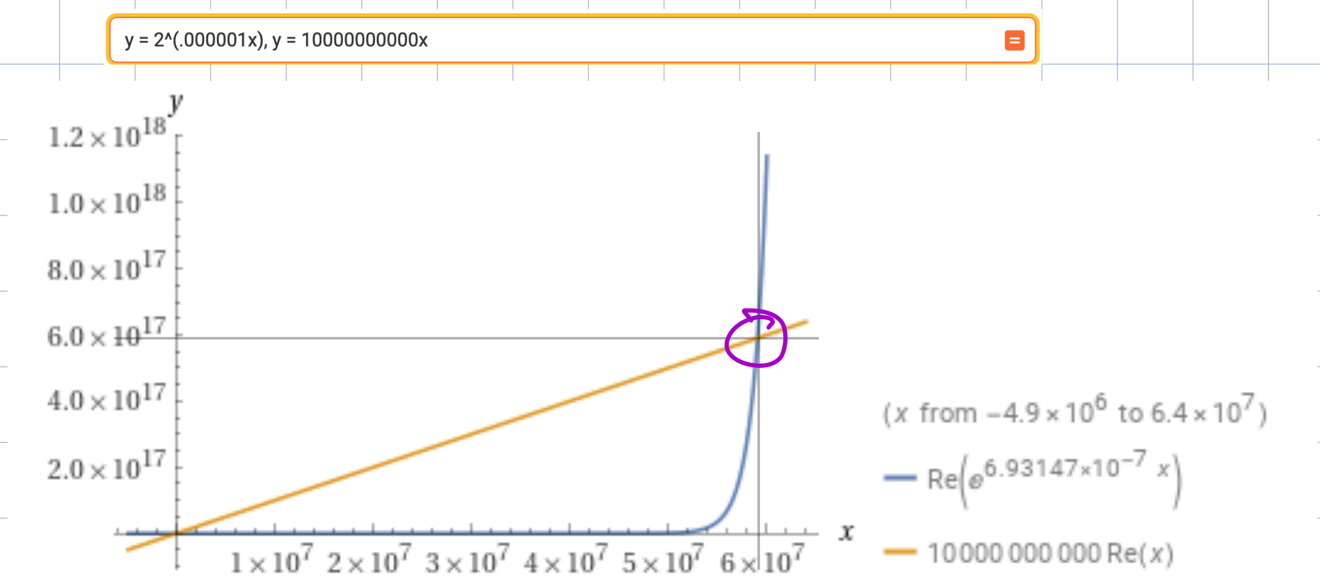
exponential penny



Takeaway: Some functions grow faster than others

An exponential function will become larger than linear growth no matter:

- how small the initial value to double is
- how large the initial value for linear growth
- how often the doubling occurs
- how steep the linear growth is



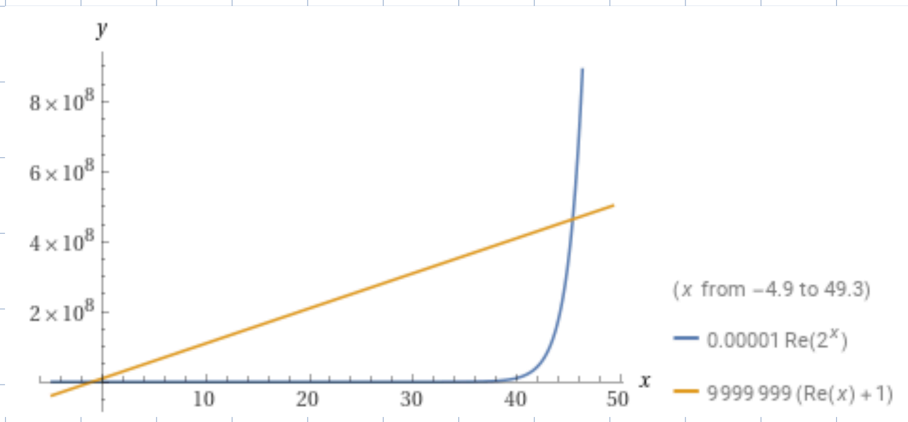
Why do we care about how fast a function grows?

Consider two different computer programs that accomplish the same thing. However, on input size n

Algorithm 1: takes $.00001 \cdot 2^n$ steps

Algorithm 2: takes $9999999 + 9999999n$

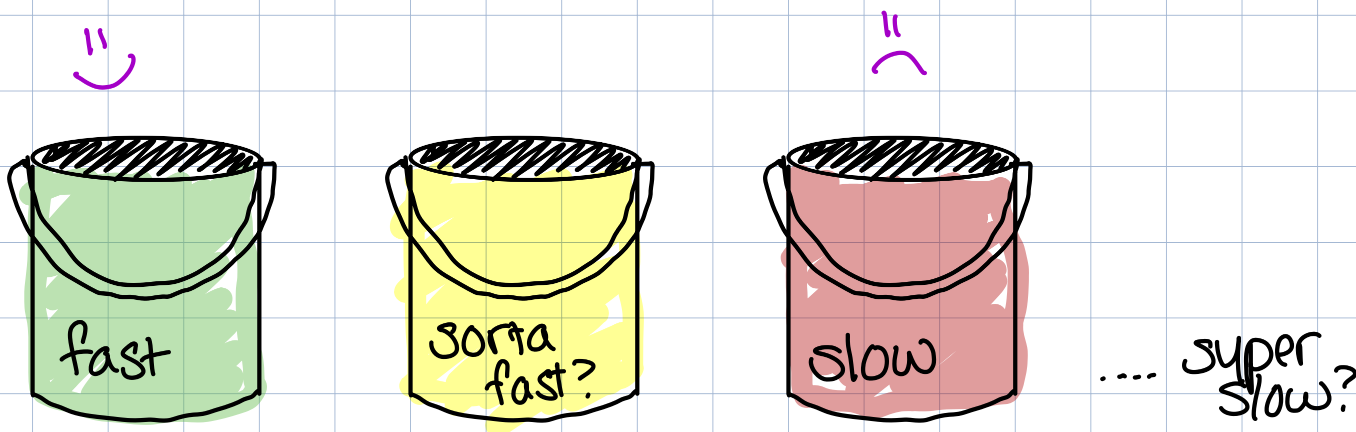
We know Algorithm 2 will at some point take fewer steps.



$x \approx 45.3992, \quad y \approx 4.63992 \times 10^8$

In fact at $n = 46$, Algorithm 2 is slower....

Comparing how fast algorithms run



on input of length n; how would we classify these algorithms?

3		
n	$n + 100000000000$	n^7
n^2	$n^2 + \log n$	3^n
2^n	$n^{10000000}$	$n \log n$

We need a better way of putting our run-times in "buckets" so we can compare them

aka. a way to say a function $(f(n))$ is...
...."bigger than"...
...."smaller than"...
...."the same as"....
... another function $g(n)$

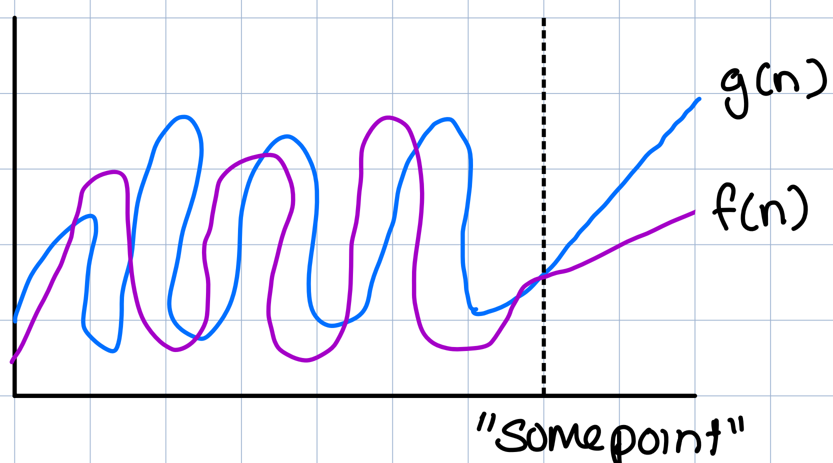
Big-O Notation

$f(n) = O(g(n))$ is similar to "$f(n) \le g(n)$"

"Big-O of g of n"

g grows faster than f

It means that at a certain point $g(n)$ will be larger than $f(n)$ past some point



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

$\forall n > n_0 \longrightarrow f(n) \le g(n)$

Formally, $\exists\ n_0, c \in \mathbb{N}$ s.t. $\forall n \geq n_0$;

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

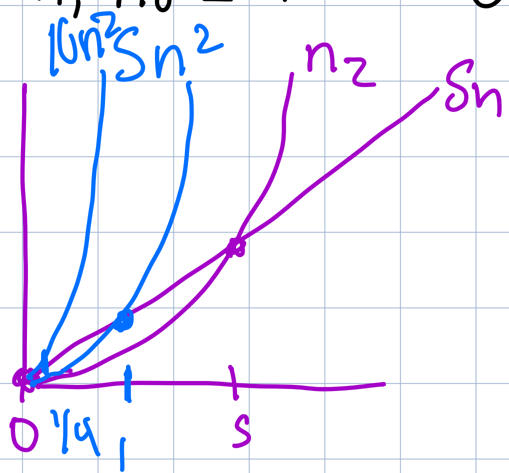
"There exists two natural numbers n_0 & c s.t. for any value of n greater than n_0, if you evaluate $f(n)$ its result will be smaller than $c \cdot g(n)$"

Cool so we have our definition how do we use it?

e.g. $\quad 5n = O(n^2)$

we need to find values of c & n_0 s.t.

$\forall n,\ n_0 \leq n \qquad 0 \leq 5n \leq \underline{c} \cdot n^2$



what is n_0? 5

$c = 1$, $\forall n > n_0$

$5n \leq n^2$

$$\boxed{c = 1 \quad n_0 = 5}$$

what is n_0? 1

$c = 5 \quad \forall n > n_0$

$5n \leq 5n^2$

$$\boxed{c = 5,\ n_0 = 1}$$

chose n_0 given c when $c \cdot g(n) > f(n)$

$$\boxed{c = 5 \quad n_0 = 10}$$

Big-O FAQs

1) Aren't there infinitely many choices for c, n_0?

Yep!

2) So why choose the ones we did?

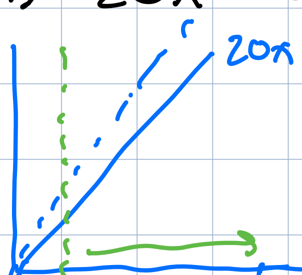
We want our proof to convince other people than us. So $c = 100000000$ and $n_0 = 100000$ is hard to think about

3) How do I know what value to use? Will I lose points if I don't choose correctly?

There are many values of c/n_0 that work → try to choose values for c/n_0 that are < 20.

Exercise For true statements state c & n_0
For false statements, give justification (possibly a graph)

1) $20x = O(x)$



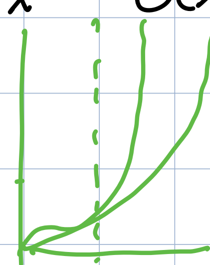
$x_0 = 2$

$\forall x > 2$

$0 \leq 20x \leq cx$

$\boxed{x_0 / n_0 = 2 \quad c = 22}$

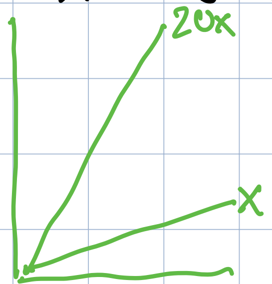
2) $x^3 = O(x^2)$



$x^3 < c \cdot x^2$

False, no c is going to work

3) $x = O(20x)$ True



$x_0 = 3$

$\forall x \geq 3$

$0 \leq x \leq c \cdot 20x$

$\boxed{x_0 / n_0 = 3 \quad c = 1}$

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

$n_0 = 1 \qquad c = 2$

$\forall x \geq 1$

$0 \leq x^2 \leq 2x^3$

$x = 1 \qquad 1^2 \leq 2 \cdot 1^3$

$f(n) = O(g(n))$: $g(n)$ grows atleast as quickly as $f(n)$

Why do we only care about really big values of n?

n = input size, f(n) is compute time.
This type of abstraction isn't really needed for small values of n, which can easily be computed.

What is up w/ this whole "c" thing?

Recall:

$20x = O(x)$ x is atleast as fast as 20x	$x = O(20x)$ 20x is atleast as fast as x

→ it seems x & 20x belong in the same "bucket" of linear growth

→ c captures idea of functions being mutually Big-O to each other

Useful insight 1: ignore constant multipliers in a function when considering Big-O

$f_1(n) = 2n \quad O(n)$ $\qquad f_2(n) = 1000000n \quad O(n)$

Common "Buckets"

log $\quad$ log-linear $\quad$ exponential

$1 \mid \log N \mid N \mid N\log N \mid N^2 \mid N^3 \mid \dots \mid 2^N \mid 3^N \mid \dots \mid N! \mid N^N$

constant $\quad$ linear $\quad$ polynomial $\quad$ factorial

slower growth :) $\qquad$ faster growth :(

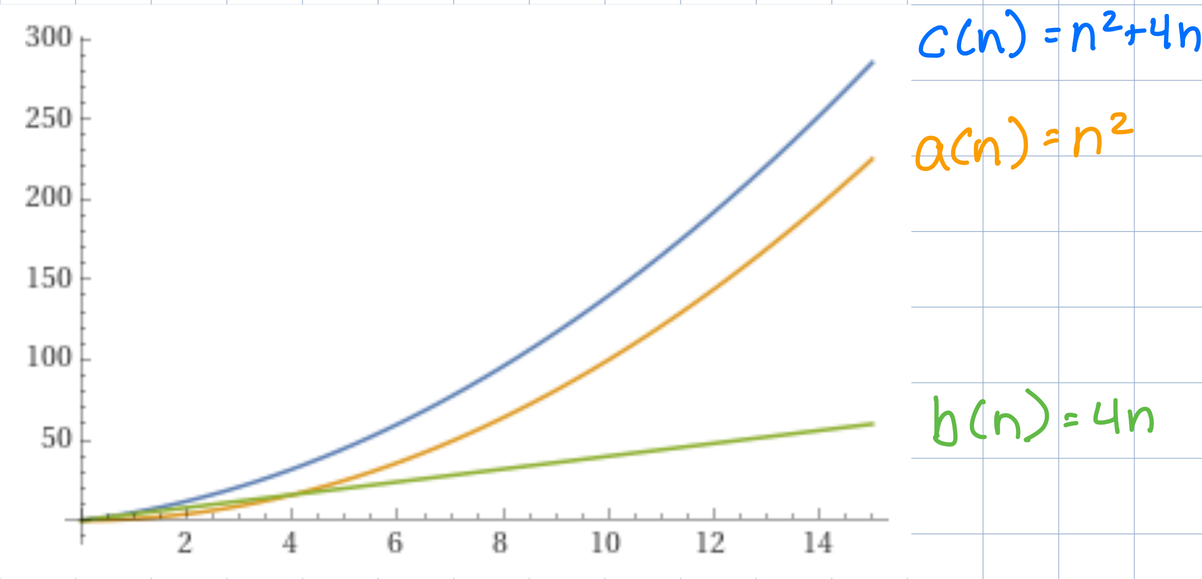
A concrete comparison:

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

run times

Only the biggest function of n matters

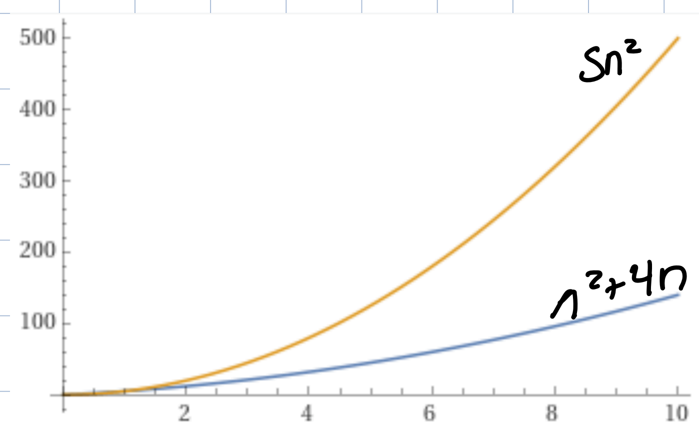


informal: if $a(n)$ grows faster than $b(n)$ then
$c(n) = a(n) + b(n)$ grows as quickly as $a(n)$

Formally: if $b(n) = O(a(n))$ then
$c(n) = O(a(n))$

Formally showing it w/ n_0 & c: $c(n) = O(n^2)$
$c(n) = n^2 + 4n$ $\quad n_0 = 1 \quad c = 5$

$n=1 \quad 1^2 + 4\cdot 1 \leq 5 \cdot 1^2$ ✓
$n=2 \quad 2^2 + 4\cdot 2 \leq 5 \cdot 4$ ✓
$n=3 \quad 3^2 + 4\cdot 3 < 5 \cdot 9$ ✓



Quickly assessing function growth

log, log-linear, exponential

1	$\log N$	N	$N\log N$	N^2	N^3	....	2^N	3^N	...	$N!$	N^N

constant, linear, polynomial, factorial

Insight 1: ignore constants
Insight 2: discard slower growing terms

$$f(n) = 1 + \log_{10} n + 14n + \pi n^2 + .00001 \cdot 2^N$$

$$\boxed{O(2^N)}$$

log, log-linear, exponential

1	$\log N$	N	$N\log N$	N^2	N^3	....	2^N	3^N	...	$N!$	N^N

constant, linear, polynomial, factorial

Exercise) Find simplest Big-O

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

$O(n^2)$

2) $f_2(n) = \cancel{1234} + \cancel{2n\log n} + \cancel{7} + \cancel{4} + \cancel{3} + \cancel{n^{100}} + 1.01^n$

$O(1.01^n)$

So far we wanted...

aka. a way to say a function ($f(n)$) is...

...."bigger than"....

...."smaller than"... Big-O

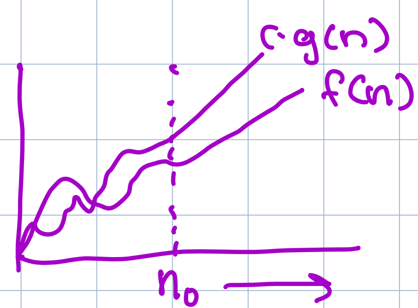
...."the same as"....

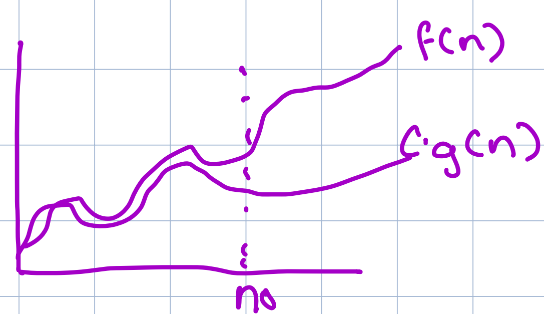
... another function $g(n)$

Now how we capture $f(n)$ is "bigger than" $g(n)$

$\boxed{\text{Big-Omega}}$ Ω

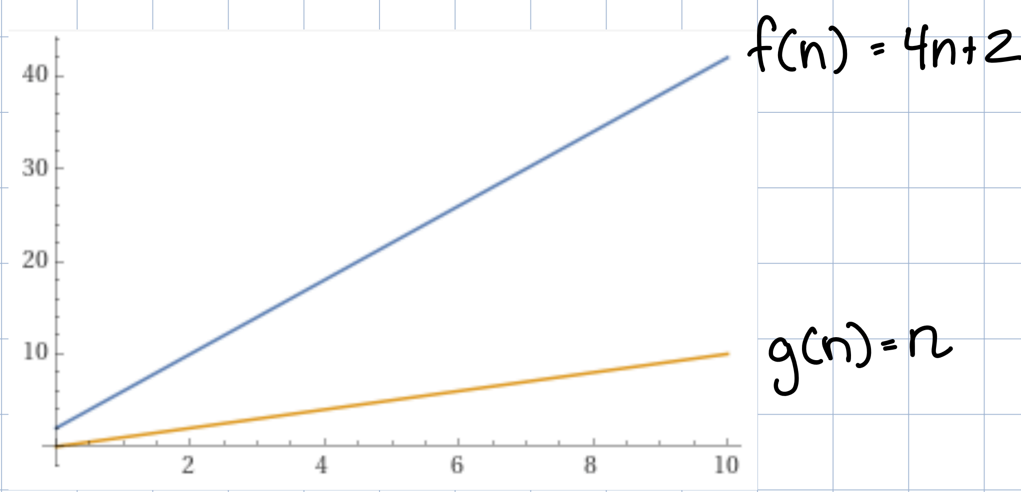
Big-O	Big-Omega
$f(n) = O(g(n))$	$f(n) = \Omega(g(n))$
$\exists\ n_0, c \in \mathbb{N}$ s.t. $\forall n \geq n_0 \rightarrow 0 \leq f(n) \leq c \cdot g(n)$	$\exists\ n_0, c \in \mathbb{N}$ s.t. $\forall n \geq n_0 \rightarrow 0 \leq c \cdot g(n) \leq f(n)$
$g(n)$ is an "upper bound" for $f(n)$	$g(n)$ is a "lower bound" for $f(n)$





Example! $f(n) = 4n+2$ $\qquad f(n) = \Omega(n)$

$n_0 = 1 \qquad c = 1$



So far we wanted...

> aka. a way to say a function $(f(n))$ is...
> ...."bigger than".... Big-Omega
> ...."smaller than"... Big-O
> ...."the same as".... Big-Theta
> ... another function $g(n)$

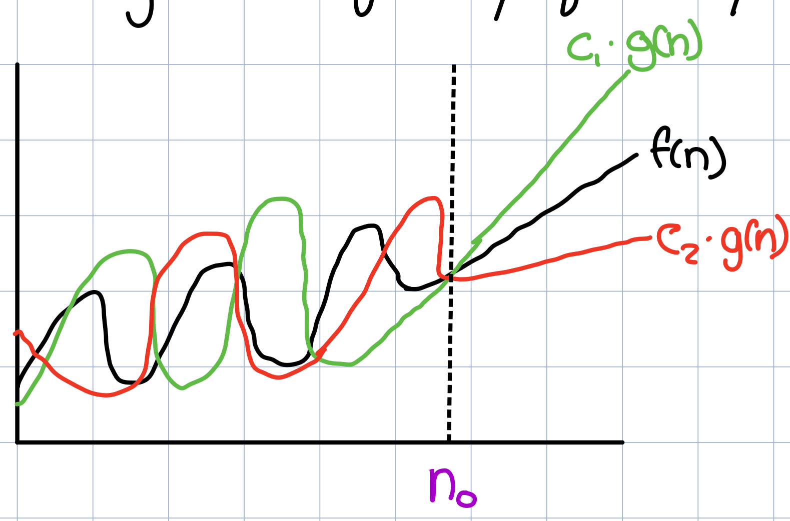
Big-Theta: two functions grow equally quickly

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

$\exists\ c_1, c_2, n_0 \in \mathbb{N}$ s.t

$\forall n > n_0$:

$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$



Big-O and Big-Omega = Big-Theta

$$f(n) = O(g(n)) \text{ AND } f(n) = \Omega(g(n))$$

$$\Updownarrow$$

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

e.g. $x \leq 1$
$x \geq 1$
$x = 1$

Example $f(n) = 5n^2 + 3n + 5$

$$f(n) = O(n^2) \qquad f(n) = \Omega(n^2)$$

then

$$f(n) = \Theta(n^2)$$

Example Are the following statements true or false

1) $n = O(n^3)$ T

2) $n^2 = \Omega(n)$ T

3) $10n + \log n = O(.1n)$ T
n $O(n)$

4) $14 + n\log_2 n = \Theta(n \log_2 n)$ T
$n\log_2 n$

5) $\log_2 n = \Theta(\log_{10} n)$
$k \cdot \log_2 n$

hint: $\log_b x = \dfrac{\log_a x}{\log_a b}$

T