COMP 250

Lecture 36

sets of $O(\ )$ functions
rules for big O

big Omega Ω

April 6, 2022

Recall Formal Definition of Big O

Let $t(n)$ and $g(n)$ be two functions, where $n \geq 0$.

We say $t(n)$ is $O(\ g(n)\ )$ if there exist two positive constants n_0 and c such that, for all $n \geq n_0$,

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

We use functions $g(\, n\,)$ below.

Note: The following inequalities hold for n sufficiently large:

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

Thus, we can write big O relationships between them,
e.g. n is $O(\ n \, log_2 n \ )$

Sets of $O(\ )$ functions

If $t(n)$ is $O(\ g(n)\ )$, we often write

$$t(n) \in O(\ g(n)\ ).$$

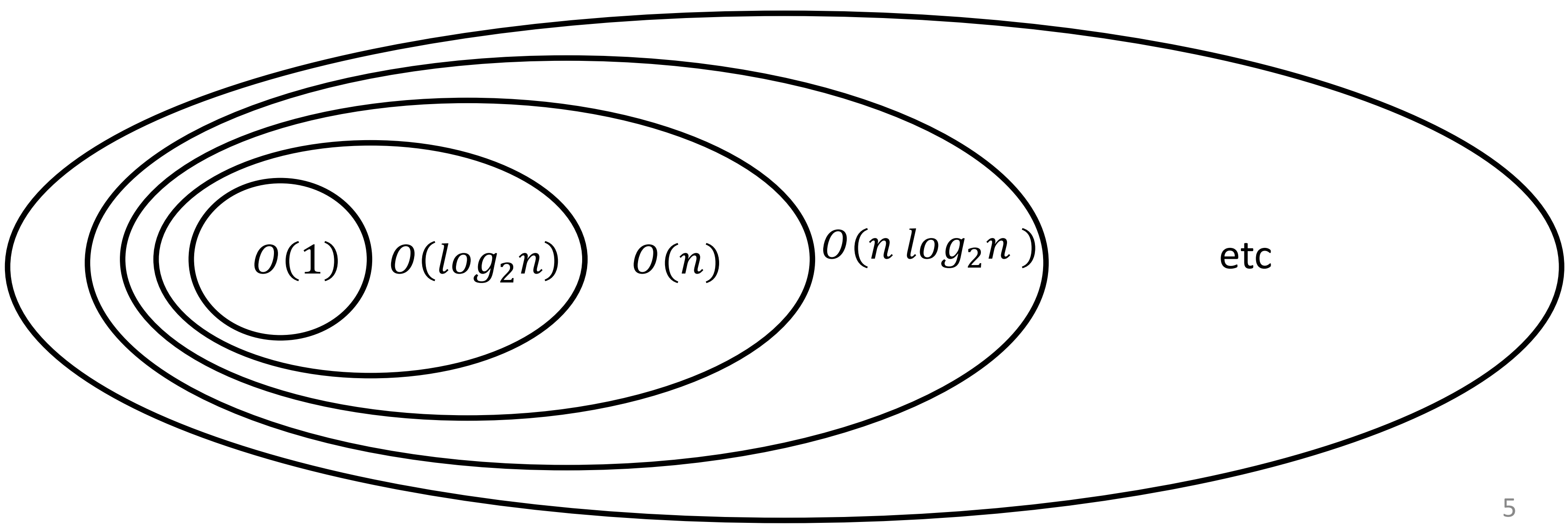
We say:

"$t(n)$ is a member of the set of functions that are $O(\ g(n)\ )$."

Thus we have the following *strict* subset relationships:

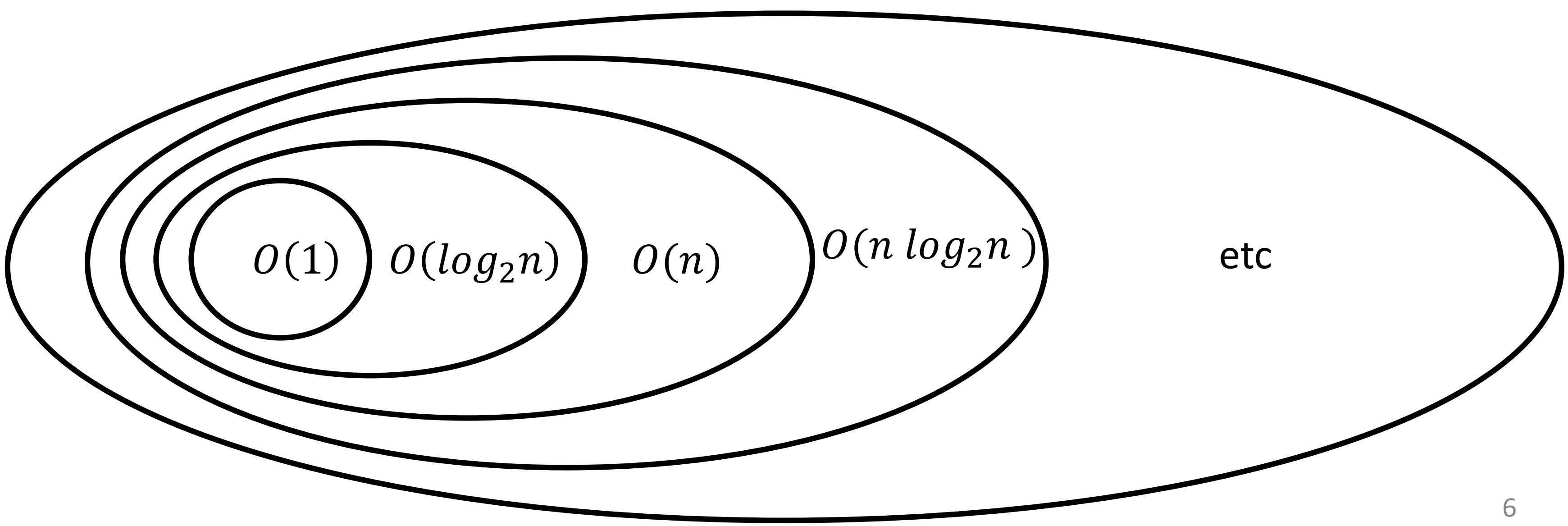
$$O(1) \subset \; O(log_2 n) \; \subset \; O(n) \subset \; O(n\, log_2 n\,) \subset O(n^2)$$

$$\ldots \subset O(n^3) \subset \; \ldots \subset O(2^n\,) \subset O(\,n!\,)$$



Tight Bounds

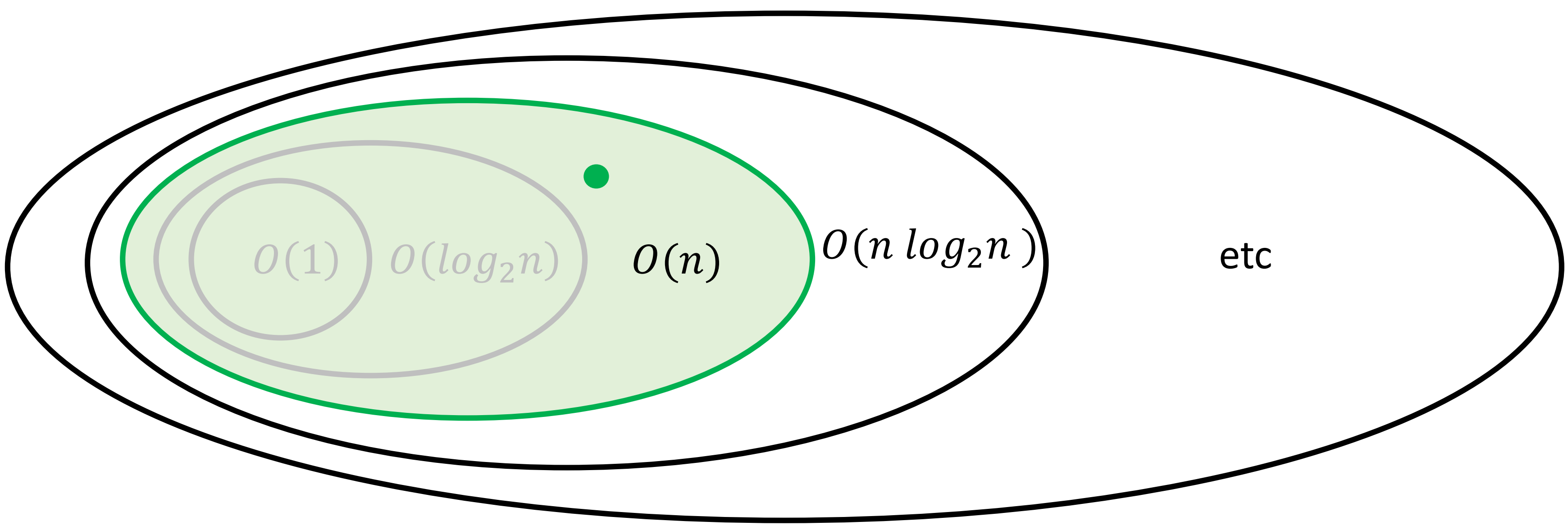
When we say "$t(n)$ is $O(g(n))$", typically we mean the *smallest set* that $t(n)$ belongs to, i.e. *tight bounds*.



Tight Bounds

When we say "$t(n)$ is $O(g(n))$", typically we mean the *smallest set* that $t(n)$ belongs to, i.e. *tight bounds*.

For example, if $t(n) = 5n + 7$, then the tight bound is $O(n)$ rather than $O(n \, log_2 n)$ or something even larger.



If we have some function $t(n)$ that is defined by a complicated expression, we would like to say "$t(n)$ is $\mathrm{O}\big(g(n)\big)$" where $g(n)$ is a simple function.

e.g. $t(n) = 5\, n\, log_2\, (n+3) + 17n\, + 4$ is $O(n\, log_2\, n)$.

What are the general rules to justify *using a simple function* ?

Scaling Rule

Suppose $f(n)$ is $\mathrm{O}(\, g(n)\,)$ and let $a > 0$.

Then $a\, f(n)$ is also $\mathrm{O}(\; g(n)\;)$.

So, multiplying a function by a scale factor doesn't change the big O set(s) that it belongs to.

If you understand the definition of big O, then this rule is obvious. Let's prove it anyhow.

Scaling Rule

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 Rule

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

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

or equivalently, $a\, f(\, n\,) \;\leq\; \underbrace{a\, c}\;\; g(\, n\,)$ where $a > 0$.

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

Sum Rule

Motivation: When terms are added, we only need to consider the term with the largest big O bound.

For example,

$$\mathbf{3 + 5}\boldsymbol{n} \ \textbf{is} \ \mathbf{O}(\boldsymbol{n})$$

$$\mathrm{O}(1) \qquad \mathrm{O}(n)$$

Sum Rule

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

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

Proof: There are constants n_1, c_1 and n_2, c_2 such that

$f_1\,(n) \;\leq\; c_1\; g_1(n)$ for all $n \;\geq n_1$

$f_2\,(n) \;\leq\; c_2\; g_2(n)$ for all $n \;\geq n_2$.

Thus, $f_1\,(n) + f_2\,(n) \;\leq\; (c_1 \;+ c_2\,)$ max($g_1(n)\,, g_2(n)$)

for all $n \;\geq$ max($n_1\,,\; n_2$)

Product Rule

We want to be able to say, for example,

$$t(n) = (3 + 5n)\, log_2(n + 7) \quad \text{is} \quad O(n\, log_2 n)\ .$$

$$O(n) \qquad\qquad O(log_2 n)$$

i.e. if two functions are multiplied together, then the O( ) of their product is the product of their O( )'s.

Product Rule

Suppose $f_1\,(n)$ is $O(\,g_1\,(n))$ 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

$$f_1\,(n) \;\le\; c_1\;g_1(n), \quad \text{for all } n \;\ge\; n_1$$
$$f_2\,(n) \;\le\; c_2\;g_2(n), \quad \text{for all } n \;\ge\; n_2.$$

So, $f_1\,(n) \;*\; f_2\,(n) \le\; c_1\;c_2\;g_1(n)\,g_2(n)$

for all $n \;\ge\; \max(\,n_1, n_2\,)$

It is because of these rules that we can say, for example:

$$t(n) = 5\, n\, log_2\,(n + 3) + 17n\, + 4 \qquad \text{is} \qquad O(n\, log_2\, n).$$

COMP 250

Lecture 36

sets of $O(\,)$ functions
rules for big O

big Omega Ω

April 6, 2022

“small omega” ω

“big omega” Ω

Big Omega (Ω): asymptotic lower bound

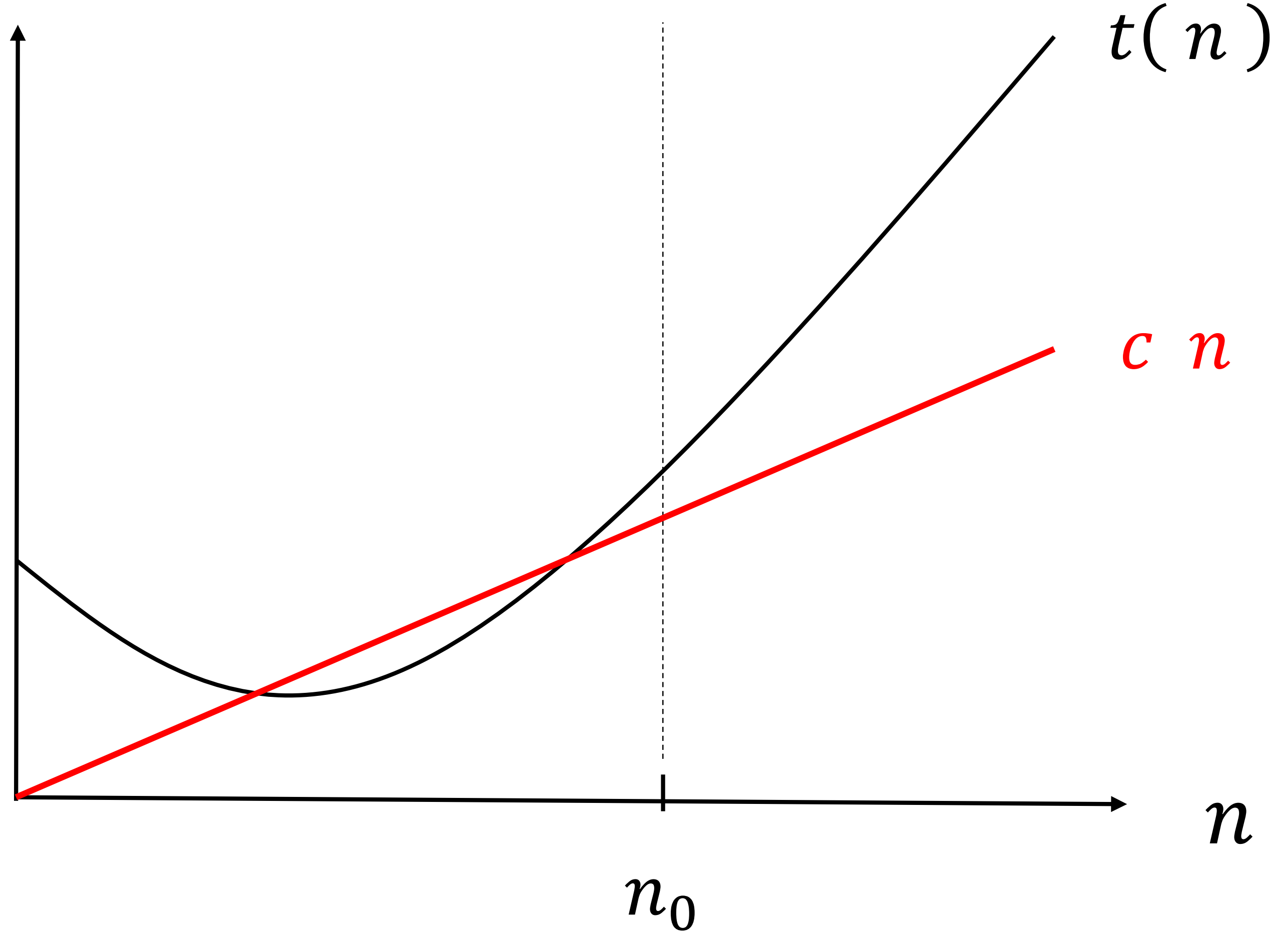
Sometimes we want to say that an algorithm takes *at least* a certain time to run, as a function of the input size n.

Example 1:

Let $t(n)$ be the time it takes for algorithm X to *find the maximum value* in an array of n numbers.

Then $t(n)$ is $\Omega(n)$. (This should be intuitively obvious.)

e.g. $t(n)$ is $\Omega(n)$



Big Omega (Ω): asymptotic lower bound

Example 2: (Comparison based sorting)

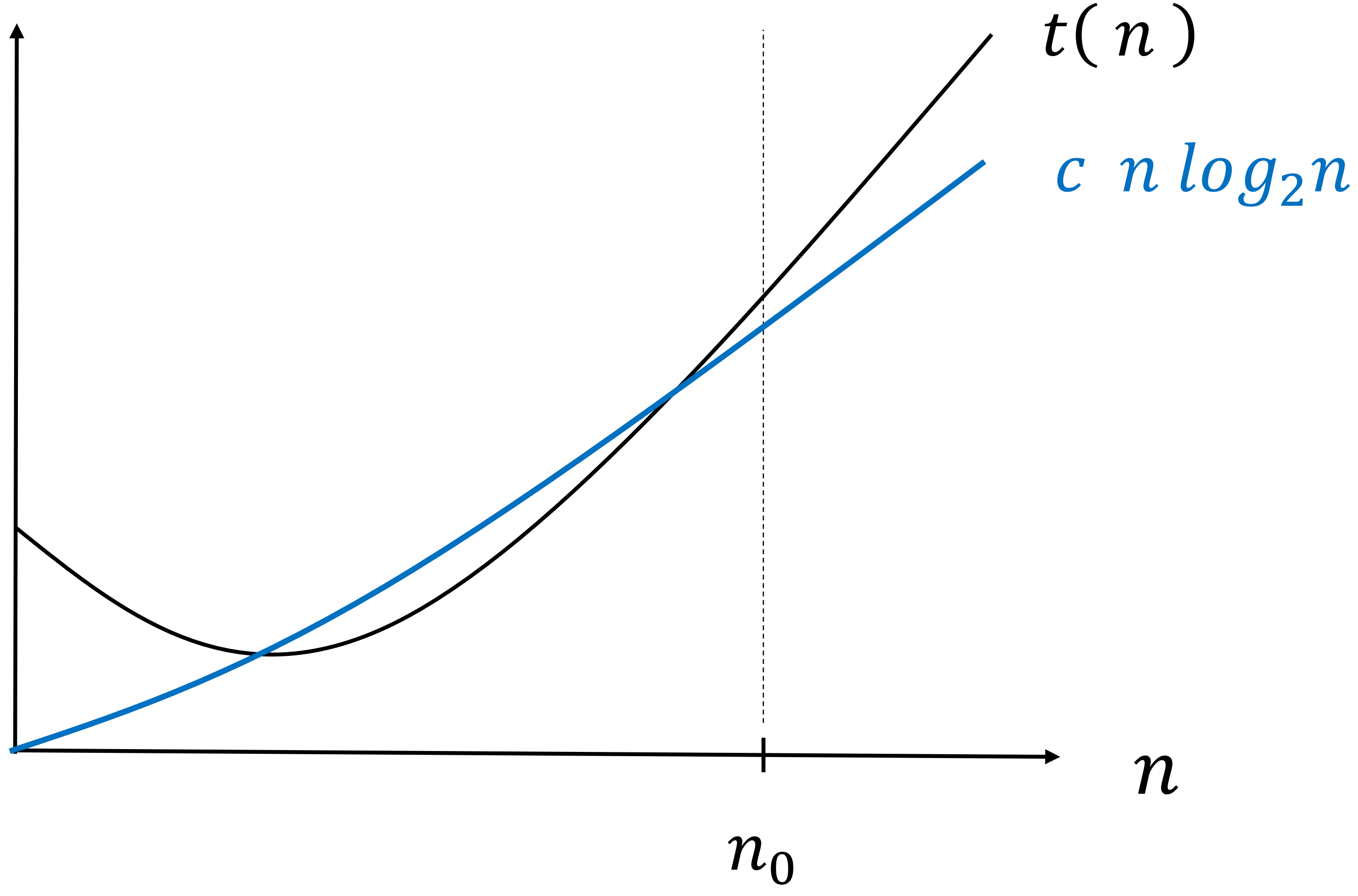
Let $t(n)$ be the number of element comparisons used by some algorithm (X) to sort an array of n numbers.

*One can prove** that $t(n)$ is $\Omega(n\, log_2\, n)$

That is, no faster comparison-based sorting algorithm is possible than the ones we have seen (e.g. X = merge/heap/quicksort).

*[Updated after lecture: Strictly speaking, this is a statement about *on average case. You will cover this in COMP 251.]*

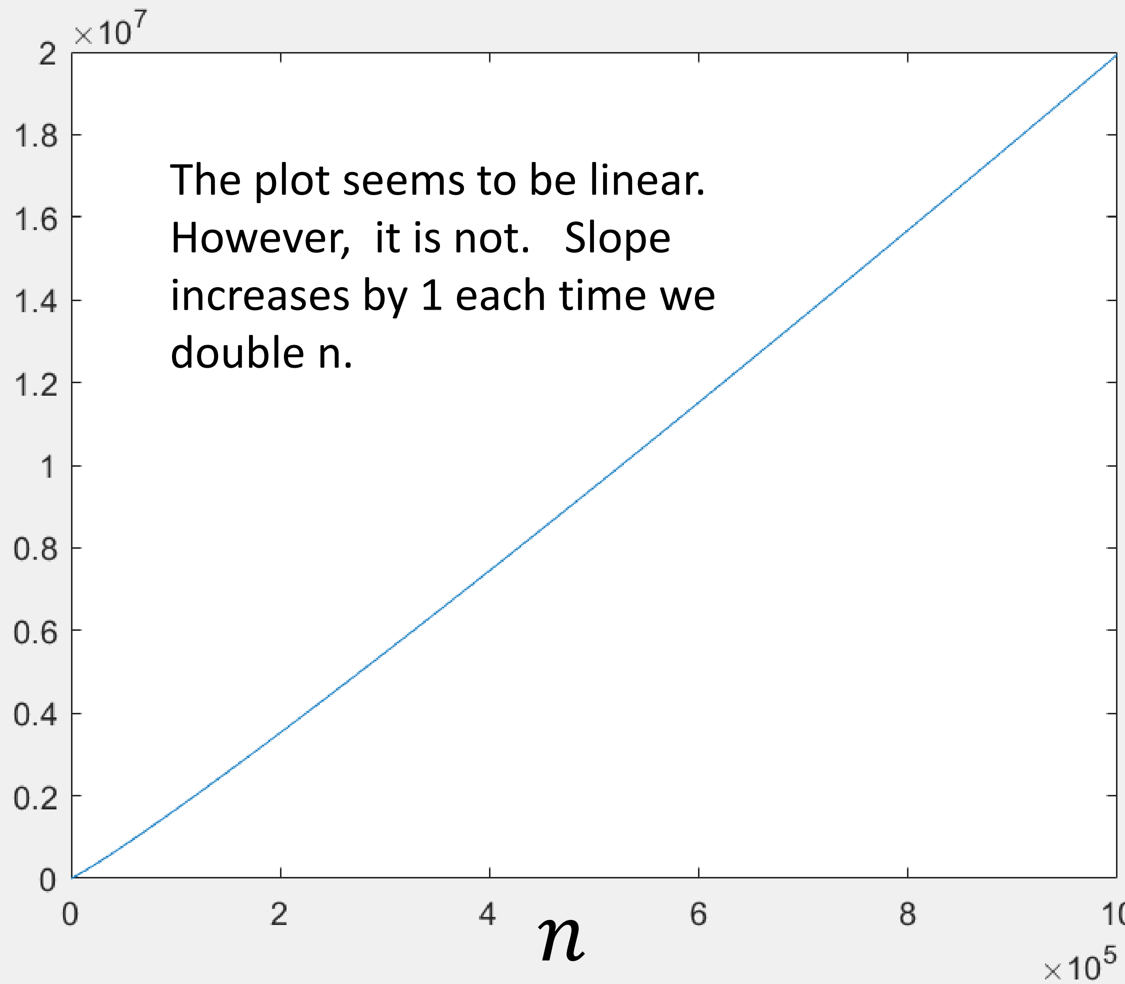
e.g. $t(n)$ is $\Omega(\, n\, log_2 n\,)$



Plot of $n\,log_2 n$ vs. n

$17 * 10^6$

$n\,log_2 n$



n

$10 * 10^5$
$= 10^6$
$\approx 2^{17}$

Towards a Formal Definition of Big Ω

Let $t(n)$ and $g(n)$ be two functions, where $n \geq 0$.
We say $t(n)$ is ***asymptotically bounded below*** by $g(n)$
if there exist a constant n_0 such that, for all $n \geq n_0$,

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

Note that $g(n)$ here might not be a simple function.

Formal Definition of Big Omega (Ω)

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

We say $t(n)$ is $\Omega(\ g(n)\ )$ if there exist two positive constants n_0 and c such that, for all $n \geq n_0$,

$$t(\ n\ ) \geq c\ g(\ n\ ).$$

As with big O, having a constant c lets $g(\ n\ )$ be a simple function.

Example : $t(n) = \frac{n(n-1)}{2}$. $t(n)$ is $\Omega\left(n^2\right)$.

Proof: How to choose c ?

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

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

Proof: Try $c = \frac{1}{4}$.

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

Heads up!
This inequality may be either true or false, depending on n.

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

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

$\Leftrightarrow \quad n \geq 2$. So we can take $n_0 = 2$.

"$\Leftrightarrow$ " means "if and only if" i.e. same true/false value

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

Proof (2): Try $c = \frac{1}{3}$

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

$\Leftrightarrow$: ← you can fill this in

$\Leftrightarrow$ $n \geq 3$

So take $n_0 = 3, \quad c = \frac{1}{3}$.

Relationship of Big O and Big Omega (Ω)

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

The following are equivalent statements:

$$f(n) \text{ is } \mathrm{O}(\ g(n)\ )$$

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

Why?

$$f(n)\ <\ c\ g(n) \qquad \equiv \qquad g(n)\ >\ \frac{1}{c}\ f(n)$$

Sets of $\Omega(\ )$ functions

If $t(n)$ is $\Omega(\ g(n)\ )$, we often write or say:

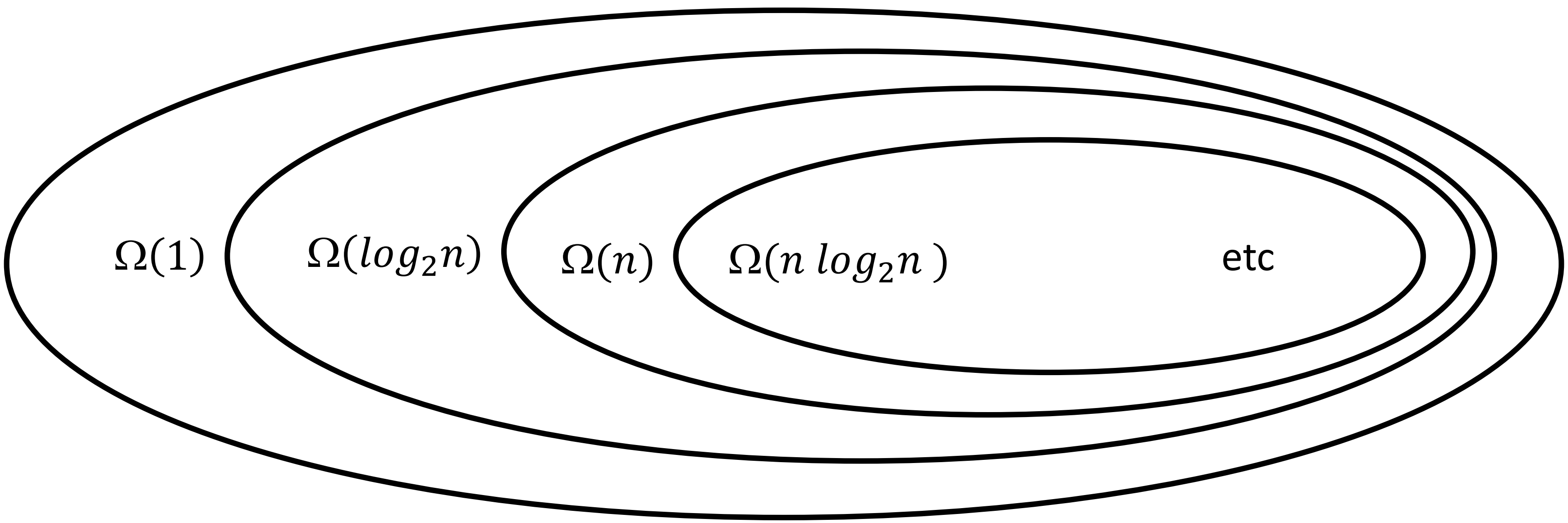
$$t(n) \in \Omega(\ g(n)\ ).$$

$t(n)$ is a member of the set of functions that are $\Omega(\ g(n)\ )$.

As with big O, we have strict subset relationships :

$$\Omega(1) \supset \Omega(log_2 n) \supset \Omega(n) \supset \Omega(n\, log_2 n)$$

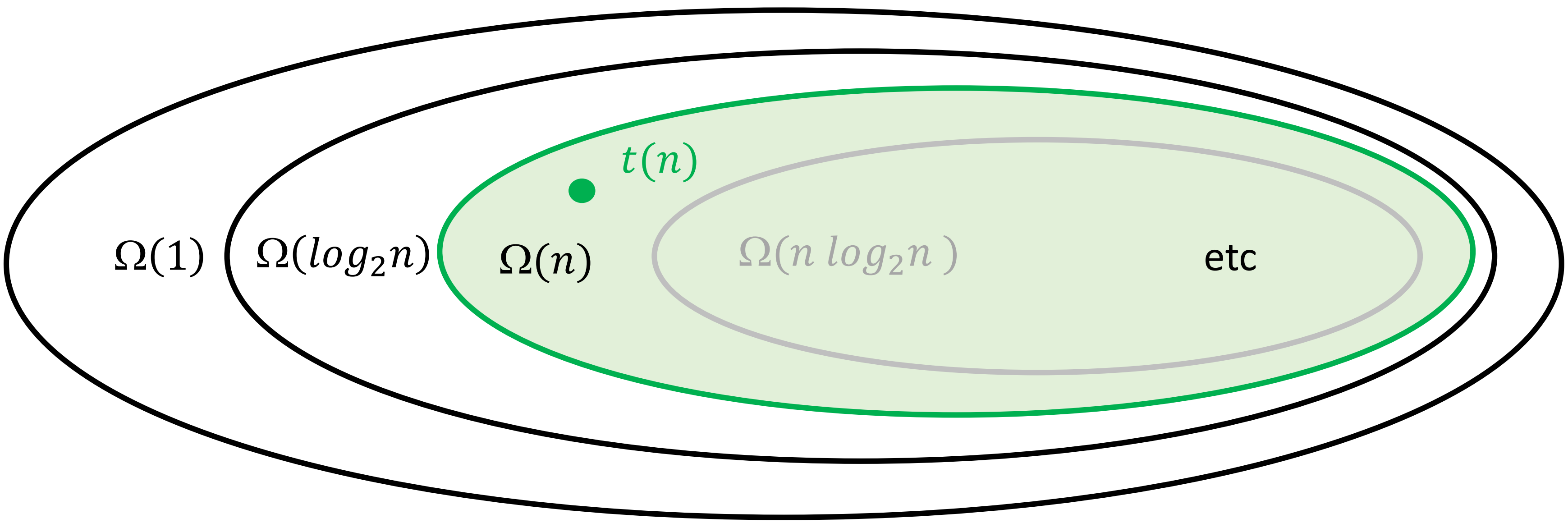
$$\supset \Omega(n^2) \dots \supset \Omega(n^3) \supset \dots \Omega(2^n) \supset \Omega(n!)$$



Note the biggest set is now $\Omega(1)$. $e.g.$ any positive non-decreasing function $t(n)$ will be bounded below by a constant.

For example, if $t(n)$ belongs to $\Omega(n)$, then $t(n)$ also belongs to $\Omega(log_2 n)$ and to $\Omega(1)$.

We can again talk about **tight lower bounds**. For example, $\Omega(n)$ is a tight lower bound for $t(n)$ in the example below.
i.e. $\Omega(n)$ is the smallest $\Omega(\ )$ set that contains $t(n)$.



Exercises (see PDF)

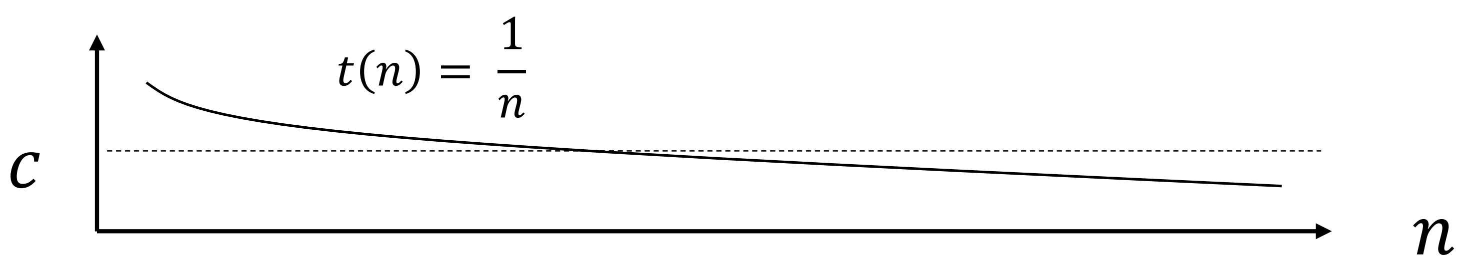
Q 12: Let $t(n) = \frac{1}{n}$. Is $t(n) \in \Omega(1)$?

A: No. Apply the definition:

$t(n)$ is $\Omega(1)$ if there exist two constants $n_0 > 0$ and $c > 0$

such that, for all $n \geq n_0$, $t(n) \geq c$.

But this is impossible. See below.



Coming up...

Lectures

Fri : April 8

big Theta, best and worst cases

Mon Aprill 11 (class cancelled)

I will try to have OH before final exam.

Assessments

Assignment 4 due today.

Final Exam Thurs. April 21 (2 weeks)