Asymptotic Analysis

Main idea:

Focus on how the runtime **scales** with n (the input size).

Some examples…

(Heuristically: only pay attention to the largest function of n that appears.)

Number of operations	Asymptotic Running Time
$\frac{1}{10} \cdot n^2 + 100$	$O(n^2)$
$0.063 \cdot n^2 - .5\,n + 12.7$	$O(n^2)$
$100 \cdot n^{1.5} - 10^{10000}\sqrt{n}$	$O(n^{1.5})$
$11 \cdot n\log(n) + 1$	$O(n\log(n))$

Why is this a good idea?

- Suppose the running time of an algorithm is:

$$T(n) = 10n^2 + 3n + 7 \quad \text{ms}$$

This constant factor of 10 depends a lot on my computing platform…

These lower-order terms don't really matter as n gets large.

We're just left with the n^2 term! That's what's meaningful.

Pros and Cons of Asymptotic Analysis

Pros:

- Abstracts away from hardware- and language-specific issues.
- Makes algorithm analysis much more tractable.
- Allows us to meaningfully compare how algorithms will perform on large inputs.

Cons:

- Only makes sense if n is large (compared to the constant factors).

1000000000 n
is “better” than n^2 ?!?!

pronounced “big-oh of …” or sometimes “oh of …”

Informal definition for $O(g(n))$

- A function grows *no faster* than a certain rate

Formal definition for $O(g(n))$

- For a given function of n, $g(n)$
- $O(g(n))$ is the *set of functions* such that,

$O(g(n))$
$= \{$
$f(n)$: there exist positive
constants c and n_0 such that
$0 \leq f(n) \leq cg(n)$ for all n $\geq n_0$
$\}$

Formal definition for $O(g(n))$

- Let $T(n)$, $g(n)$ be functions of positive integers.
  - Think of $T(n)$ as a runtime: positive and increasing in n.

- We say "$T(n)$ is $O\big(g(n)\big)$" if:

for all large enough n,

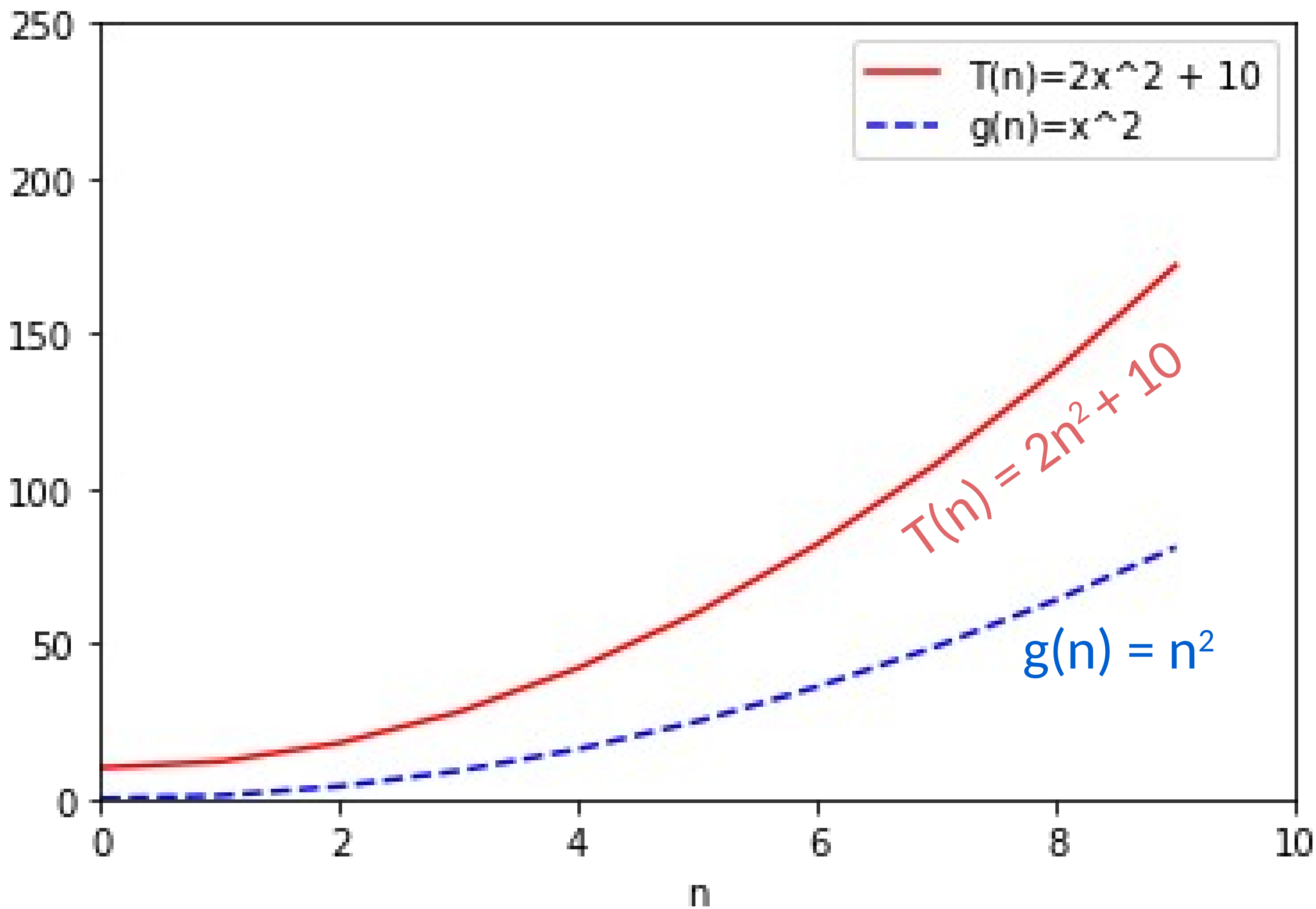
$T(n)$ is at most some constant multiple of $g(n)$.

Here, "constant" means "some number that doesn't depend on n."

Example

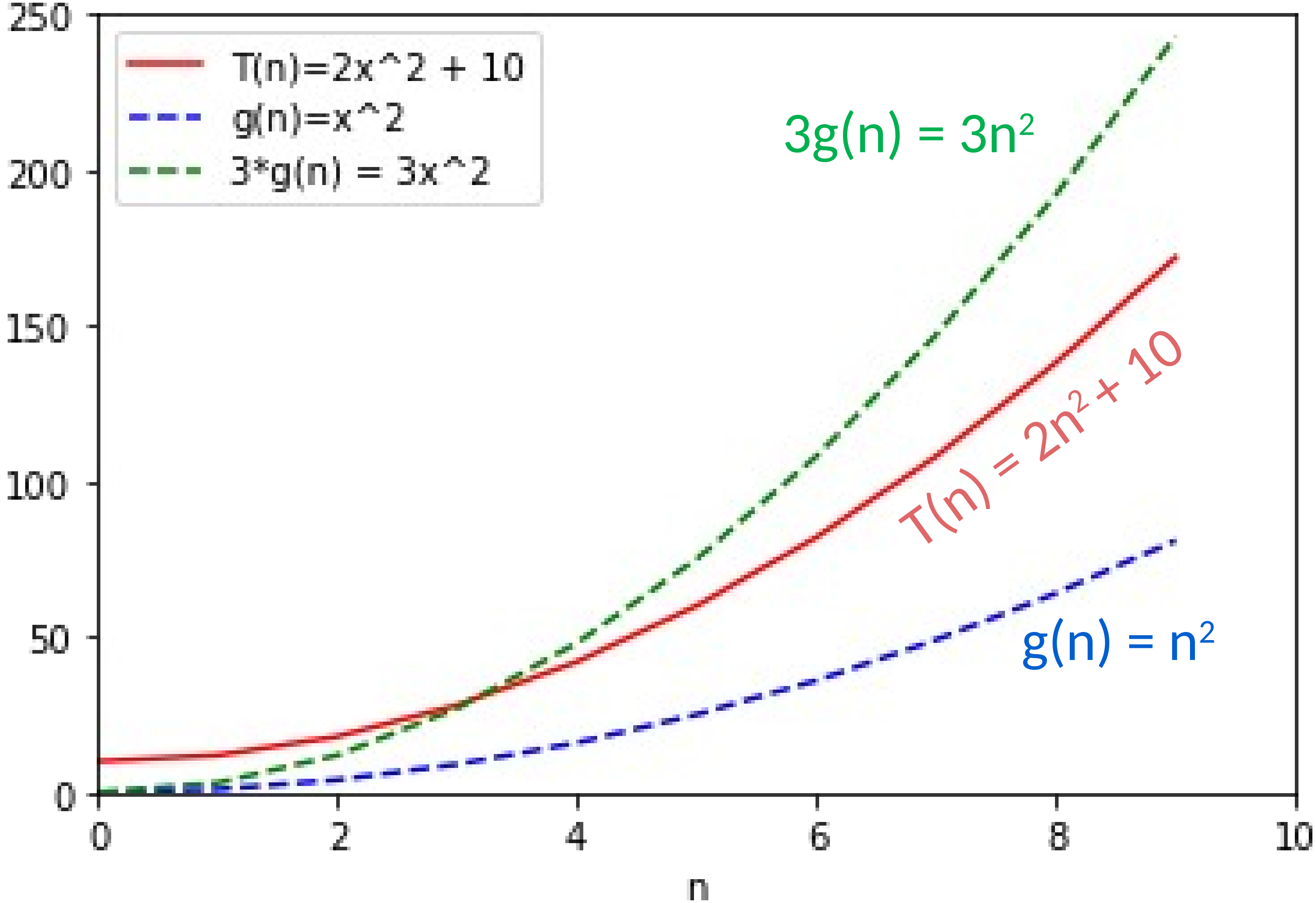
$2n^2 + 10 = O(n^2)$

for large enough n,
$T(n)$ is at most some constant
multiple of $g(n)$.



Example
$2n^2 + 10 = O(n^2)$

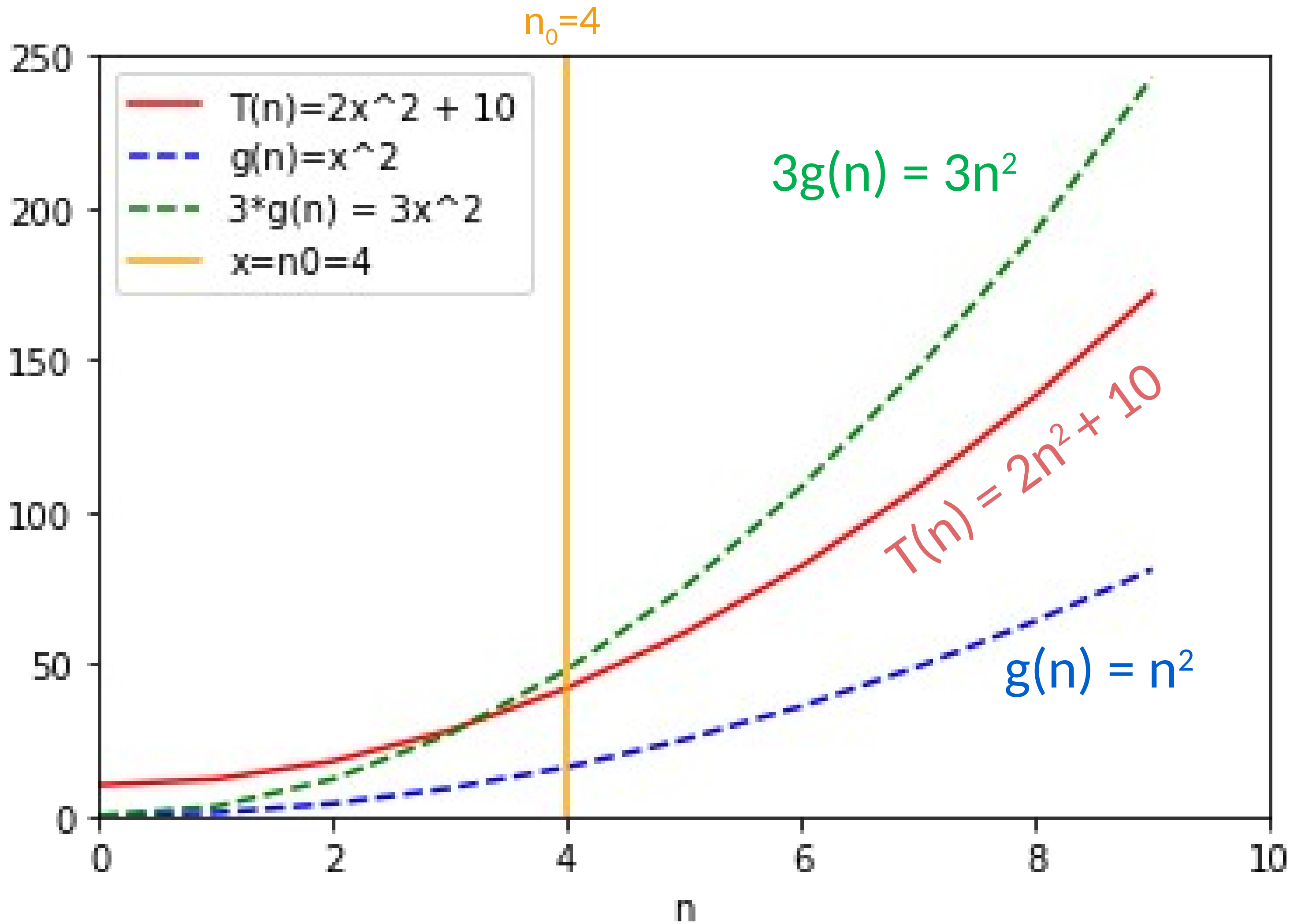
for large enough n,
$T(n)$ is at most some constant
multiple of $g(n)$.



Example

$2n^2 + 10 = O(n^2)$

for large enough n,
$T(n)$ is at most some constant multiple of $g(n)$.



Formal definition of $O(g(n))$

- Let $T(n)$, $g(n)$ be functions of positive integers.
  - Think of $T(n)$ as a runtime: positive and increasing in n.

- Formally,

$$T(n) = O(g(n))$$

“If and only if”

$$\Longleftrightarrow$$

“For all”

$$\exists c, n_0 > 0, s.t. \; \forall n \geq n_0,$$

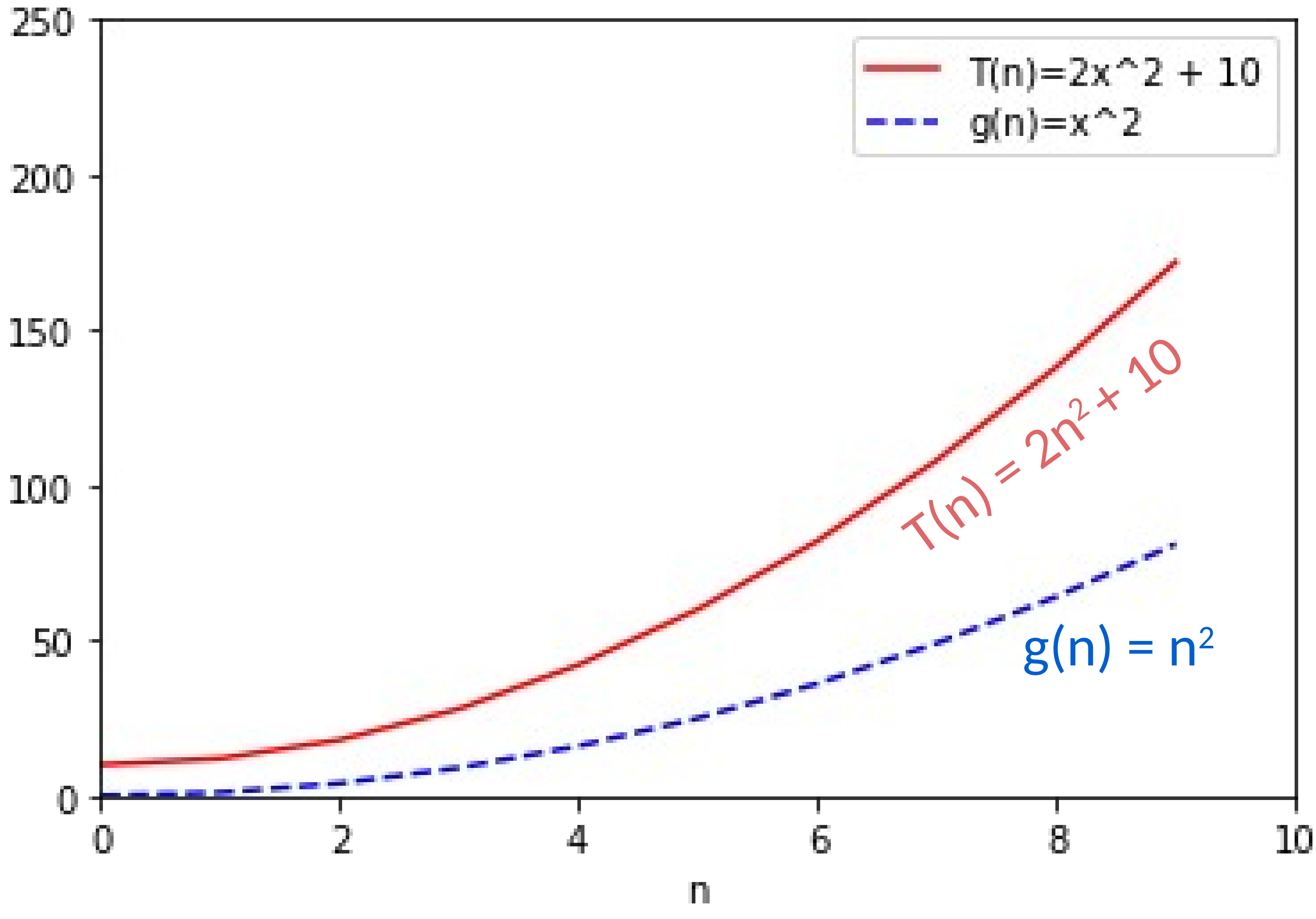
$$T(n) \leq c \cdot g(n)$$

“There exists”

“such that”

Example
$2n^2 + 10 = O(n^2)$

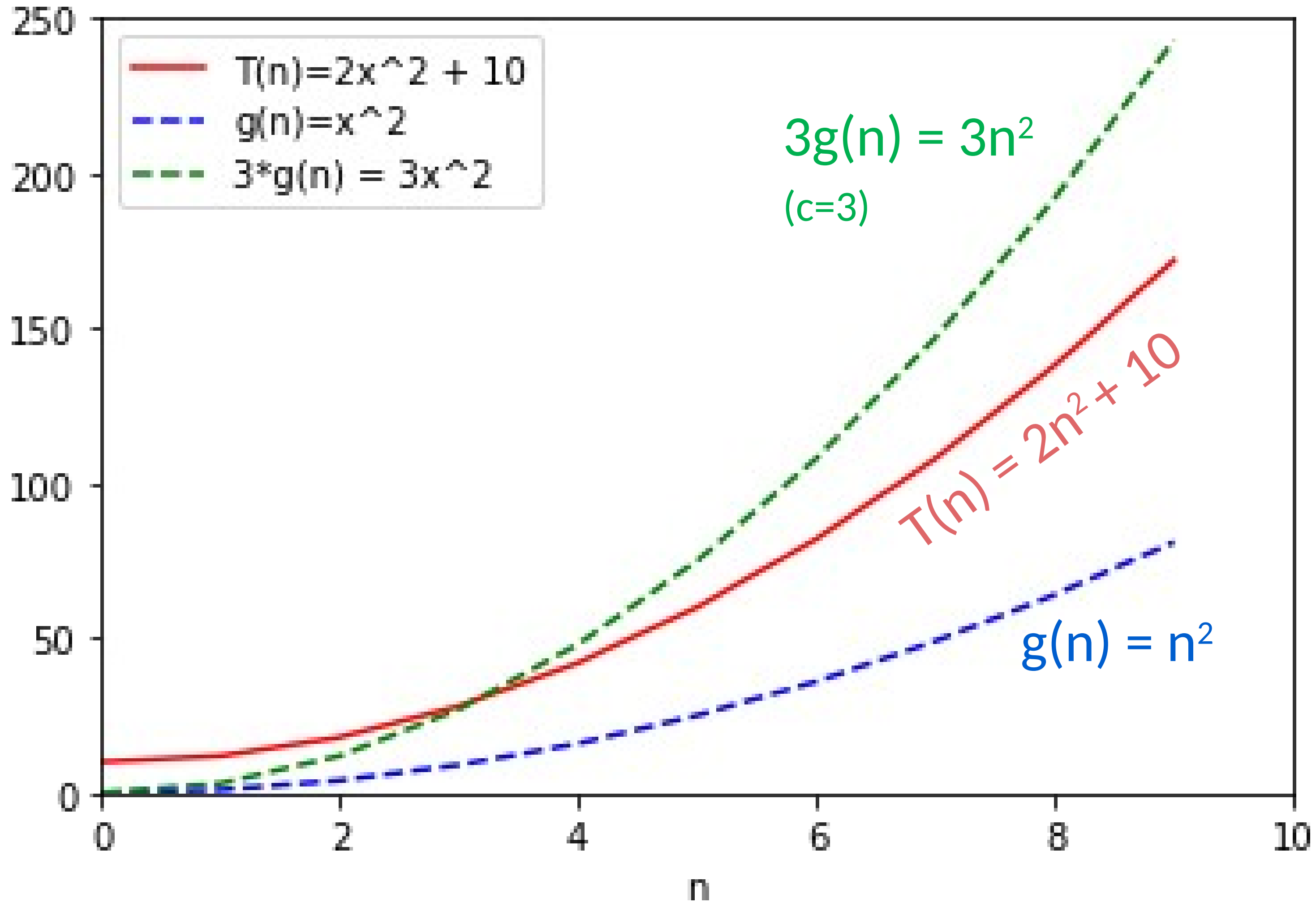
$$T(n) = O(g(n)) \Leftrightarrow \exists c, n_0 > 0 \ s.t. \ \forall n \ge n_0, T(n) \le c \cdot g(n)$$



Example

$2n^2 + 10 = O(n^2)$

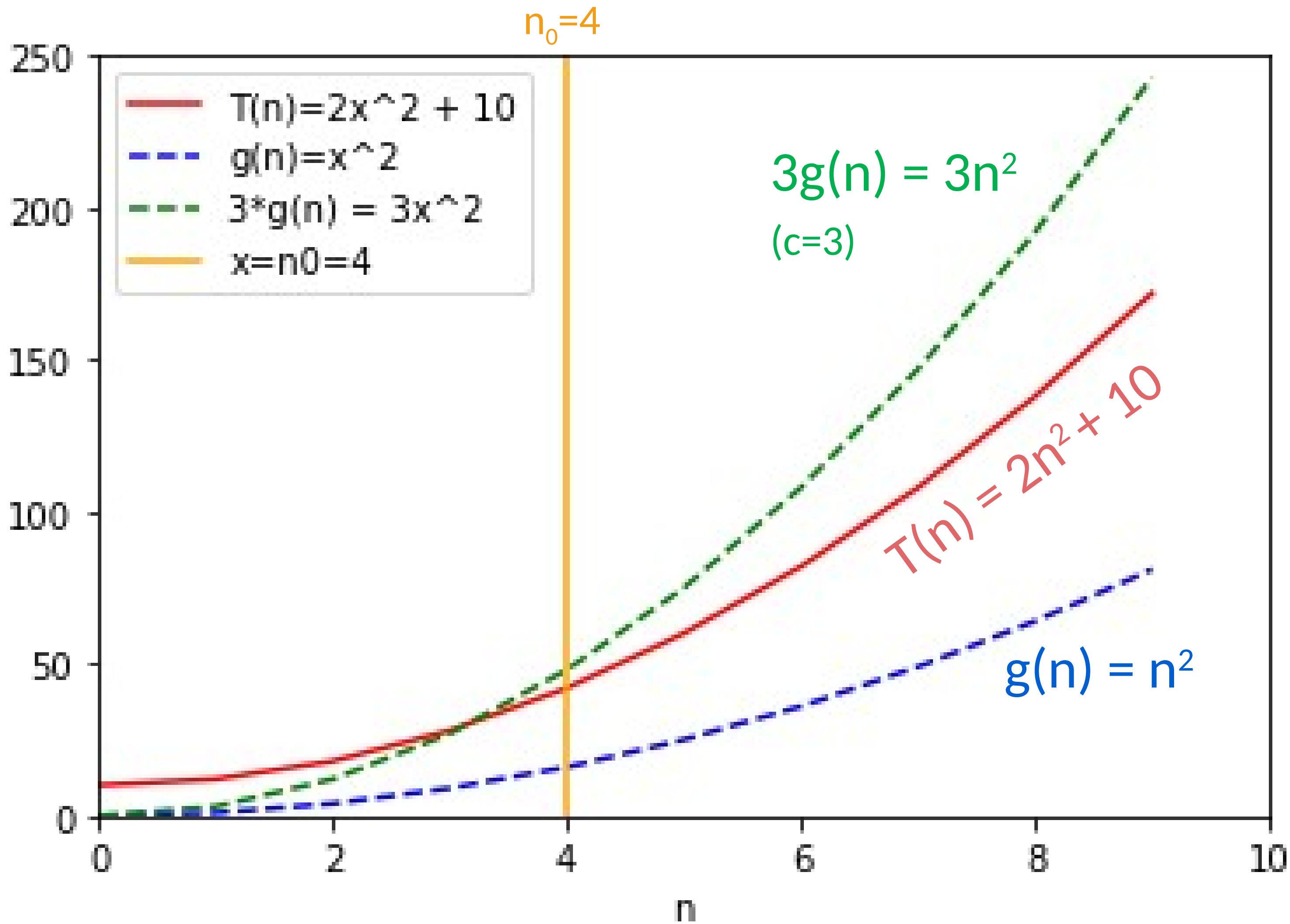
$$T(n) = O\big(g(n)\big)$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$



Example

$2n^2 + 10 = O(n^2)$

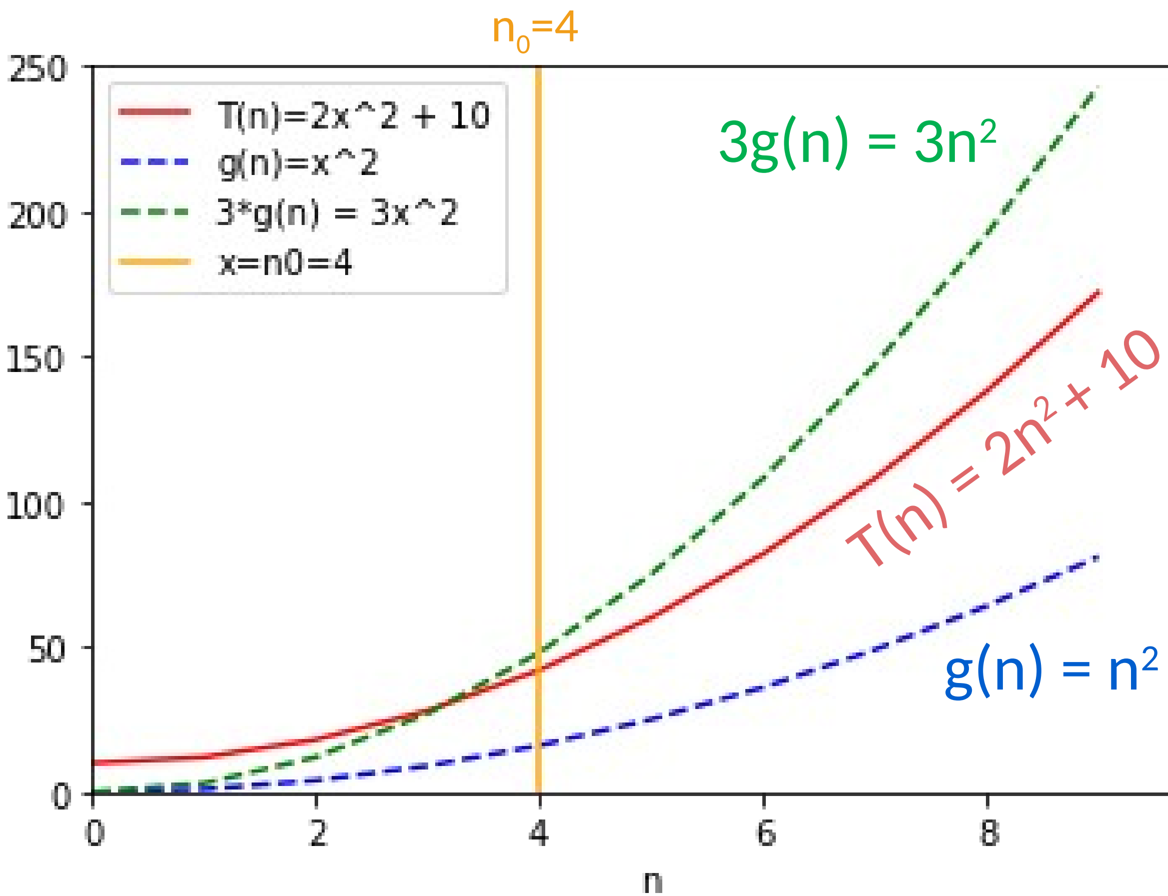
$$T(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$



Example

$2n^2 + 10 = O(n^2)$

$$T(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$



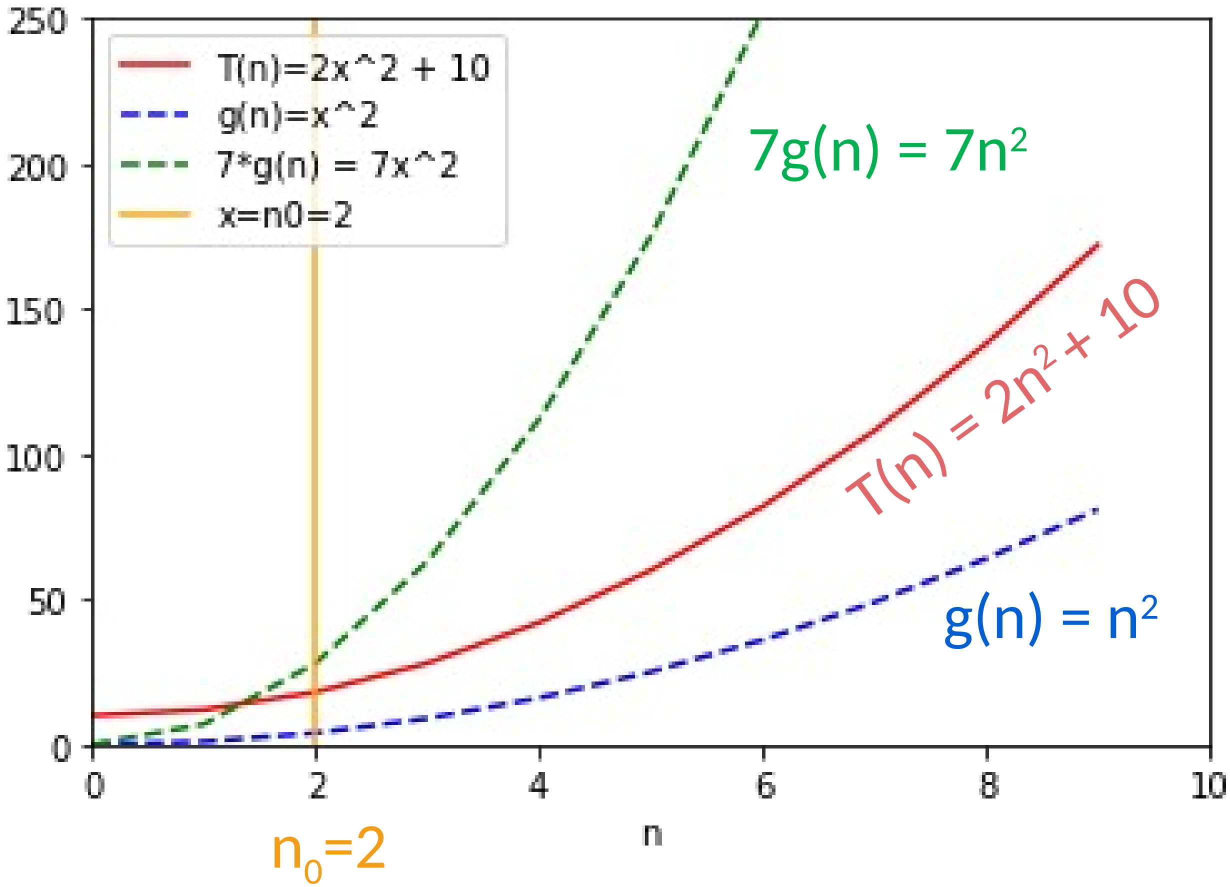
Formally:

- Choose c = 3
- Choose n_0 = 4
- Then:

$$\forall n \geq 4,$$
$$2n^2 + 10 \leq 3 \cdot n^2$$

Same example
$2n^2 + 10 = O(n^2)$

$$T(n) = O(g(n)) \Leftrightarrow \exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0, \; T(n) \leq c \cdot g(n)$$



Formally:

- Choose c = 7
- Choose n_0 = 2
- Then:

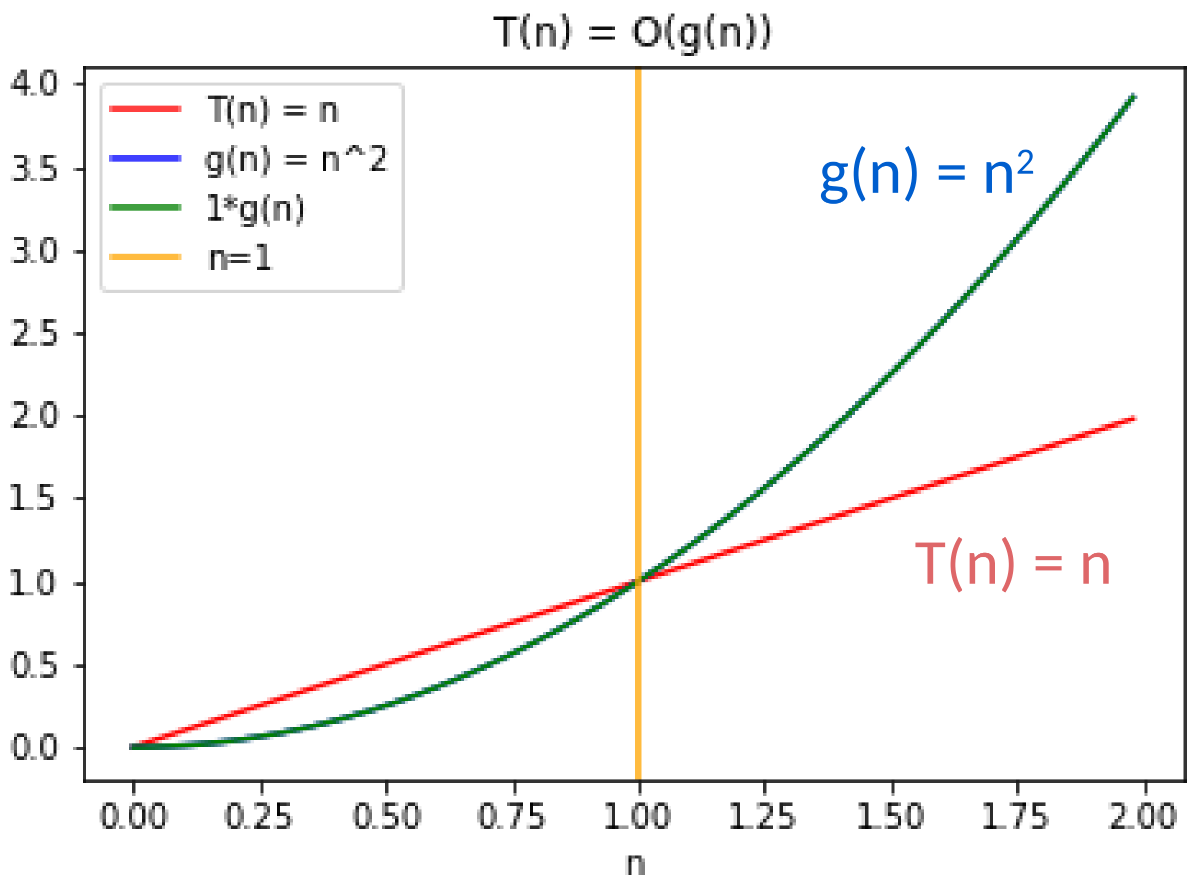
$$\forall n \geq 2,$$

$$2n^2 + 10 \leq 7 \cdot n^2$$

O(g(n)) is an upper bound

$$n = O(n^2)$$

$$T(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$



- Choose c = 1
- Choose n_0 = 1
- Then

$$\forall n \geq 1,$$
$$n \leq n^2$$

Informal definition for $\Omega(g(n))$

- A function grows *at least as fast as* a certain rate

Formal definition for $\Omega(g(n))$

- For a given function of n, `g(n)`
- $\Omega(g(n))$ is the *set of functions* such that,

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

f(n): there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all n $\geq n_0$

}

$\Omega(g(n))$ means a lower bound

- We say "$T(n)$ is $\Omega(g(n))$" if, for large enough n, $T(n)$ is at least as big as a constant multiple of $g(n)$.

- Formally,

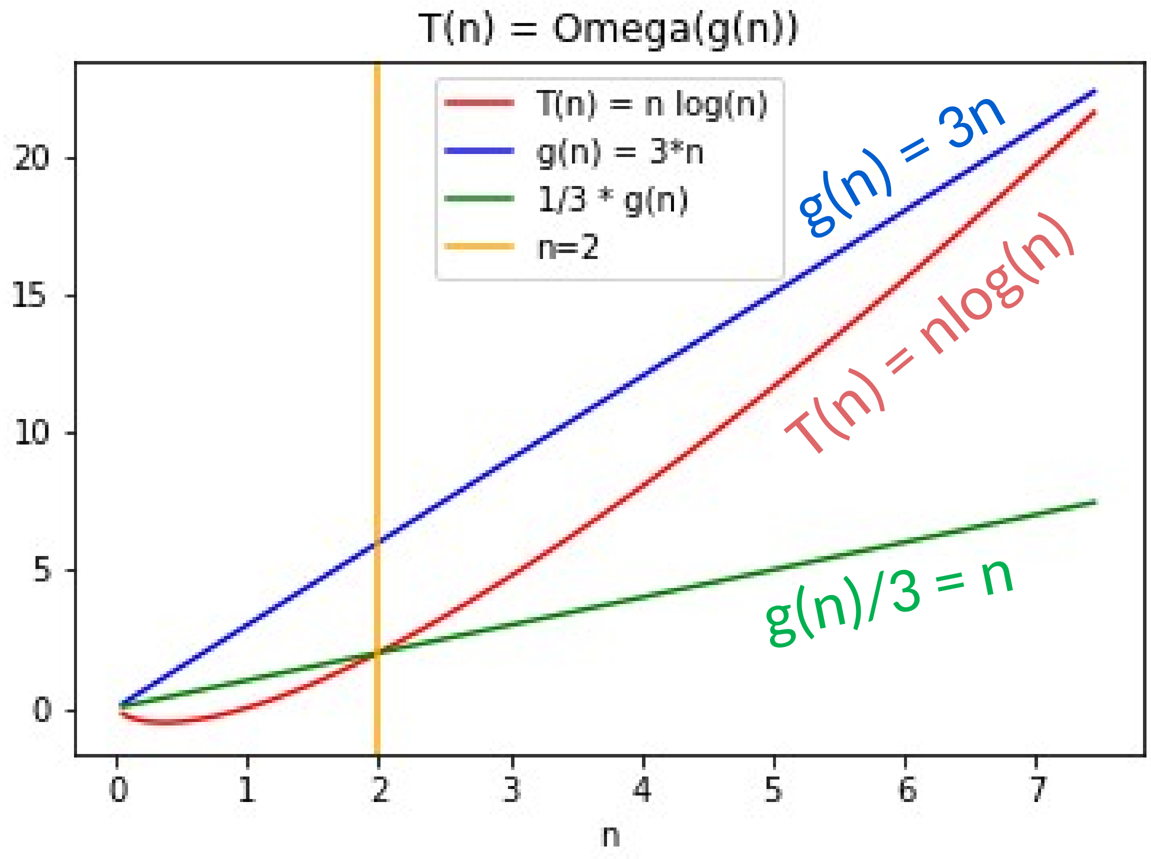
$$T(n) = \Omega(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \; s.t. \; \forall n \geq n_0,$$
$$c \cdot g(n) \leq T(n)$$

Switched these!!

Example

$n \log_2(n) = \Omega(3n)$

$$T(n) = \Omega(g(n)) \\ \Leftrightarrow \\ \exists c > 0, n_0 \;\; s.t. \;\; \forall n \geq n_0, \\ c \cdot g(n) \leq T(n)$$



- Choose c = 1/3
- Choose n_0 = 2
- Then

$$\forall n \geq 2,$$

$$\frac{3n}{3} \leq n \log_2(n)$$

Informal definition for $\Theta(g(n))$

- A function grows *precisely at* a certain rate

Θ(g(n)) means both!

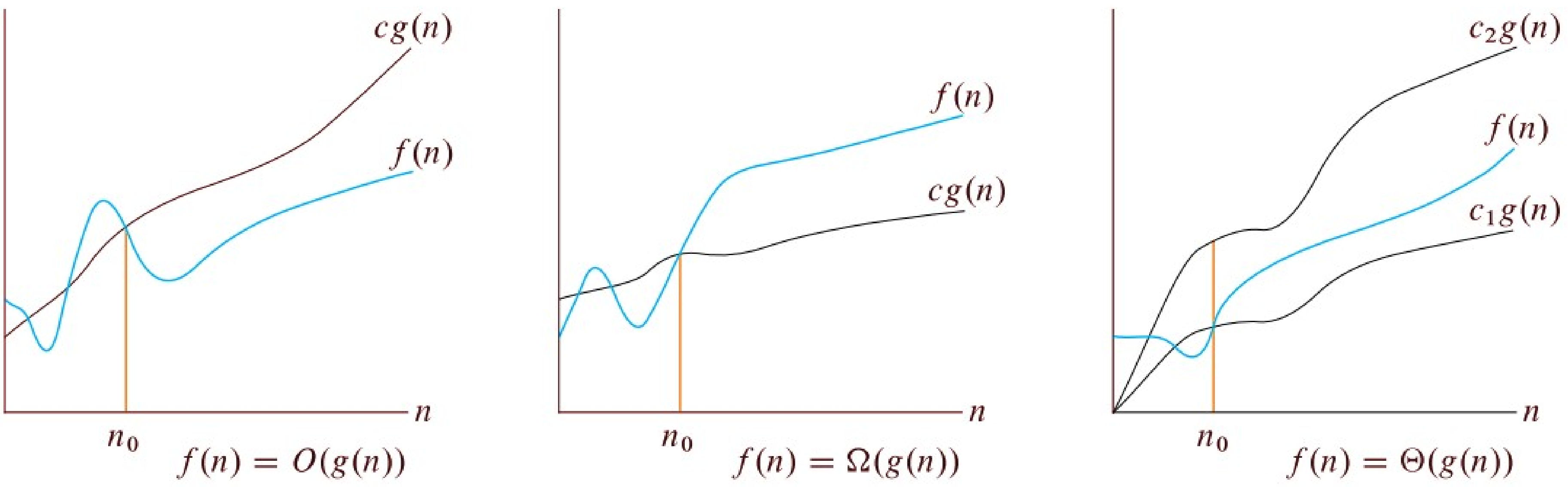
- We say "$T(n)$ is $\Theta(g(n))$" iff both:

$$T(n) = O\big(g(n)\big)$$

and

$$T(n) = \Omega\big(g(n)\big)$$

Summary of Asymptotic Notations



$$T(n) = O\big(g(n)\big)$$
$$\Leftrightarrow$$
$$\exists c > 0, n_0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$

Non-Example:
n^2 is not $O(n)$

- Proof by contradiction:
- Suppose that $n^2 = O(n)$.
- Then there is some positive c and n_0 so that:

$$\forall n \geq n_0, \qquad n^2 \leq c \cdot n$$

- Divide both sides by n:

$$\forall n \geq n_0, \qquad n \ \leq c$$

- That's not true!!! What about $\mathrm{n} = n_0 + c + 1$?
  - Then $n \geq n_0$, but $n \ > c$.
- Contradiction!

Take-away from examples

- To prove $T(n) = O(g(n))$, you have to come up with c and n_0 so that the definition is satisfied.

- To prove $T(n)$ is NOT $O(g(n))$, one way is **proof by contradiction**:
  - Suppose (to get a contradiction) that someone gives you a c and an n_0 so that the definition ***is*** satisfied.
  - Show that this someone must be lying to you by deriving a contradiction.

Formal definition of $o(g(n))$

- For a given function of n, $g(n)$
- o$(g(n))$ is the *set of functions* such that,

$\circ(g(n))$
= {

$f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) < cg(n)$ for all n $\geq n_0$

}

Formal definition of $\omega(g(n))$

- For a given function of n, $g(n)$
- $\omega(g(n))$ is the *set of functions* such that,

$$\omega(g(n)) = \{ f(n): \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) < f(n) \text{ for all n } \geq n_0 \}$$

Asymptotic Analysis

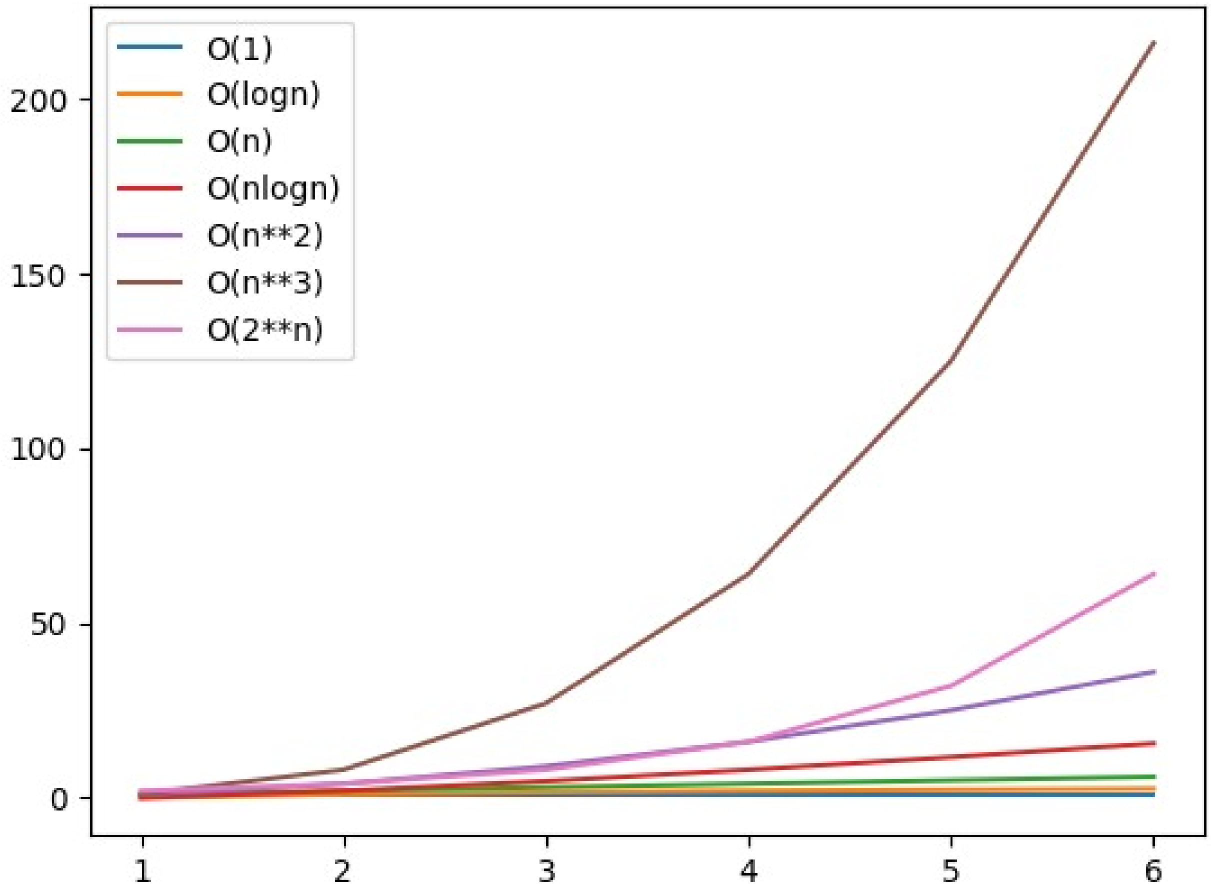
```
BUBBLE-SORT(A)
n = length[A]
for i = 1 to n - 1
    for j = i + 1 to n
        if A[j] < A[j - 1]
            exchange A[j] with A[j - 1]
```

`n−1` iterations on the outer loop

`n−i−1` iterations on the inner loop

Runtime $O(n^2)$

Common Bounds



Some Notations

- Asymptotic notations are defined as sets.
- But we use,

$$f(n) = O(g(n)) \text{ instead of } f(n) \in O(g(n))$$

- We can also write,

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$$

- Provide the simplest and most precise bounds possible

Reference

- CLRS Chapter 3