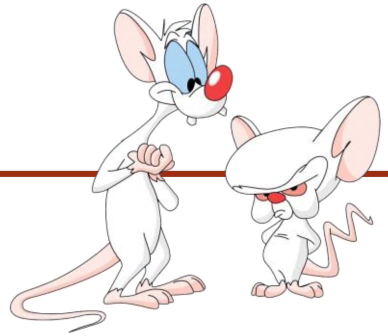
COMP 250

INTRODUCTION TO COMPUTER SCIENCE

Week 6-3 : Asymptotic Notation 2

Giulia Alberini, Fall 2020

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Properties of Asymptotic notations
- Big-Omega, $\Omega(\cdot)$
- Big-Theta, $\Theta(\cdot)$

RULES OF BIG-OH

- Scaling
- Sum rule
- Product Rule
- Transitivity

SCALING

For all constant factors $a > 0$,

if $f(n)$ is $O\big(g(n)\big)$, then $a \cdot f(n)$ is also $O\big(g(n)\big)$.

(This rule is obvious if you understand the definition of big O)

SCALING

For all constant factors $a > 0$,

if $f(n)$ is $O(g(n))$, then $a \cdot f(n)$ is also $O(g(n))$.

Proof: By definition, if $f(n)$ is $O(g(n))$ then there exist two positive constants n_0 and c such that, for all $n \geq n_0$,

$$f(n) \leq c\ g(n).$$

Thus, …?

SCALING

For all constant factors $a > 0$,

if $f(n)$ is $O(g(n))$, then $a \cdot f(n)$ is also $O(g(n))$.

Proof: By definition, if $f(n)$ is $O(g(n))$ then there exist two positive constants n_0 and c such that, for all $n \geq n_0$,

$$f(n) \leq c\, g(n).$$

Thus,

$$a \cdot f(n) \leq \underbrace{a\,c}\, g(n)$$

This constant satisfies the definition that $a \cdot f(n)$ is $O(g(n))$

SUM RULE

If $f_1(n)$ is $O\big(g(n)\big)$ and $f_2(n)$ is $O\big(g(n)\big)$, then $f_1(n) + f_2(n)$ is $O\big(g(n)\big)$.

Proof: …

SUM RULE

If $f_1(n)$ is $O\big(g(n)\big)$ and $f_2(n)$ is $O\big(g(n)\big)$, then $f_1(n) + f_2(n)$ is $O\big(g(n)\big)$.

Proof: Let n_1, c_1 and n_2, c_2 be constants such that

$f_1(n) \le c_1\, g(n)$ for all $n \ge n_1$ *and* $f_2(n) \le c_2\, g(n)$ for all $n \ge n_2$

SUM RULE

If $f_1(n)$ is $O(g(n))$ and $f_2(n)$ is $O(g(n))$, then $f_1(n) + f_2(n)$ is $O(g(n))$.

Proof: Let n_1, c_1 and n_2, c_2 be constants such that

$f_1(n) \leq c_1\, g(n)$ for all $n \geq n_1$ *and* $f_2(n) \leq c_2\, g(n)$ for all $n \geq n_2$

Then,

$$f_1(n) + f_2(n) \leq (c_1 + c_2)\, g(n) \quad \text{for all } n \geq \max(n_1, n_2)$$

These constants satisfy the big O definition

SUM RULE (MORE GENERAL)

If $f_1(n)$ is $O\big(g_1(n)\big)$ and $f_2(n)$ is $O\big(g_2(n)\big)$,

Then $f_1(n) + f_2(n)$ is $O(g_1(n) + g_2(n))$.

Proof: Try it!

PRODUCT RULE

If $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$, then $f_1(n) * f_2(n)$ is $O(g_1(n) * g_2(n))$.

Proof: …

PRODUCT RULE

If $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$, then $f_1(n) * f_2(n)$ is $O(g_1(n) * g_2(n))$.

Proof: Let n_1, c_1 and n_2, c_2 be constants such that

$f_1(n) \leq c_1\, g_1(n)$ for all $n \geq n_1$ *and* $f_2(n) \leq c_2\, g_2(n)$ for all $n \geq n_2$

PRODUCT RULE

If $f_1(n)$ is $O\big(g_1(n)\big)$ and $f_2(n)$ is $O\big(g_2(n)\big)$, then $f_1(n) * f_2(n)$ is $O\big(g_1(n) * g_2(n)\big)$.

Proof: Let n_1, c_1 and n_2, c_2 be constants such that

$f_1(n) \le c_1\, g_1(n)$ for all $n \ge n_1$ *and* $f_2(n) \le c_2\, g_2(n)$ for all $n \ge n_2$

Then,

$$f_1(n) * f_2(n) \le \underbrace{c_1\, c_2}\, g_1(n) * g_2(n) \text{ for all } n \ge \underbrace{\max(n_1, n_2)}$$

These constants satisfy the big O definition

TRANSITIVITY RULE

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then… ?

TRANSITIVITY RULE

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.

TRANSITIVITY RULE

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.

Proof: Let n_1, c_1 and n_2, c_2 be constants such that

$f(n) \leq c_1 \; g(n)$ for all $n \geq n_1$ *and* $g(n) \leq c_2 \, h(n)$ for all $n \geq n_2$

TRANSITIVITY RULE

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.

Proof: Let n_1, c_1 and n_2, c_2 be constants such that

$$f(n) \le c_1 \, g(n) \text{ for all } n \ge n_1 \text{ and } g(n) \le c_2 \, h(n) \text{ for all } n \ge n_2$$

Then,

$$f(n) \le \underbrace{c_1 \, c_2}\, h(n) \text{ for all } n \ge \underbrace{\max(n_1, n_2)}$$

These constants satisfy the big O definition

COMMON FUNCTIONS

Claim: each of the following holds for n sufficiently large

$$\underbrace{1 < \log_2 n}_{n \geq 3} < \underbrace{n < n\log_2 n}_{n \geq 3} < n^2 < n^3 < \ldots < \underbrace{2^n < n!}_{n \geq 4}$$

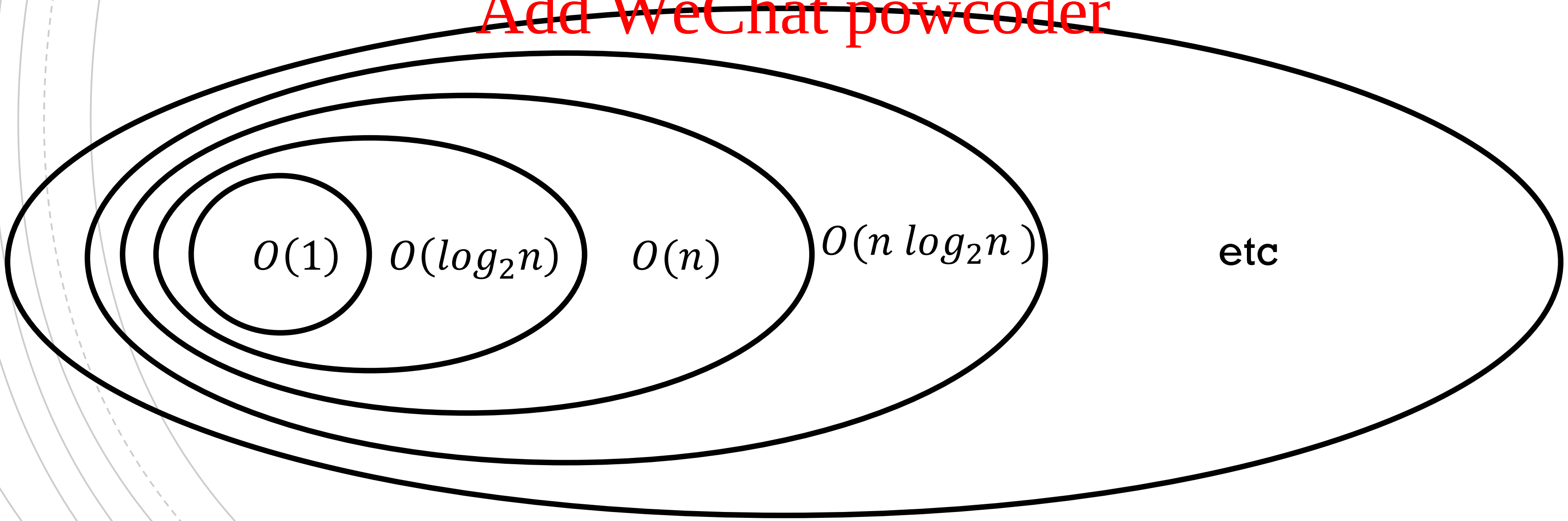
$n^3 < 2^n$ for $n \geq 10$

COMMON FUNCTIONS

Each of the following holds for n sufficiently large:

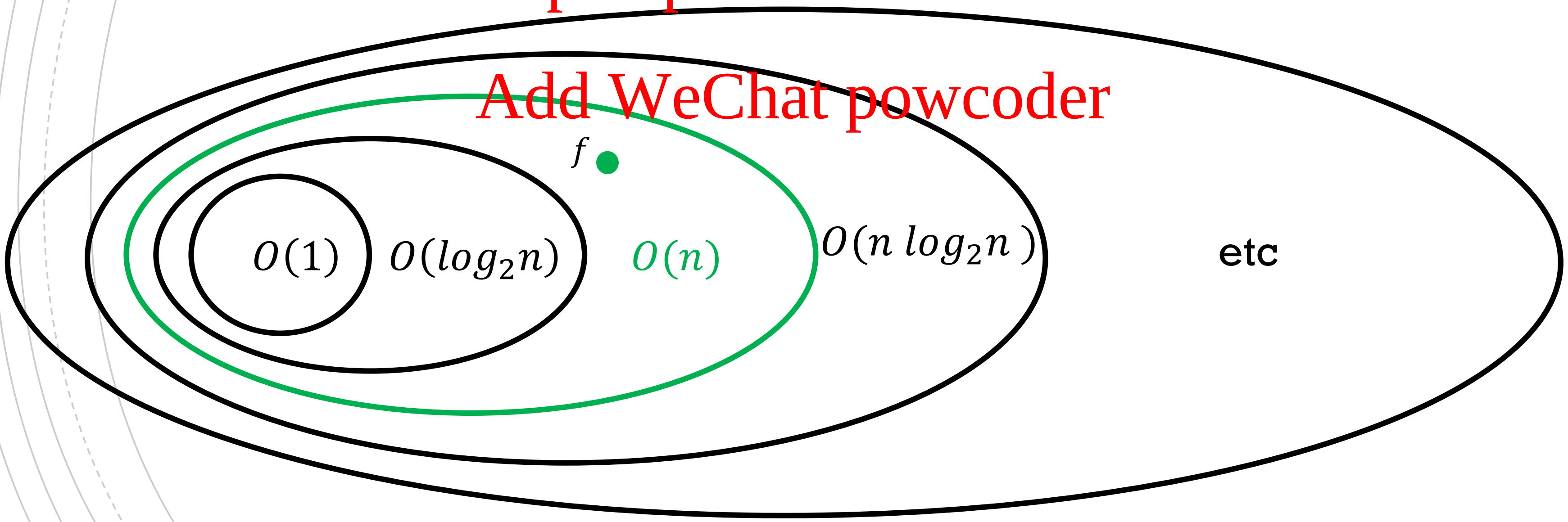
$$1 < \log_2 n < n < n \log_2 n < n^2 < n^3 < \ldots < 2^n < n!$$

Thus we have the following strict subset relationships:



BACK TO TIGHT BOUNDS

If we consider the function $\boldsymbol{f(n) = 5n + 7}$, then the ***tight upper bound*** for f is $O(n)$ and not $O(n \log_2 n)$ for instance.



EXAMPLE

Using these claims/rules allow us to say, for example, that

$$f(n) = 3 + 17\, log_2 n + 4\, n + \frac{n(n+1)}{6} \text{ is } O(n^2).$$

GENERAL OBSERVATION

Never write $O(3n)$, $O(5\,log_2 n)$, etc.

Instead, write $O(n)$, $O(\log n)$, etc.

Why? The set $O(3n)$ is exactly the same set defined by $O(n)$, and so are the others.

It is still *technically* correct to write the above. We just don't do it to avoid dealing with constant factors.

BIG-OMEGA Ω()

ASYMPTOTIC LOWER BOUNDS

Sometimes we want to say that algorithms take *at least* a certain time to run as a function of the input size n.

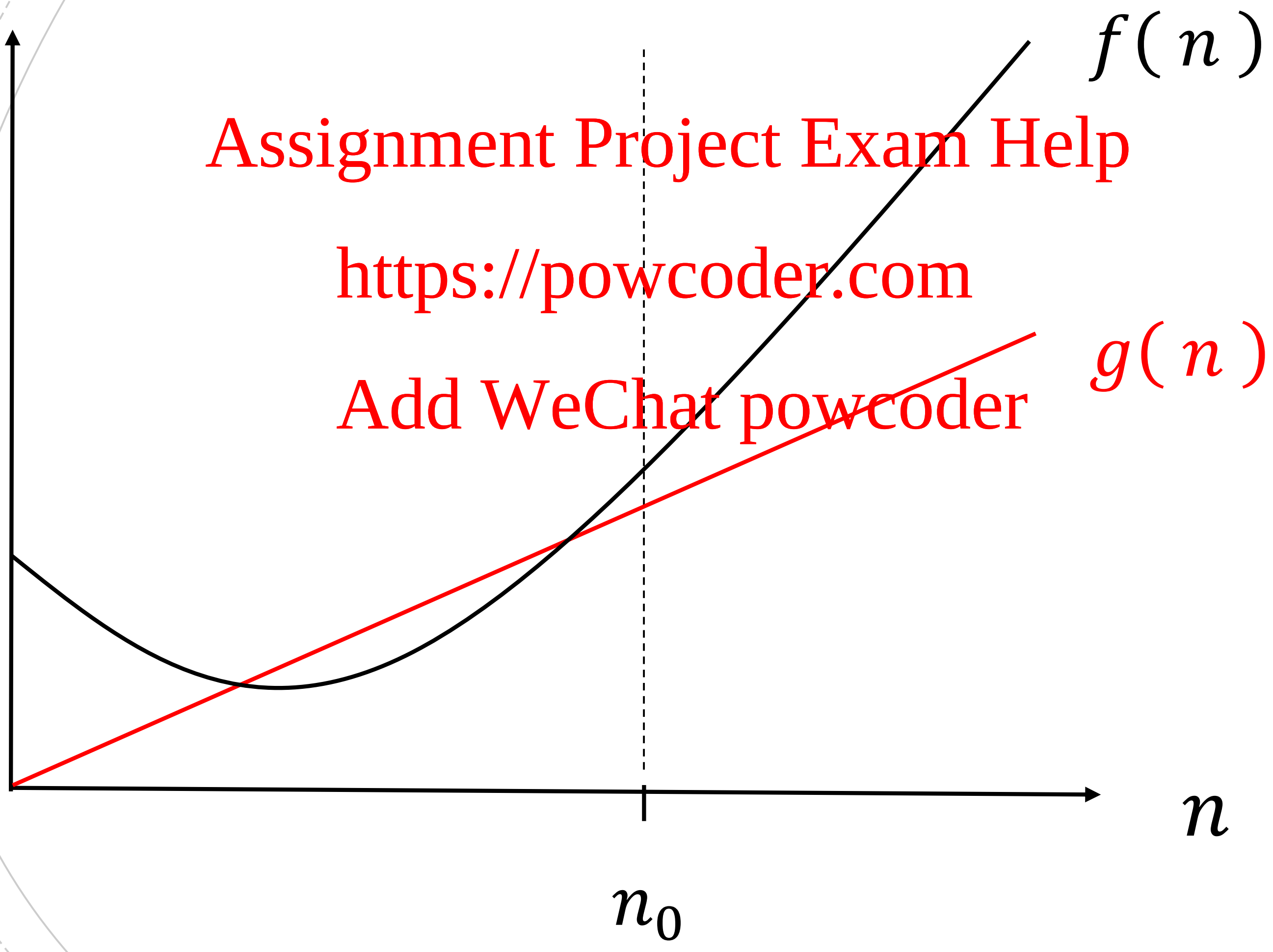
PRELIMINARY DEFINITION (LOWER BOUND)

$f(n)$ is *asymptotically bounded below* by $g(n)$ if there exists an n_0 such that, for all $n \geq n_0$,

$$f(n) \geq g(n).$$

Note: As with big O, the constant n_0 is not unique. If the definition works for some n_0 then it will work for larger n_0 too.

GRAPHICALLY



EXAMPLE

Claim: $f(n) = \frac{n(n-1)}{2}$ is *asymptotically bounded below* by $g(n) = \frac{n^2}{4}$.

To prove: show that there exists an n_0 such that, for all $n \geq n_0$,

$$\frac{n(n-1)}{2} \geq \frac{n^2}{4}$$

EXAMPLE

Claim: $f(n) = \frac{n(n-1)}{2}$ is *asymptotically bounded below* by $g(n) = \frac{n^2}{4}$.

Proof: $\frac{n(n-1)}{2} \geq \frac{n^2}{4}$

$\Leftrightarrow \quad 2n(n-1) \geq n^2$

$\Leftrightarrow \quad 2n^2 - 2n \geq n^2$

$\Leftrightarrow \quad n^2 \geq 2n$

$\Leftrightarrow \quad n \geq 2$

So, we can use $n_0 = 2$.

FORMAL DEFINITION OF BIG OMEGA (Ω)

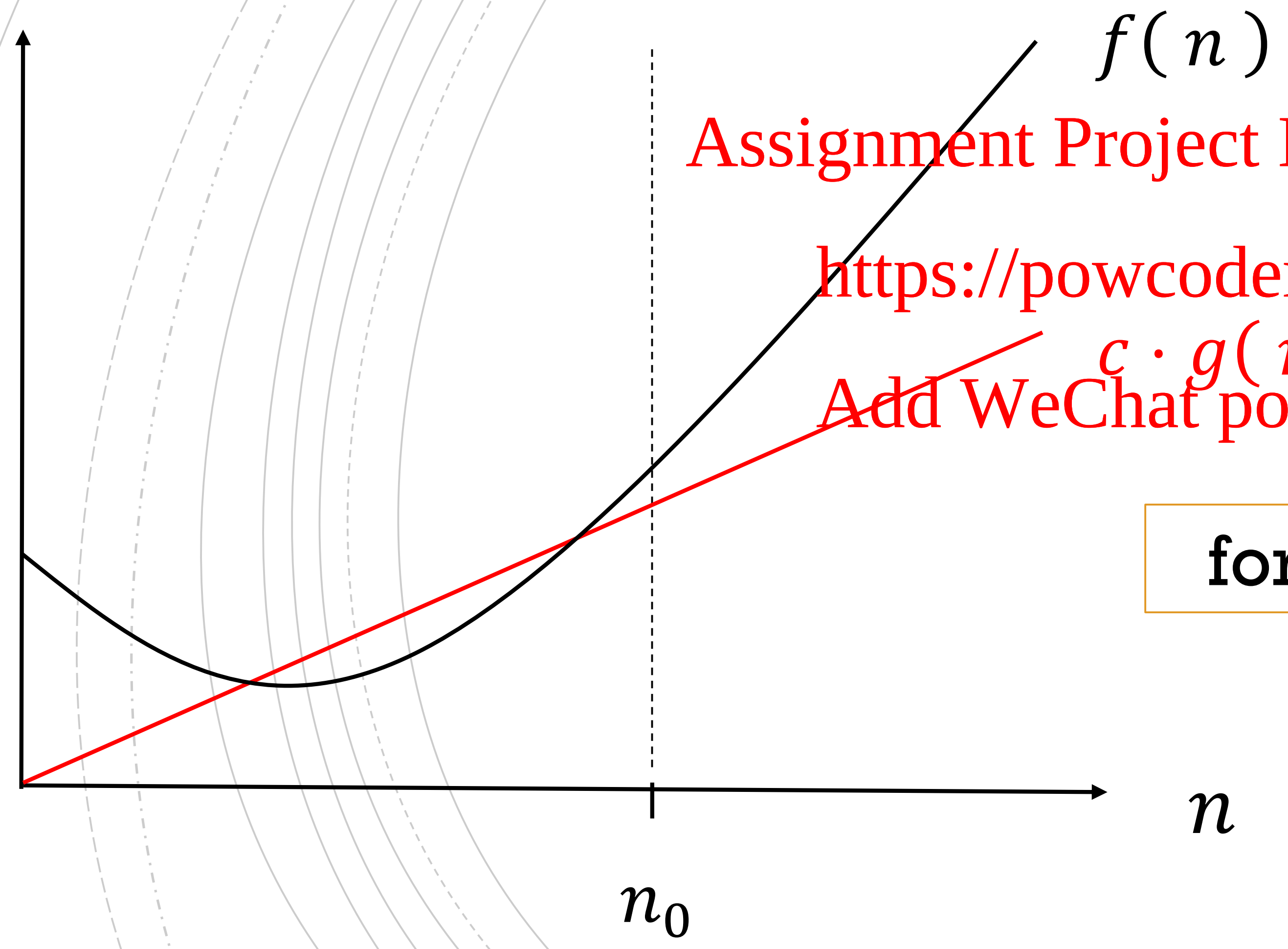
Given a function $g(n)$, we denote by $\Omega(g(n))$ ("big-omega of g of n") the following set of functions

$$\Omega(g(n)) = \{f(n)\colon \text{there exist positive constants } c \text{ and } n_0 \text{ such that}$$

$$f(n) \geq cg(n) \text{ for all } n \geq n_0\ \}$$

We use the Ω-notation to describe an **asymptotic lower bound.**

GRAPHICALLY



for all $n_0 \geq n,\quad f(n) \geq c \cdot g(n)$

EXAMPLE

Claim: $f(n) = \frac{n(n-1)}{2}$ is $\Omega(n^2)$.

Proof(1): Use $c = \frac{1}{4}$ and the derivation we just saw..

$$\frac{n(n-1)}{2} \geq \frac{n^2}{4}$$

$\Leftrightarrow$:

$\Leftrightarrow \quad n \geq 2$

So, we can take $n_0 = 2$ and $c = \frac{1}{4}$

EXAMPLE

Claim: $f(n) = \frac{n(n-1)}{2}$ is $\Omega(n^2)$.

Proof (2): Let's try $c = \frac{1}{3}$

$$\frac{n(n-1)}{2} \geq \frac{n^2}{3}$$

$$\Leftrightarrow \quad 3n(n-1) \geq 2n^2$$

$$\Leftrightarrow \quad n^2 \geq 3n$$

$$\Leftrightarrow \quad n \geq 3$$

So, we can take $n_0 = 3$ and $c = \frac{1}{3}$

BACK TO INSERTION SORT

At the beginning of last lecture we found the function describing the best-case running time for insertion sort.

$$T_{best}(n) = an + b$$

where a, and b are some constants. $a > 0, b < 0$

Claim: $T_{best}(n)$ is $\Omega(n)$

$T_{best}(n)$ IS $\Omega(n)$ – PROOF

Claim: $T_{best}(n)$ is $\Omega(n)$

Proof: $T_{best}(n) = an + b$

$\geq an + bn$, for all $n \geq 1$ since $b < 0$

$= (a + b)n$

So we can take $c = a + b$ (which is positive since it is equal to $T_{best}(1)$) and $n_0 = 1$.

OBSERVATION ON BEST-CASE LOWER BOUNDS

- Since Ω-notation describes a lower bound, when we use it to bound the best-case running time of an algorithm, then we have a lower bound on the running time of the algorithm *on every input*.

  That is,

  Since $T(n) \geq T_{best}(n)$, if $T_{best}(n) = \Omega\big(g(n)\big)$, then $T(n) = \Omega(g(n))$

INSERTION SORT

What do we know about the running time of insertion sort up to know?

- We computed $T(n)$, $T_{best}(n)$, and $T_{worst}(n)$

- We have proved that

  $T_{worst}(n)$ is $O(n^2)$, and

  $T_{best}(n)$ is $\Omega(n)$

- Therefore, $T(n)$ is both $O(n^2)$ and $\Omega(n)$.

TRY IT!

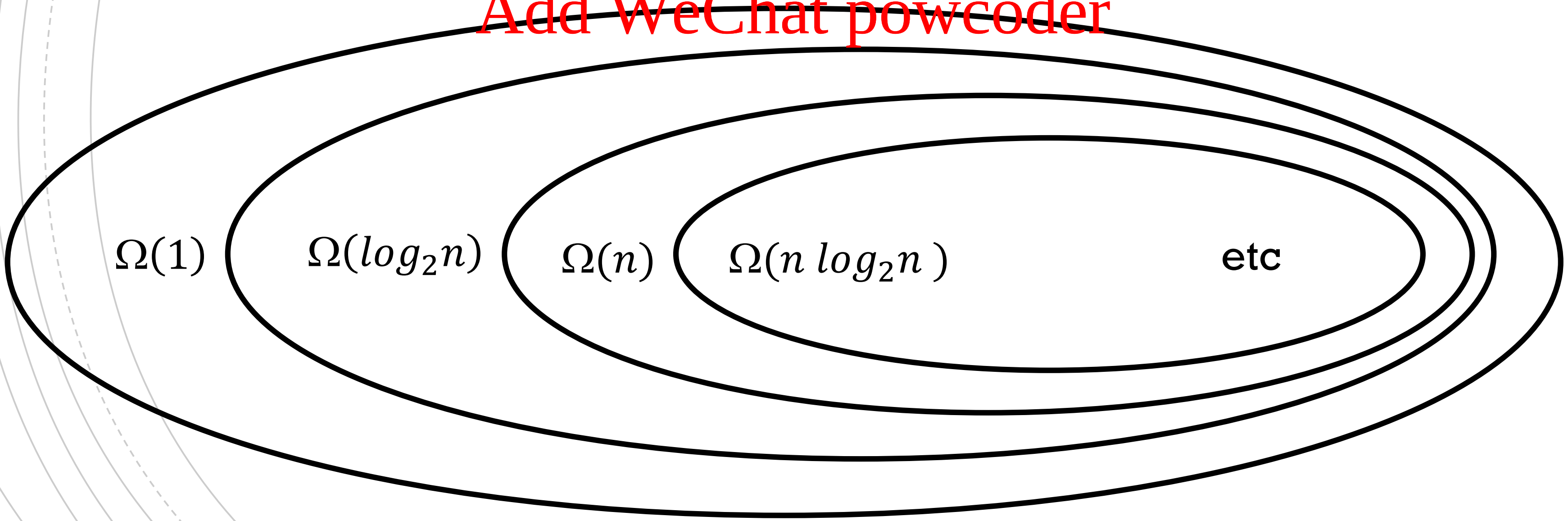
Prove that the scaling, sum, product, and transitivity rules all hold for big Omega also.

BACK TO THE COMMON FUNCTIONS

Each of the following holds for n sufficiently large:

$$1 < \log_2 n < n < n\log_2 n < n^2 < n^3 < \ldots < 2^n < n!$$

Thus we have the following strict subset relationships:



BIG-THETA Θ()

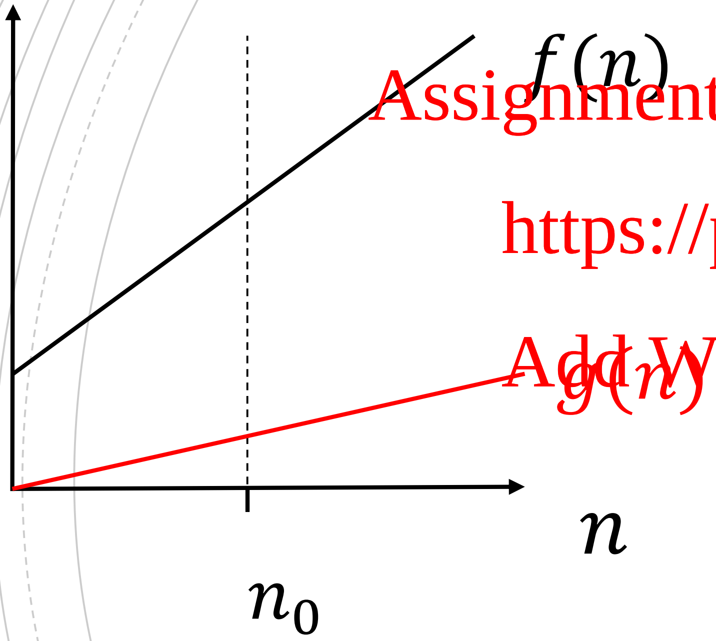
FORMAL DEFINITION OF BIG THETA (Ω)

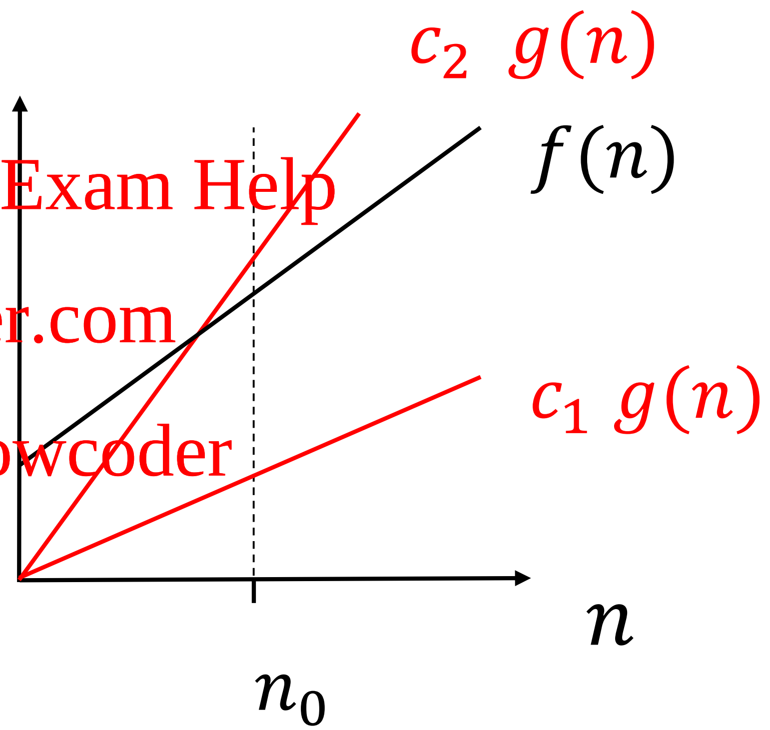
Given a function $g(n)$, we denote by $\Theta(g(n))$ ("big-theta of g of n") the following set of functions:

$$\Theta\big(g(n)\big) = \{f(n)\text{: there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that}$$
$$c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\ \}$$

We use the Θ-notation to describe an **asymptotic tight bound**.

GRAPHICALLY





$$f(n) \text{ is } \Theta(g(n)).$$

EXAMPLE

Claim: $f(n) = \frac{1}{2}n^2 - 3n$ is $\Theta(n^2)$.

Proof: We need to find 3 positive constants c_1, c_2, and n_0 such that

$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2$$

for all $n \geq n_0$.

EXAMPLE

Claim: $f(n) = \frac{1}{2}n^2 - 3n$ is $\Theta(n^2)$.

Proof: We need to find 3 positive constants c_1, c_2, and n_0 such that

$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2$$

for all $n \geq n_0$.
Dividing by n^2 we get

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$$

The right hand inequality holds for all $n \geq 1$ if we chose any $c_2 \geq \frac{1}{2}$.

The left hand inequality holds for all $n \geq 7$ if we chose any $c_1 \leq \frac{1}{14}$.

EXAMPLE

Claim: $f(n) = \frac{1}{2}n^2 - 3n$ is $\Theta(n^2)$.

Proof: We need to find 3 positive constants c_1, c_2, and n_0 such that

$$c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$$

for all $n \ge n_0$.

The right hand inequality holds for all $n \ge 1$ if we chose any $c_2 \ge \frac{1}{2}$.

The left hand inequality holds for all $n \ge 7$ if we chose any $c_1 \le \frac{1}{14}$.

Pick $n_0 = 7, c_1 = 1/14, c_2 = 1/2$.

DEFINITION OF BIG THETA (Θ)

Let $f(n)$ and $g(n)$ be two functions of $n \geq 0$.

We say $f(n)$ is $\Theta(\, g(n)\,)$, if there exist three positive constants

$n_0, \;\; c_1, \;\; c_2$ such that for all $n \geq n_0$,

$$c_1 \; g(\,n\,) \;\leq\; f(\,n\,) \;\leq\; c_2 \; g(\,n\,)$$

$f(n)$ is $O(\, g(n)\,)$

DEFINITION OF BIG THETA (Θ)

Let $f(n)$ and $g(n)$ be two functions of $n \geq 0$.

We say $f(n)$ is $\Theta(\, g(n)\,)$, if there exist three positive constants n_0, c_1, c_2 such that for all $n \geq n_0$,

$$c_1 \; g(n) \;\leq\; f(n) \;\leq\; c_2 \; g(n)$$

$f(n)$ is $\Omega(\, g(n)\,)$

THEOREM

For any two functions $f(n)$ and $g(n)$,

$$f(n) = \Theta(g(n)) \quad \Leftrightarrow \quad f(n) = O\big(g(n)\big) \ and \ f(n) = \Omega(g(n))$$

if and only if

EXAMPLE 2

Claim: $f(n) = 4 + 17\, log_2 n + 3n + 9\, n\, log_2 n + \frac{n(n-1)}{2}$ is $\Theta(n^2)$.

Proof:

$$\frac{n^2}{4} \le f(n) \le \left(4 + 17 + 3 + 9 + \frac{1}{2}\right) n^2$$

In general, you want to set c_1 to be a value that is slightly smaller than the coefficient of the highest-order term and c_2 to be a value that is slightly larger.

DOES A TIGHT BOUND ALWAYS EXIST?

For every $f(n)$, does there exist a "simple" $g(n)$ such that $f(n)$ is $\Theta(g(n))$?

No, as this contrived example shows:

Let $f(n) = \begin{cases} n, & n \text{ is odd} \\ n^2, & n \text{ is even} \end{cases}$

$f(n)$ is $O(n^2)$, but $f(n)$ is *not* $O(n)$.

$f(n)$ is $\Omega(n)$, but $f(n)$ is *not* $\Omega(n^2)$.

DOES A TIGHT BOUND ALWAYS EXIST?

We can also think about the function representing the running time of insertion sort.

$T_{best}(n) \in \Theta(n)$ and $T_{worst}(n) \in \Theta(n^2)$

$$\Rightarrow \quad T(n) \in O(n^2), \; T(n) \in \Omega(n)$$

AND

$$T(n) \notin \Theta(n), \quad T(n) \notin \Theta(n^2)$$

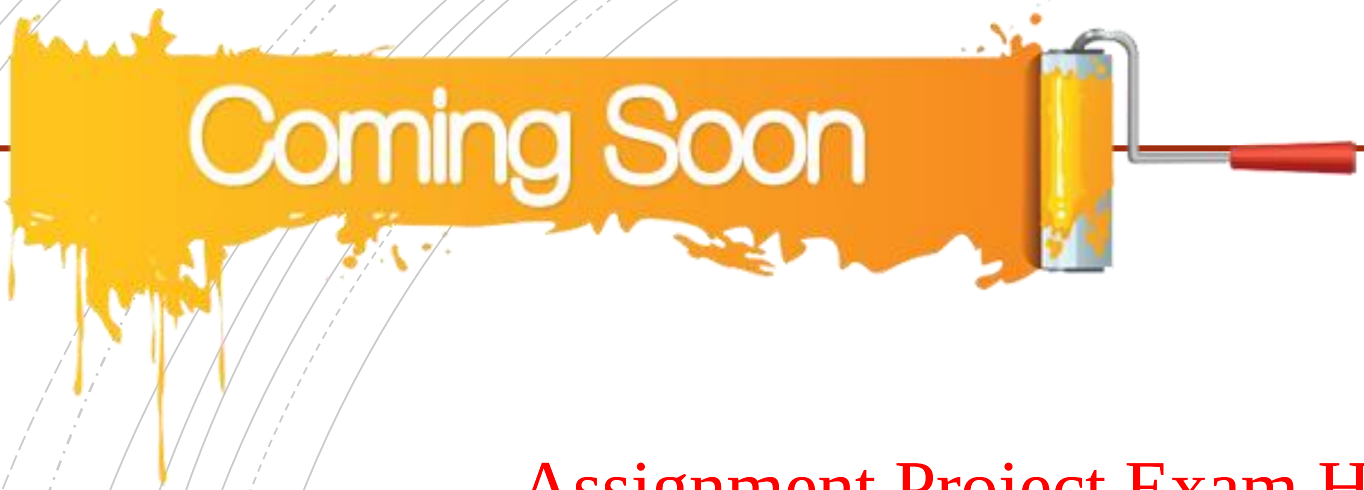
Note that it is improper to say that $T(n)$ is $O(n^2)$ (for instance), since for a given n, the actual running time varies, depending on the particular input. When we say that, what we mean is that there exists a function $f(n)$ which is $O(n^2)$ and T is bounded above by f, no matter the particular input of size n.

SETS OF Θ( ) FUNCTIONS

Each of the following holds for n sufficiently large:

$$1 < \log_2 n < n < n \log_2 n < n^2 < n^3 < \ldots < 2^n < n!$$

$\Theta(1)$ $\Theta(log_2 n)$ $\Theta(n)$ … $\Theta(2^n)$ $\Theta(n!)$



In the next videos:

- Stacks and Queues