COMP 250

Lecture 36

rules for big O
big Omega Ω
big Theta Θ

Dec. 3, 2018

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 $n \geq n_0$,

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

By convention, we will use simple functions $g(n)$ below.

$$1 \qquad log_2 n \qquad n \qquad n\, log_2 n \qquad n^2 \qquad n^3 \qquad \ldots \qquad 2^n \qquad n!$$

There are other functions like $log_2\,(log_2 n)$ that one can use too, but not in COMP 250!

By convention, we will use simple functions $g(n)$ below.

<u>Claim:</u> 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 $n \geq 10$

Thus, we can write big O relationships between them.

Sets of $O(\ )$ functions

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

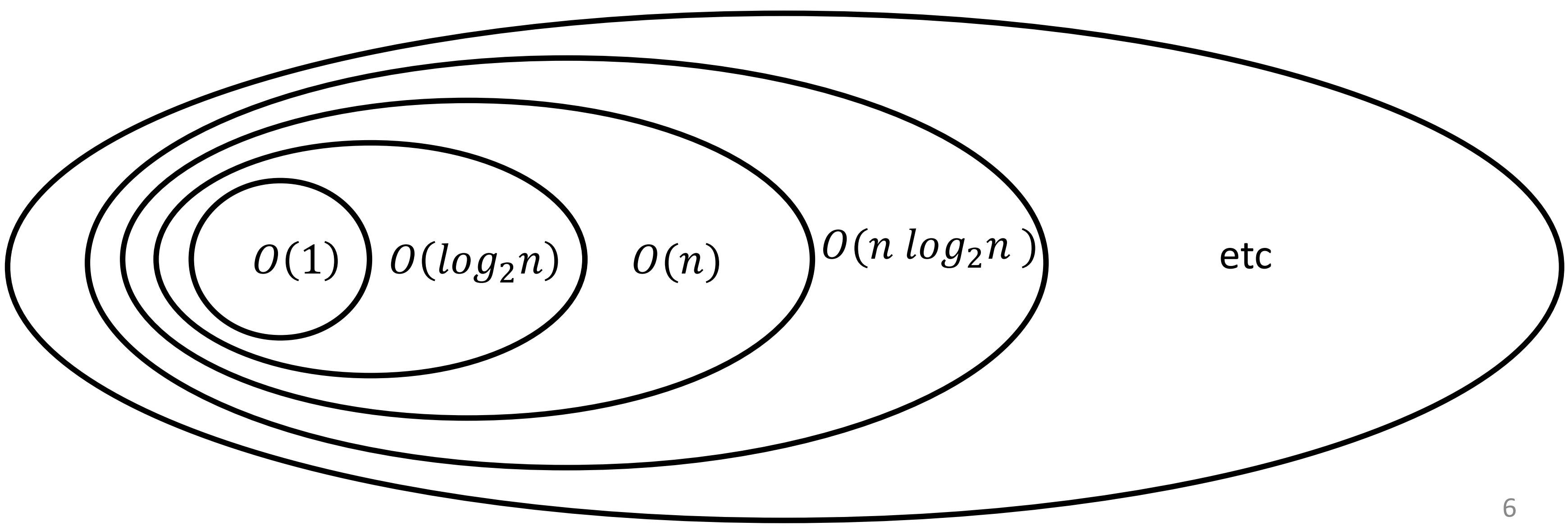
$$t(n) \in O(\ g(n)\ ).$$

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

Thus we have the following *strict* subset relationships:

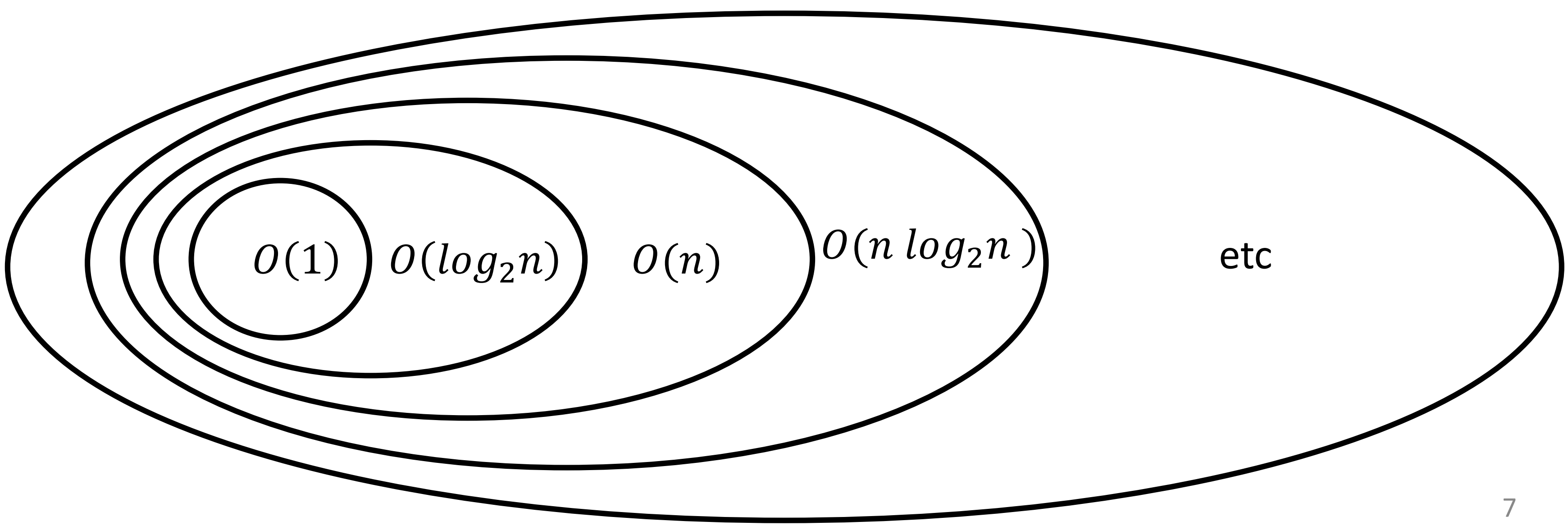
$$O(1) \subset O(log_2 n) \subset O(n) \subset O(n\, log_2 n) \subset O(n^2) \ldots$$

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



Recall "Tight Bounds"

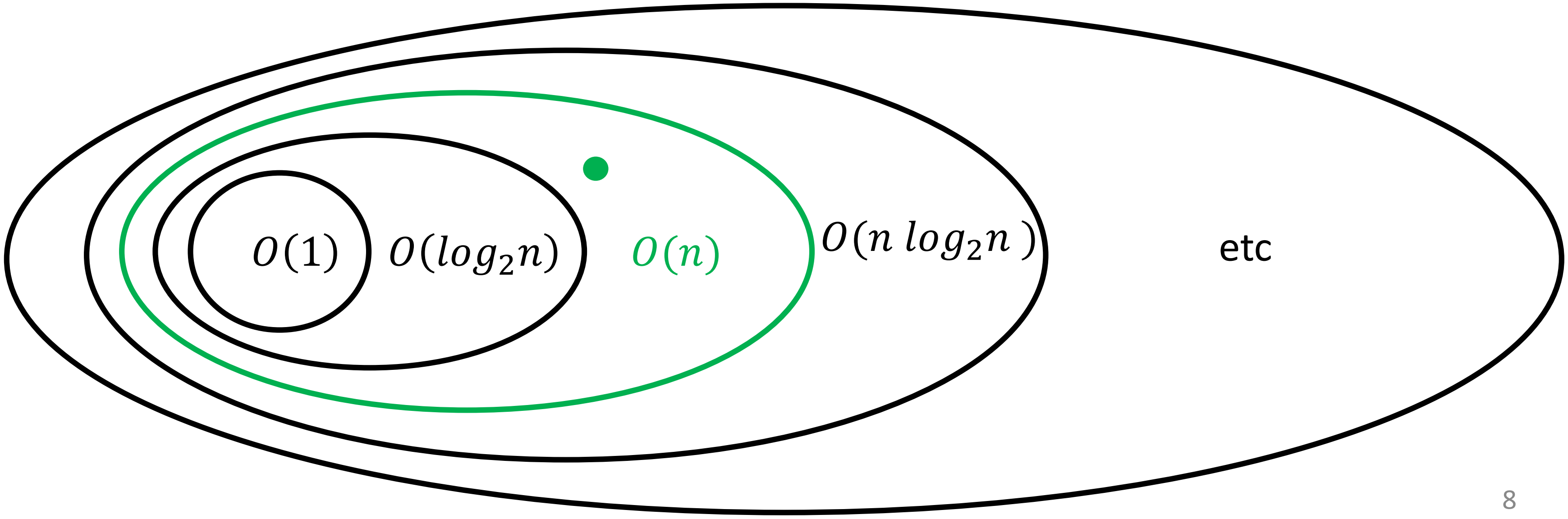
When we say $t(n)$ is $O(g(n))$, we typically want to know the smallest set that $t(n)$ belongs to, i.e. tight bounds.



Recall "Tight Bounds"

When we say $t(n)$ is $O(g(n))$, we typically want to know 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 $O(n\ log_n n)$ or something larger.



If we have some function $t(n)$ that is defined by a complicated expression, we would still like to be able to *informally* look for the term with the largest growth rate and say that $t(n)$ is $\mathrm{O}\big(g(n)\big)$.

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

Are there any general *formal* rules to justify this?

Yes!

Constant Factor Rule

Suppose $f(n)$ is $O(\, g(n)\,)$ and a is a positive constant.

Then $a\, f(n)$ is also $O(\, g(n)\,)$

(This rule is obvious if you understand the definition of big O.
But let's prove it anyhow...)

Proof of Constant Factor 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,

?

Proof of Constant Factor Rule

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

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

Equivalently, if $a > 0$ then

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

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

Sum Rule

Motivation: We want to be able to say, for example,

$$3 + 5n \text{ is } O(n)$$

because the second term has a bigger O( ) bound than the first term.

i.e. when two terms are added, then we need to consider only the term with the largest big O bound.

Sum Rule

3 $\quad O(1)$ $\quad 5n$ $\quad O(n)$

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

and $g_1(n)$ is $O\big(g_2(n)\big)$.

1 $\quad n$

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

$3 + 5n$ $\quad O(n)$

Sum Rule Proof (apologies!)

Suppose $f_1\,(n)$ is $\mathrm{O}(g_1(n))$ and $f_2(n)$ is $\mathrm{O}\big(g_2(n)\big)$ and $g_1(n)$ is $\mathrm{O}\big(g_2(n)\big)$

Then $f_1\,(n) + f_2\,(n)$ is $\mathrm{O}\big(g_2(n)\big)$.

Proof: Let $n_1, c_1, n_2, c_2\,, n_3, c_3$ be constants 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 $n \;\geq\; n_2$.

$g_1\,(n) \;\leq\; c_3\, g_2\,(n)$ for all $n \;\geq\; n_3$.

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

when $n \;\geq max\{n_1, n_2\,\}$

Sum Rule Proof (apologies!)

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

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

Proof: Let $n_1, c_1, n_2, c_2\ , n_3, c_3$ be constants 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 $n\ \geq\ n_2$.

$g_1\ (n)\ \leq\ c_3\ g_2\ (n)$ for all $n\ \geq\ n_3$.

Then, $f_1(n)\ +\ f_2(n)\ \leq\ c_1\ g_1\ (n) + c_2\ g_2\ (n) \leq c_1\ c_3\ g_2\ (n) + c_2\ g_2\ (n)$

when $n\ \geq max\{n_1,\ n_2\,,n_3\}$. Take $c\ =\ c_1\ c_3\ +c_2$.

Product Rule (Motivation)

We want to be able to say, for example,

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

In general, if two terms are multiplied, then the O( ) bound of their product is the product of their O( ) bounds.

Product Rule

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

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

Product Rule (Proof)

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

Then $f_1\,(n) * f_2\,(n)$ is $\mathrm{O}\big(\,g_1(n) * g_2(n)\big)$.

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 $n \;\ge\; n_1$

$f_2\,(n) \;\le\; c_2\; g_2(n)$ for all $n \;\ge\; n_2$.

Product Rule (Proof)

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

$$f_1\,(n) \;\le\; c_1\; g_1(n) \text{ for all } n \;\ge\; n_1$$

$$f_2\,(n) \;\le\; c_2\; g_2(n) \text{ for all } n \;\ge\; n_2.$$

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

These constants satisfy the big O definition

COMP 250

Lecture 36

rules for big O

big Omega Ω

Dec. 3, 2018

“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.)

Big Omega (Ω): asymptotic lower bound

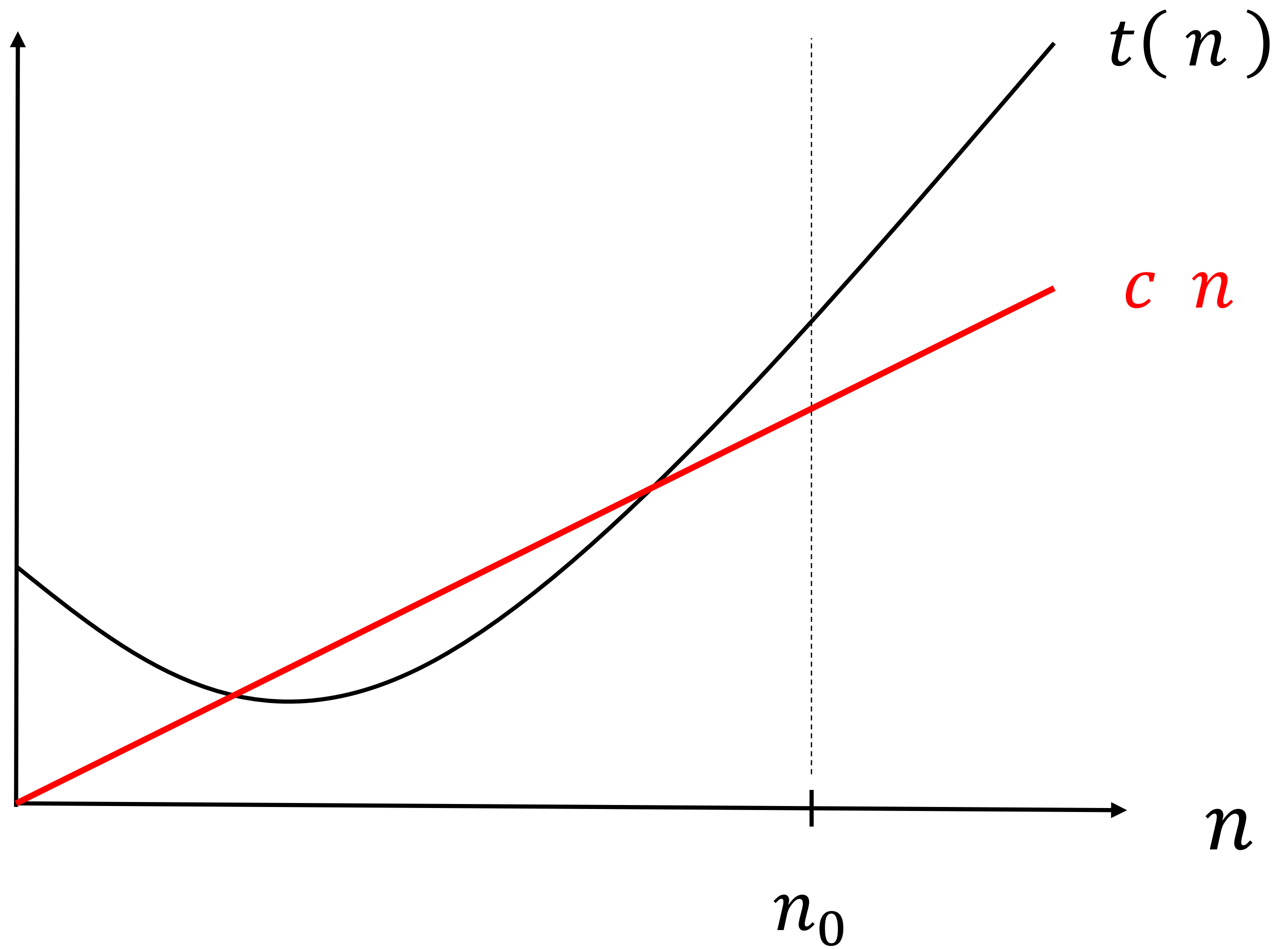
Example 2: (Comparison based sorting)

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

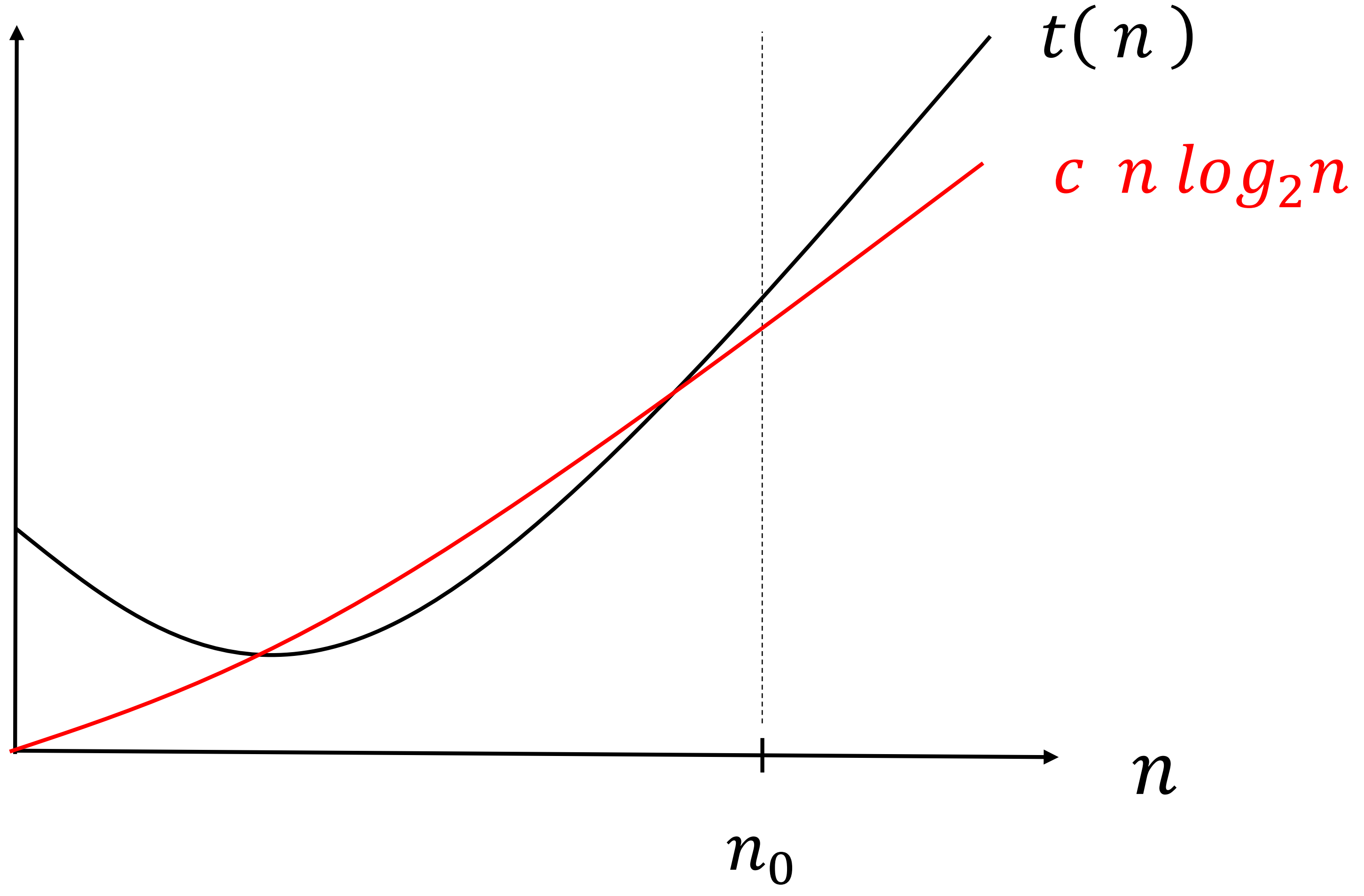
One can prove that $t(n)$ is $\Omega(n\ log_2\ n)$ e.g. COMP 251

That is, there is no faster algorithm possible than the ones we have seen (e.g. merge/heap/quicksort)

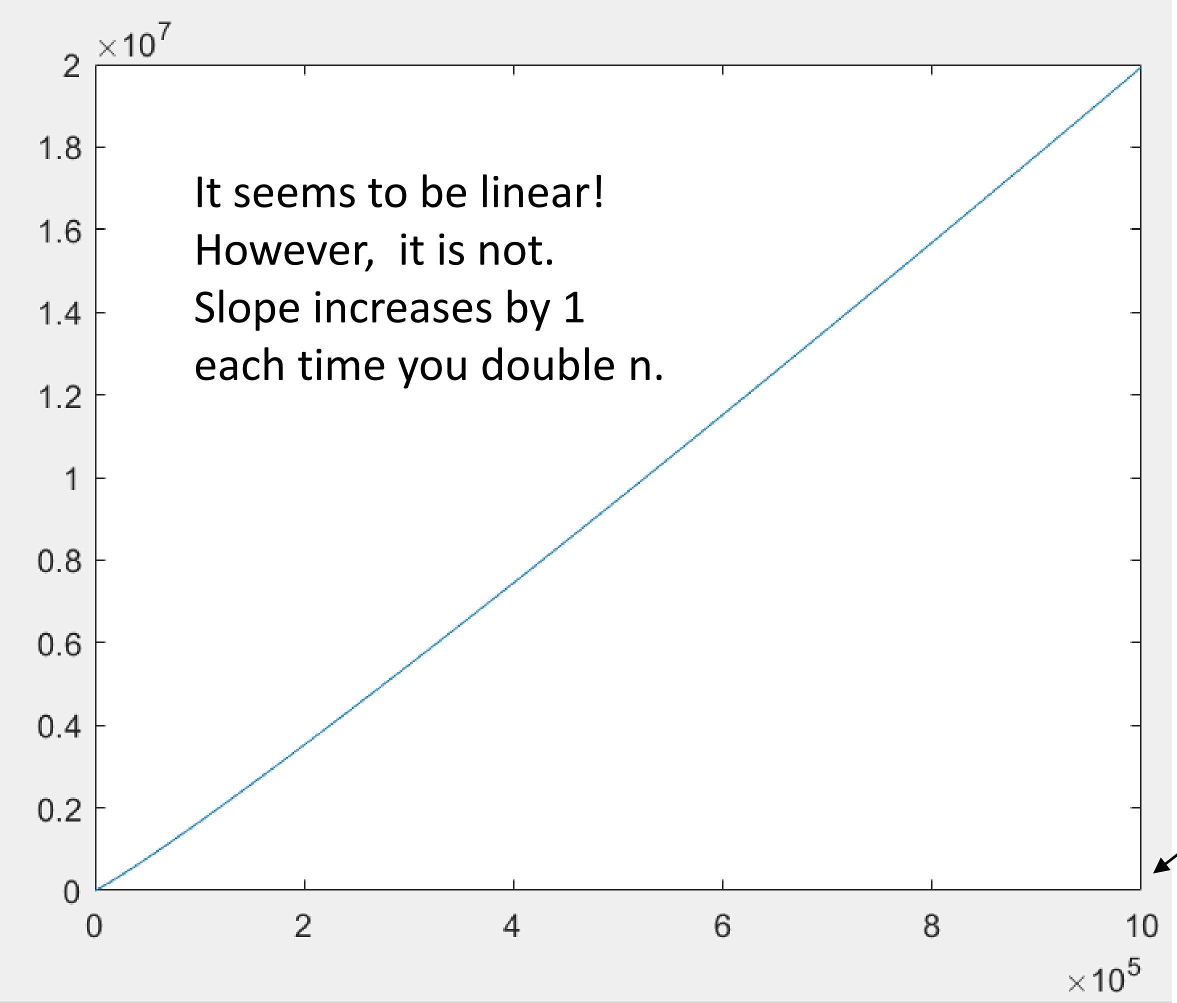
e.g. $t(n)$ is $\Omega(n)$



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



Heads up: Plot of $n \, log_2 n$



$10^6 \approx 2^{17}$

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 $n \geq n_0$,

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

Formal Definition of 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?

Formal Definition of Big Omega (Ω)

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

The following are equivalent statements:

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

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

i.e. For any n, $\quad f(n) < c\, g(n) \;\Leftrightarrow\; g(n) > \frac{1}{c} f(n)$

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

Proof:

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

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

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

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

Heads up!
For each n, this inequality may be either true or false.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 1: $t(n) = \frac{n(n-1)}{2}$ 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 $n_0 = 3$, $c = \frac{1}{3}$.

Sets of $\Omega(\,)$ functions

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

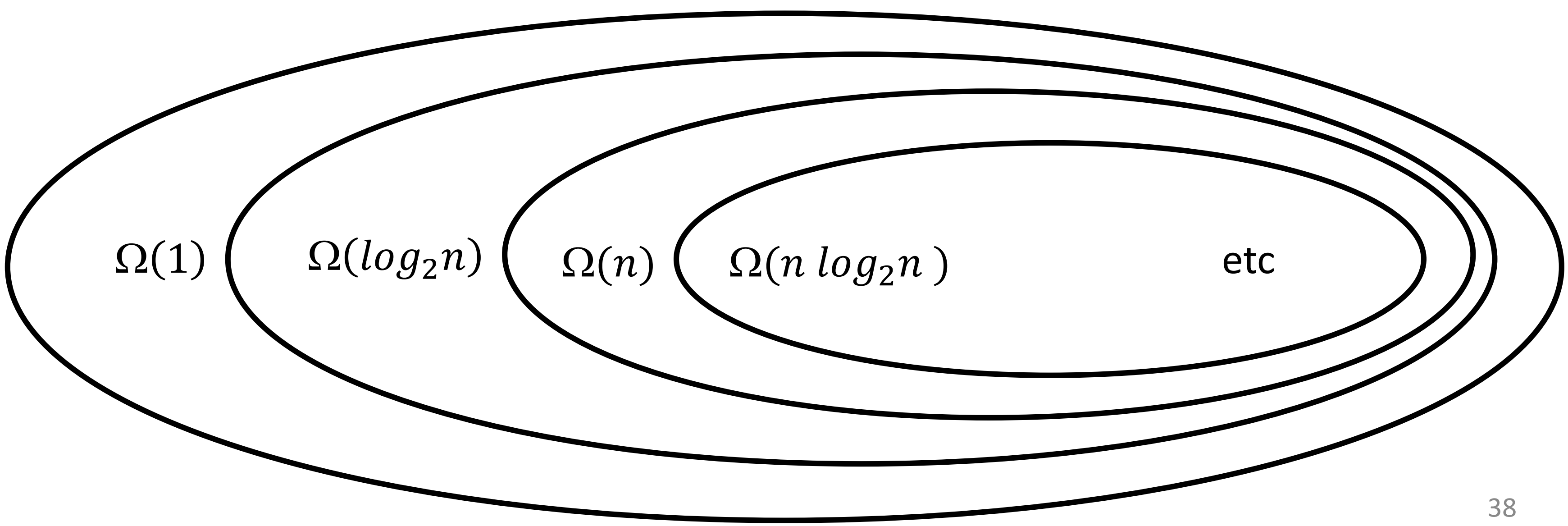
$$t(n) \in \Omega(\,g(n)\,).$$

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

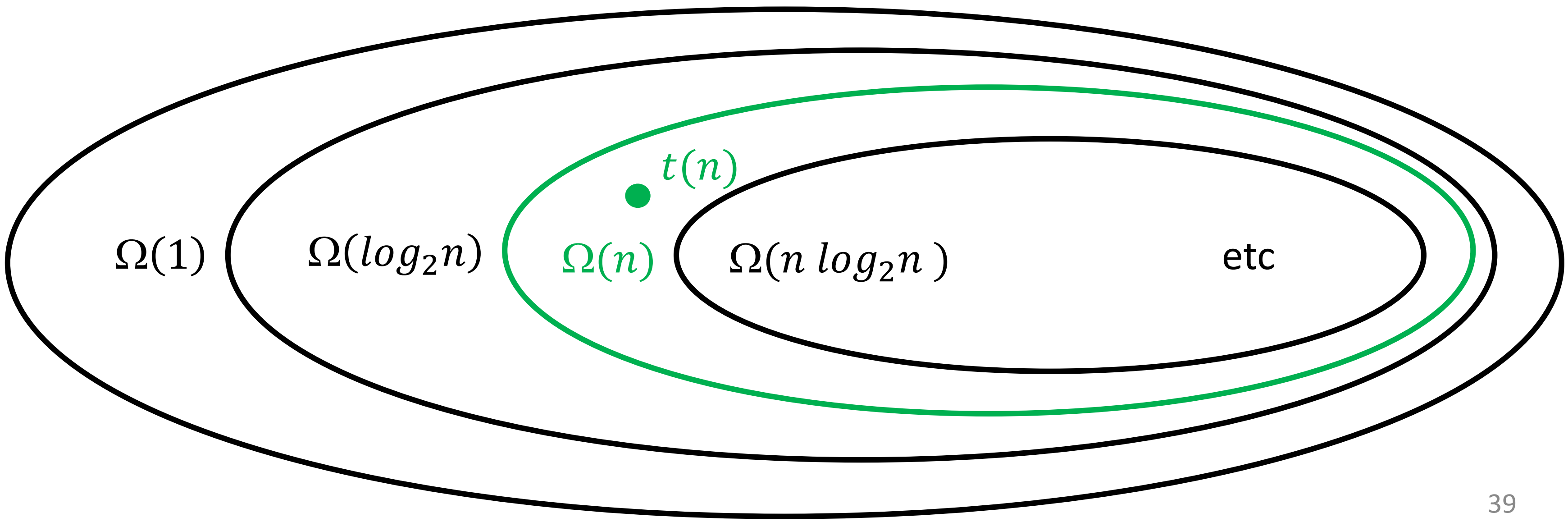
We have the following strict subset relationships:

$$\Omega(1) \supset \Omega(log_2 n) \supset \Omega(n) \supset \Omega(n\, log_2 n\,) \supset \Omega(n^2) \ \boldsymbol{...}$$

$$\supset \Omega(n^3) \supset \ \ldots \ \ \Omega(2^n\,) \supset \Omega(\,n!\,)$$

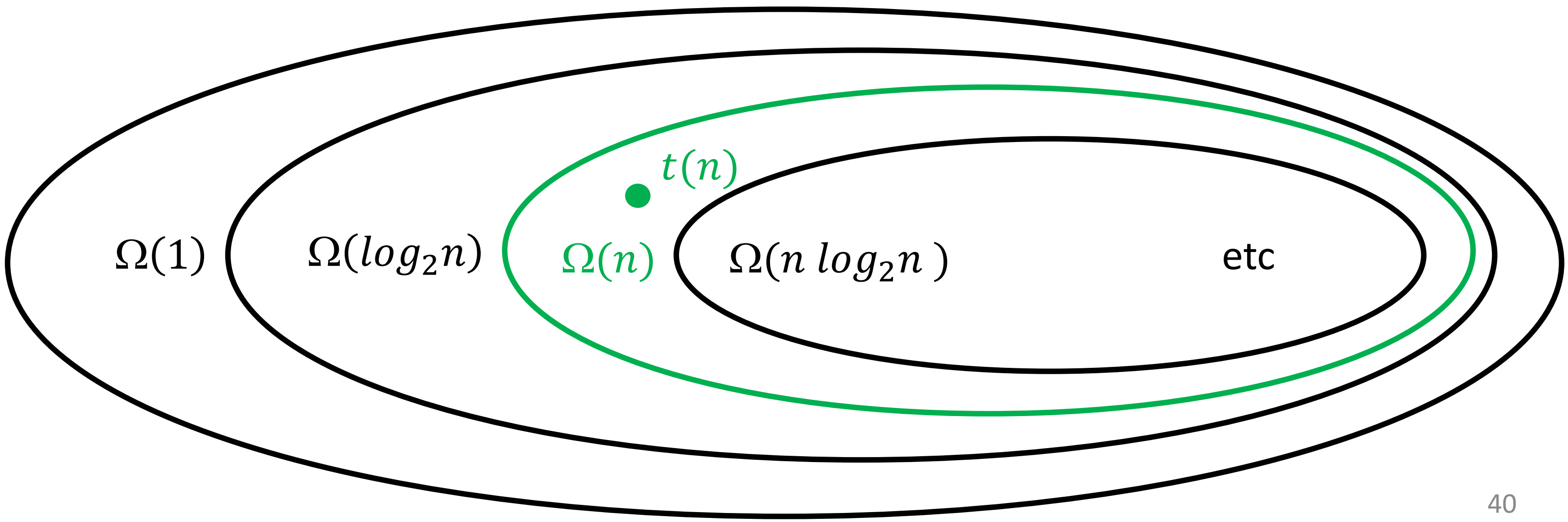


For example, if $t(n)$ belongs to $\Omega(n)$, then $t(n)$ also belongs to $\Omega(log_2 n)$ and to $\Omega(1)$.



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 also talk about **tight lower bounds**. For example, $\Omega(n)$ is a tight lower bound of $t(n)$ in the example below.



“small theta” θ

“big theta” Θ

Definition of Big Theta (Θ)

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

We say $t(n)$ is $\Theta(\, g(n)\,)$

if $t(n)$ is $\mathrm{O}(\, g(n)\,)$ and $t(n)$ is $\Omega(\, g(n)\,)$.

Definition of Big Theta (Θ)

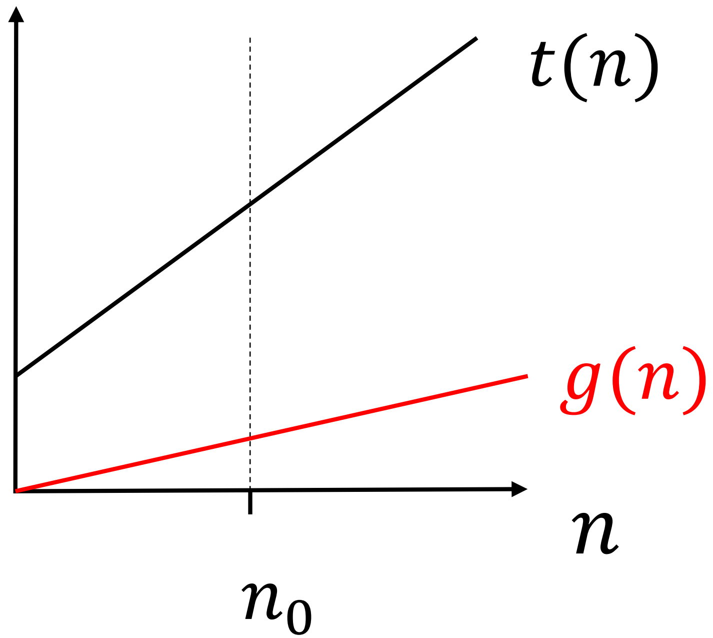
Let $t(n)$ and $g(n)$ be two functions of $n \geq 0$.

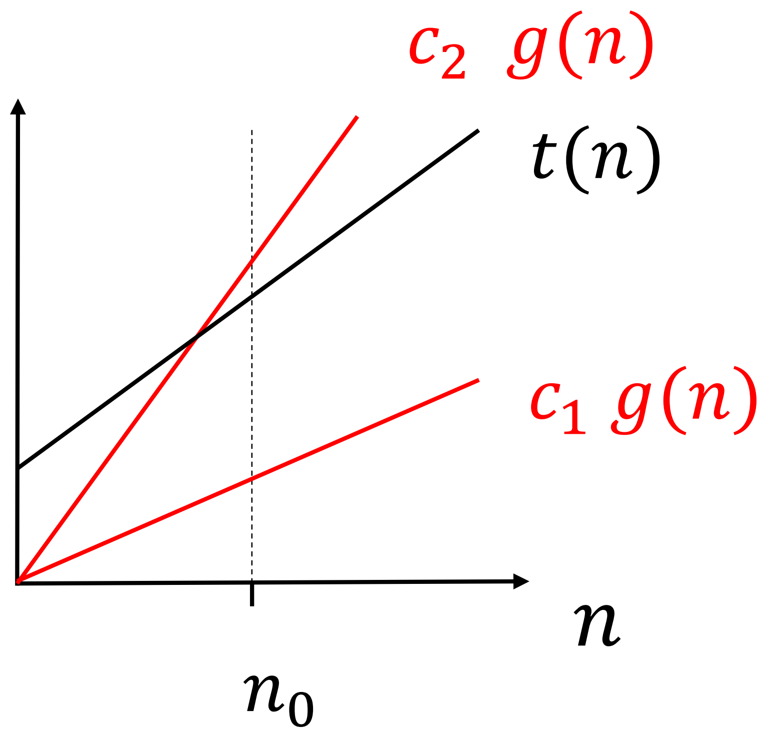
We say $t(n)$ is $\Theta(\, g(n) \,)$, if there exist three positive constants n_0, c_1, c_2 such that for all $n \geq n_0$,

$$c_1 \; g(n) \leq t(n) \leq c_2 \; g(n).$$

Note that we require $c_1 \leq c_2$.

Example





$t(n)$ is $\Theta\big(\, g(n)\big)$.

Definition of Big Theta (Θ)

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

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

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

$t(n)$ is O($g(n)$)

Definition of Big Theta (Θ)

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

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

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

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

Example

Let $t(n) = 4 + 17\, log_2\, n + 3n + 9\, n\, log_2\, n + \frac{n(n-1)}{2}$

Claim: $t(n)$ is $\Theta(\ n^2\ )$.

Proof:

Example

Let $t(n) = 4 + 17\,log_2\,n + 3n + 9\,n\,log_2\,n + \frac{n(n-1)}{2}$

Claim: $t(n)$ is $\Theta(\ n^2\ )$.

Proof:

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

A few more slides…

(time permitting or next lecture)

For every $t(n)$, does there exist a “simple” $g(n)$ such that $t(n)$ is $\Theta(\ )$?

For every $t(n)$, does there exist a “simple” $g(n)$ such that $t(n)$ is $\Theta(\;)$?

No, as this contrived example shows:

$$\text{Let } t(n) = \begin{cases} 5, & n \text{ is odd} \\ n^2, & n \text{ is even.} \end{cases}$$

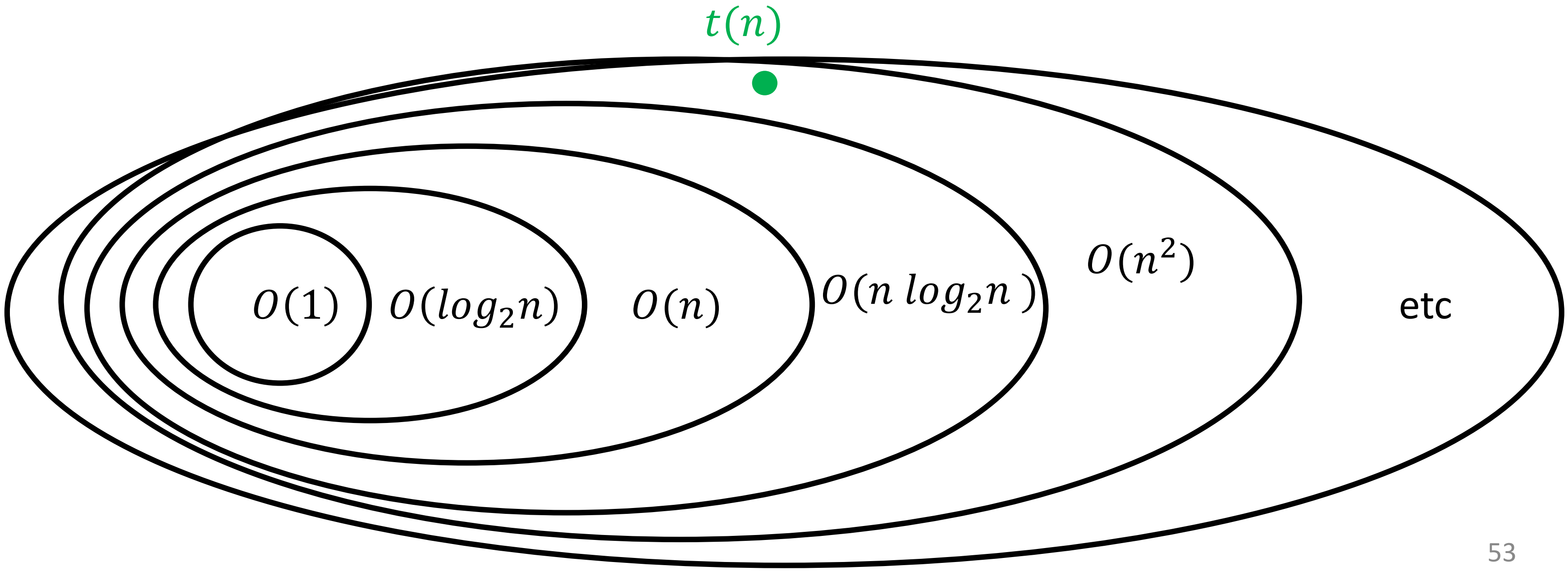
For every $t(n)$, does there exist a "simple" $g(n)$ such that $t(n)$ is $\Theta(\;)$?

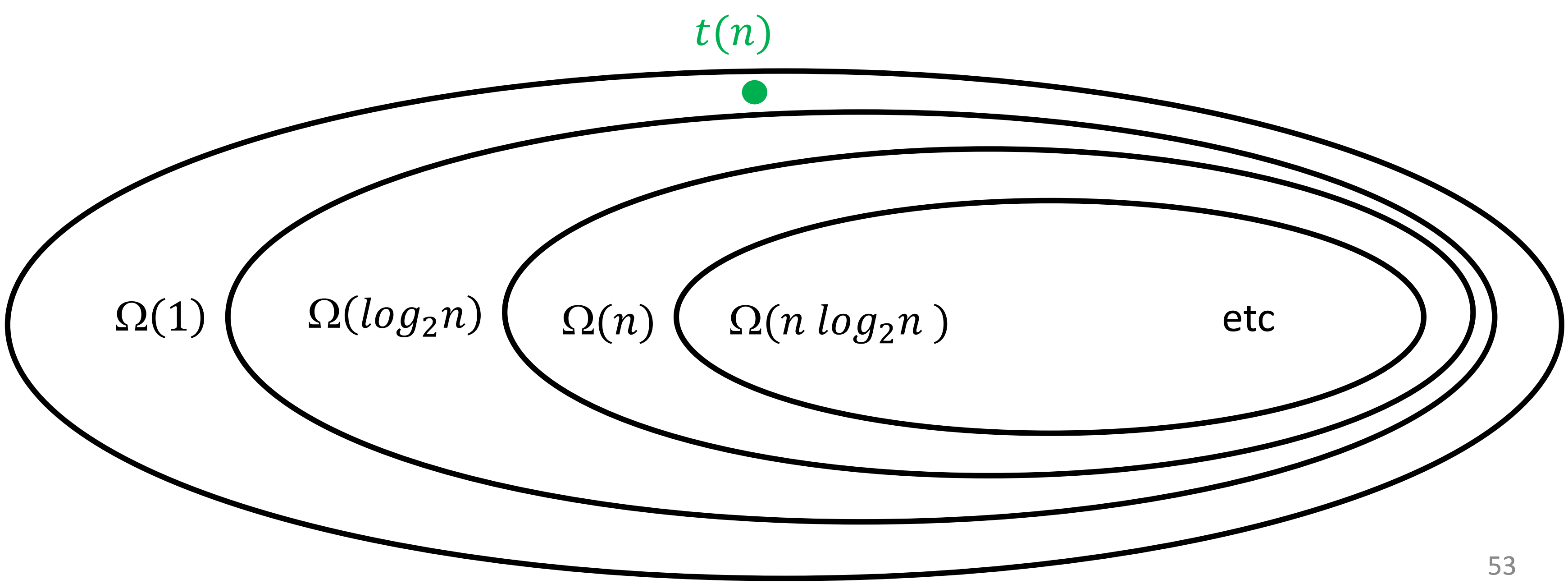
No, as this contrived example shows:

$$\text{Let } t(n) = \begin{cases} 5, & n \text{ is odd} \\ n^2, & n \text{ is even.} \end{cases}$$

$t(n)$ is $O(n^2)$, and this is a tight upper bound.

$t(n)$ is $\Omega(1)$, and this is a tight lower bound.


$t(n)$
$O(1)$
$O(log_2 n)$
$O(n)$
$O(n\ log_2 n\ )$
$O(n^2)$
etc


$t(n)$
$\Omega(1)$
$\Omega(log_2 n)$
$\Omega(n)$
$\Omega(n\ log_2 n\ )$
etc

Sets of Θ ( ) functions

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

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

$\Theta(1)$ $\quad \Theta(log_2 n\ )$ $\quad \Theta(n)$ $\quad \ldots \quad$ $\Theta(2^n\ )$ $\quad \Theta(n!\ )$

For most of this semester, we've been talking about big O. But really, what we had in mind was big Theta. That is, we had a function and we were discussing its growth rate.

The reason people usually talk about big O is that they are most concerned with upper bounds than they are about lower bounds.

Lower bounds do come up, but not as often.