CS11313 - **Fall** 2023

Design & Analysis of Algorithms

The Big-*O* Notation and Its Relatives

Ibrahim Albluwi

Today's Agenda

- Running Time Orders of Growth.
- A formal definition of Big-O
- Big-O Relatives

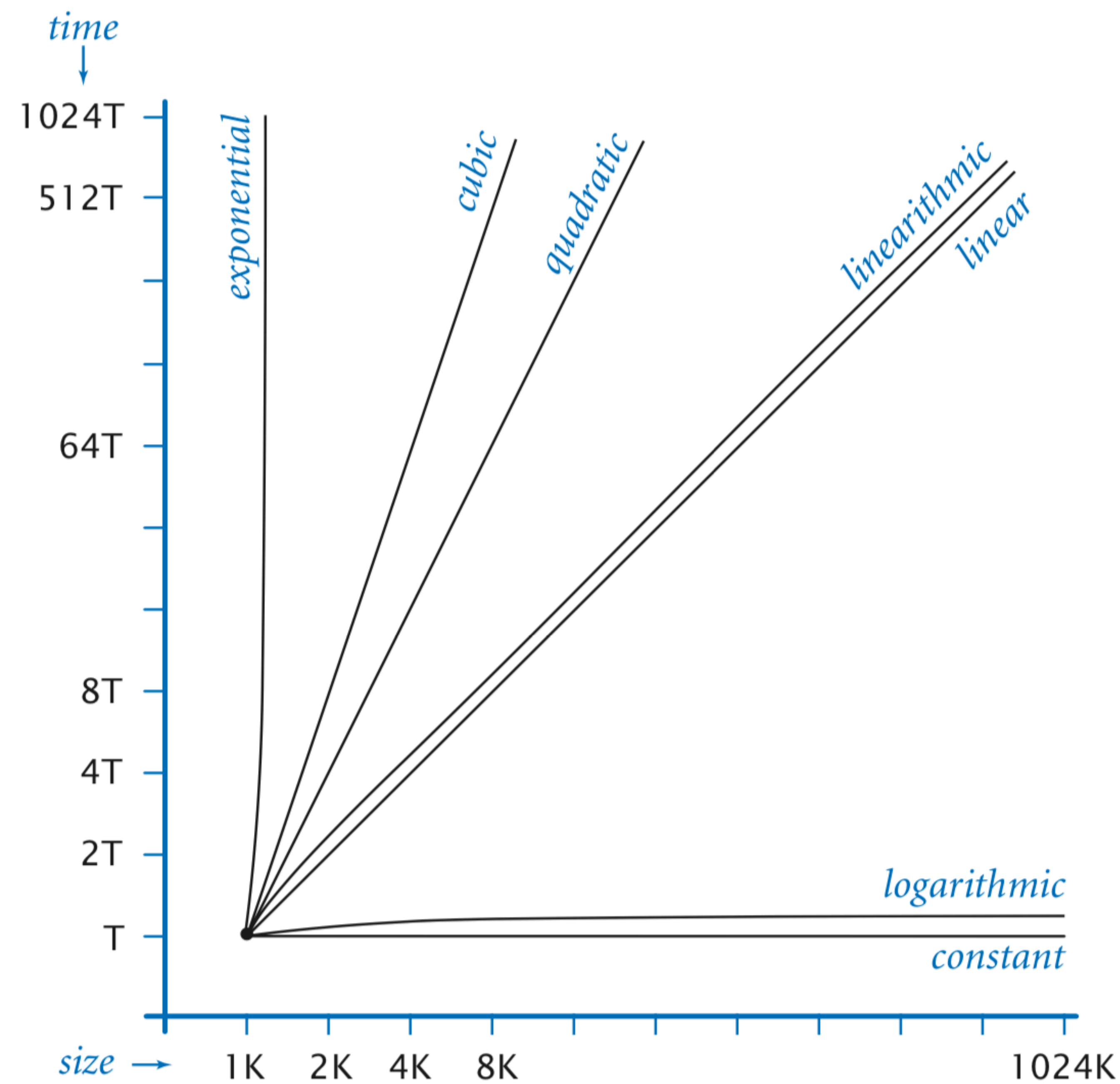
Orders of Growth (Review)

- Order of Growth of the running time: How quickly the running time of an algorithm grows as the input size grows.
  Examples: $\log n, \quad n, \quad n^2, \quad n^3, \quad 2^n,$ etc.

Examples of Growth Rates (Review)

graph by Kevin Wayne and Robert Sedgewick

	order of growth name	order of growth function
	constant	1
good	logarithmic	$\log(n)$
good		$\sqrt{n}$
fine	linear	n
fine	linearithmic	$n \log(n)$
fine		$n\sqrt{n}$
bad	quadratic	n^2
bad	cubic	n^3
horrible	exponential	2^n
horrible	exponential	3^n
horrible	factorial	$n!$



Orders of growth (log–log plot)

! constant < logarithmic < polynomial < exponential < factorial < n^n

$\log_b^a(n)$ $\quad$ $n^c\ (c > 0)$ $\quad$ $c^n\ (c > 1)$

Orders of Growth (Review)

- Order of Growth of the running time: How quickly the running time of an algorithm grows as the input size grows.
  Examples: $\log n, \quad n, \quad n^2, \quad n^3, \quad 2^n,$ etc.

- Focus on the highest order term:
  - Example: $n^2 + n + \log n$ is in the order of n^2.
  - Rationale: When n becomes large, time due to the lower order terms becomes insignificant compared to the highest order term.

- Drop the coefficient of the highest order term:
  - Example: $n^2, \quad \frac{1}{2}n^2$ and $10n^2$ are all in the order of n^2.
  - Rationale:
    - Quadratic growth is not the same as, linear or cubic growth, etc.
    - Algorithms have different constants when implemented, based on hardware, software and implementation factors.

Quiz # 1

Assume $T(n)$ is the order of growth of the running time of Bubble Sort as a function of the input size n. Which of the following is *true* about $T(n)$?

- **A.** $T(n) = O(n^2)$
- **B.** $T(n) = O(n^3)$
- **C.** $T(n) = O(n^4)$
- **D.** All of the above.
- **E.** None of the above.

What is

Big-*O* anyway?

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute

Help
Learn to edit
Community portal
Recent changes
Upload file

Big O notation

From Wikipedia, the free encyclopedia

Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. Big O is a member of a family of notations invented by Paul Bachmann,[1] Edmund Landau,[2] and others, collectively called **Bachmann–Landau notation** or **asymptotic notation**.

In computer science, big O notation is used to classify algorithms according to how their run time or space requirements grow as the input size grows.[3] In analytic number theory, big O notation is often used to

$$O(), \sim$$

Fit approximation

Concepts

Orders of approximation
Scale analysis · **Big O notation**
Curve fitting · False precision
Significant figures

Other fundamentals

Approximation · Generalization error
Taylor polynomial
Scientific modelling

V · T · E

Big-O

Definition. Let $f(n)$ and $g(n)$ be two functions that are always positive, $f(n)$ is said to be $O(g)$ if and only if :

> There are two positive constants c and n_o , such that
> $0 \leq f(n) \leq c \bullet g(n)$ for all $n \geq n_o$

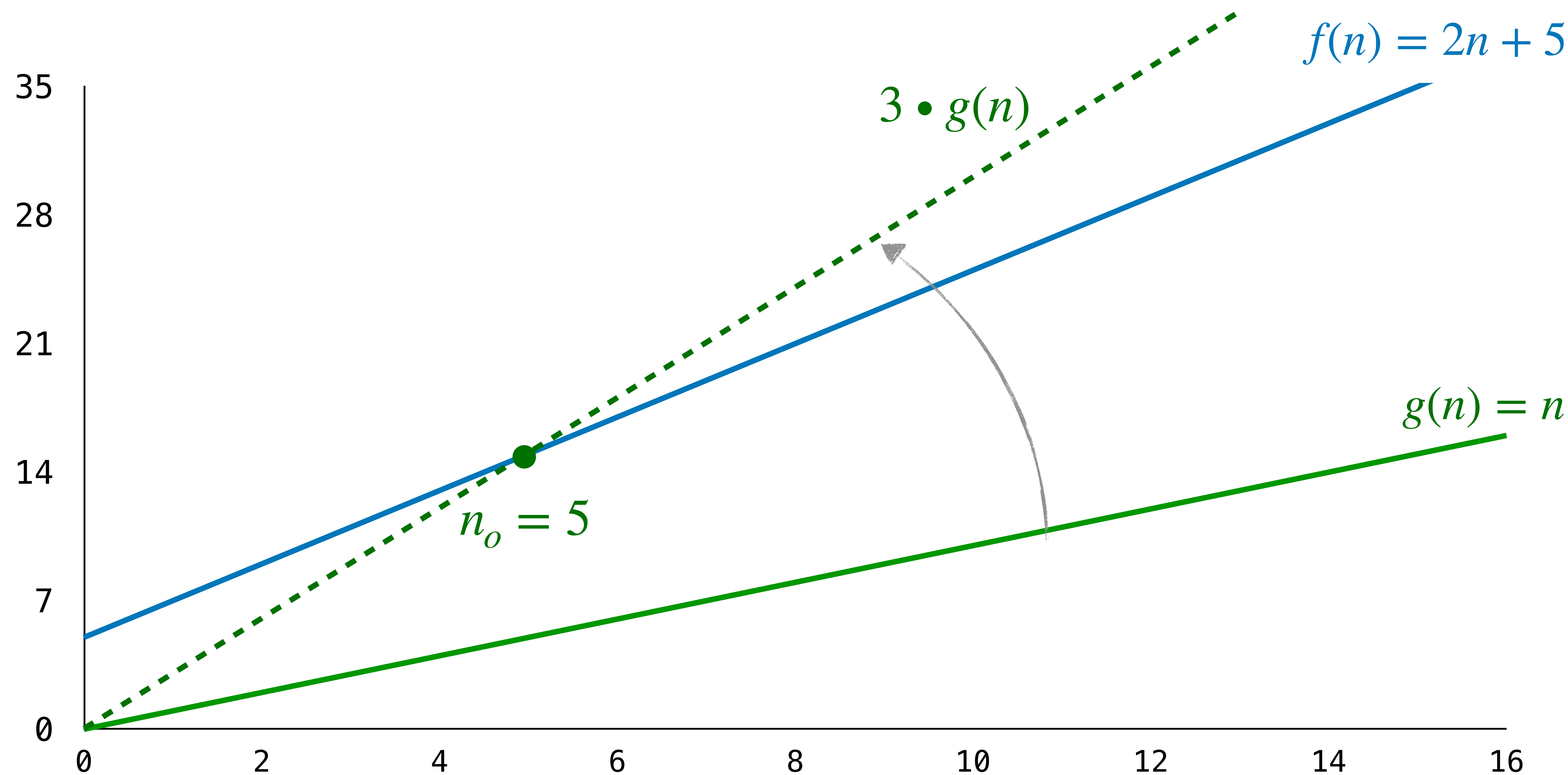
Less formally: If multiplying $g(n)$ by a constant makes it an upper bound for $f(n)$ after some point, then f is $O(g)$.

Example # 1

Assume $f(n) = 2n + 5$ and $g(n) = n$.

f is $O(g)$ because there are c and n_o such that $0 \leq f(n) \leq c \bullet g(n)$ for all $n \geq n_o$:

If $c = 3$, then $0 \leq f(n) \leq 3 \bullet g(n)$ for all $n \geq 5$

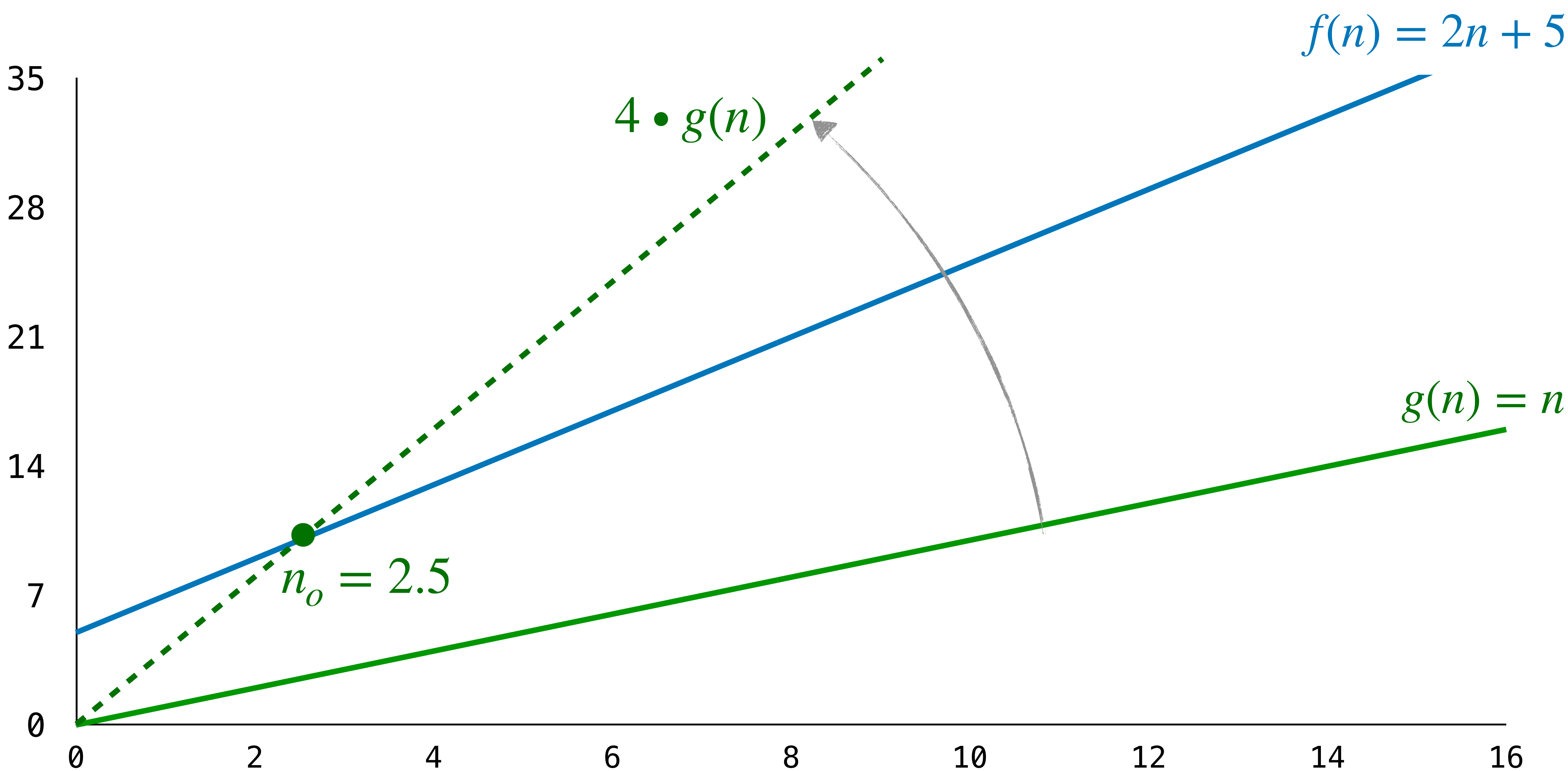


Example # 1

Assume $f(n) = 2n + 5$ and $g(n) = n$.

f is $O(g)$ because there are c and n_o such that $0 \le f(n) \le c \bullet g(n)$ for all $n \ge n_o$:

If $c = 4$, then $0 \le f(n) \le 4 \bullet g(n)$ for all $n \ge 2.5$

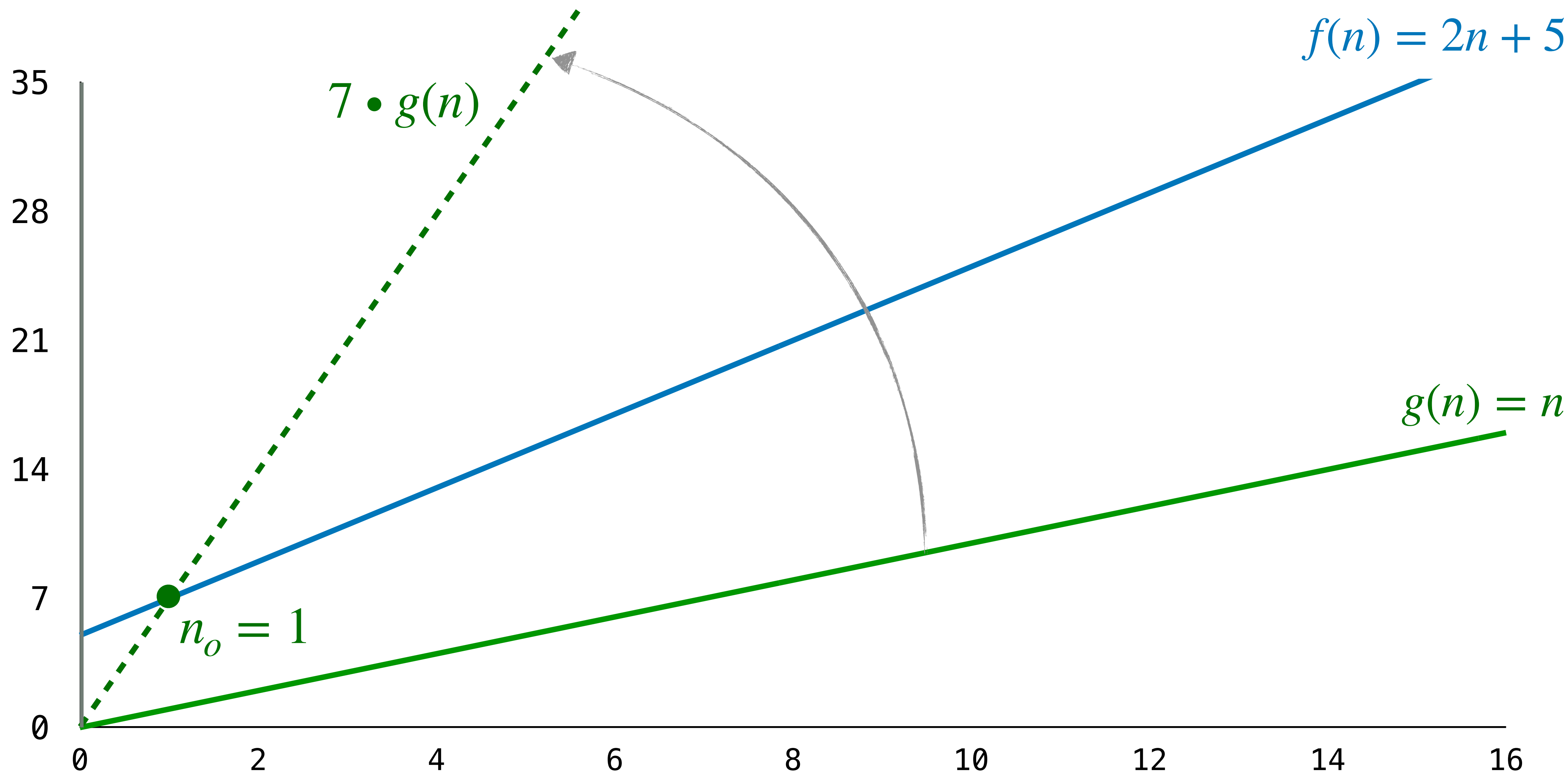


Example # 1

Assume $f(n) = 2n + 5$ and $g(n) = n$.

f is $O(g)$ because there are c and n_o such that $0 \leq f(n) \leq c \bullet g(n)$ for all $n \geq n_o$:

If $c = 7$, then $0 \leq f(n) \leq 7 \bullet g(n)$ for all $n \geq 1$

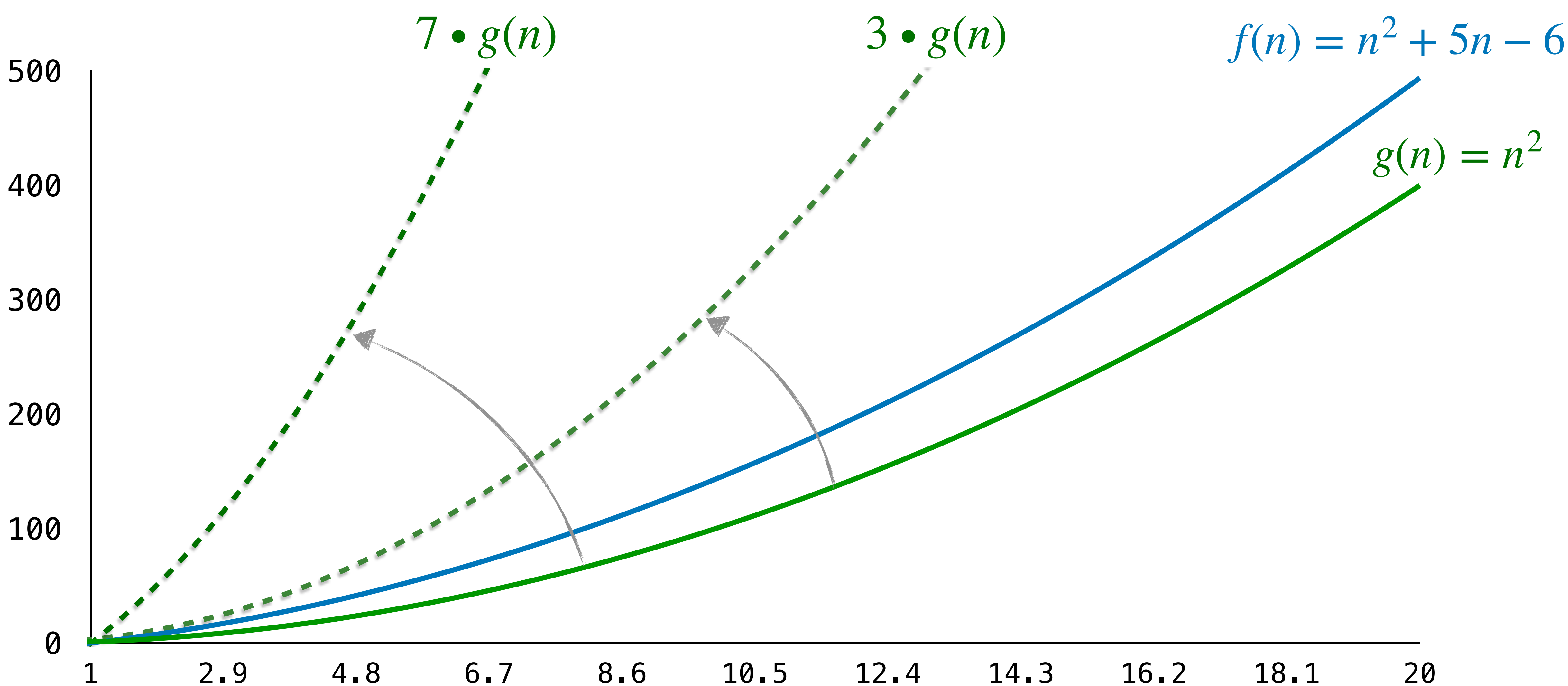


Example # 2

Assume $f(n) = n^2 + 5n - 6$ and $g(n) = n^2$.

f is $O(g)$ because there are c and n_o such that $0 \leq f(n) \leq c \bullet g(n)$ for all $n \geq n_o$:

If $c = 7$, then $0 \leq f(n) \leq 7 \bullet g(n)$ for all $n \geq 1$



Exercise # 1

For each of the following functions, show that f is $O(g)$.

A. $f(n) = 3n + 3$ and $g(n) = n$

Solution Attempt # 1.

Let's (arbitrarily) pick $c = 1$:

We need to show that there are two constants, c and n_0 such that:

$$0 \leq f(n) \leq c \bullet g(n) \text{ for all } n \geq n_0.$$

I.e. we need to show that: $3n + 3 \leq n$ for all $n \geq n_0$.

Solving for n:

$$3 \leq n - 3n$$

$$3 \leq -2n$$

$$-\frac{3}{2} \geq n$$

IMPOSSIBLE

This choice of c has no corresponding n_0 that makes the inequality always true!

Exercise # 1

For each of the following functions, show that f is $O(g)$.

A. $f(n) = 3n + 3$ and $g(n) = n$

Solution Attempt # 2.

Let's (arbitrarily) pick $c = 9$:

We need to show that there are two constants, c and n_0 such that:

$$0 \leq f(n) \leq c \bullet g(n) \text{ for all } n \geq n_0.$$

I.e. we need to show that: $3n + 3 \leq 9n$ for all $n \geq n_0$.

Solving for n:

$$3 \leq 9n - 3n$$

$$3 \leq 6n$$

$$\frac{3}{6} \leq n$$

In other words: $3n + 3 \leq 9n$ for all $n \geq \frac{3}{6}$

😁 $\boxed{c = 9 \text{ and } n_0 = \frac{3}{6}}$

🤔 How do we find a c that works?

A. Look for hints in $f(n)$ and $g(n)$ (or try a large number!)

Exercise # 1

For each of the following functions, show that f is $O(g)$.

A. $f(n) = 3n + 3$ and $g(n) = n$

Solution # 3

We need to show that there exist two constants c and n_o such that

$$0 \leq 3n + 3 \leq c \bullet n \qquad \text{for all } n \geq n_o.$$

Since $\quad 0 \leq 3n + 3 \leq 3n + 3n \qquad \text{for all } n \geq 1$

$\quad\quad 0 \leq 3n + 3 \leq 6n \qquad \text{for all } n \geq 1$

We can pick $c = 6$ and $n_o = 1$

Exercise # 1

For each of the following functions, show that f is $O(g)$.

A. $f(n) = 3n + 3$ and $g(n) = n$

Solution.

If we pick $c = 6$, we can show that $0 \leq f(n) \leq c \bullet g(n)$ for all $n \geq 1$.

$$0 \leq 3n + 3 \leq 3n + 3n \leq 6n \quad \text{for all } n \geq 1.$$

B. $f(n) = n^2 + 5n + 6$ and $g(n) = n^2$

Solution.

If we pick $c = 12$, we can show that $0 \leq n^2 + 5n + 6 \leq 12n^2$ for all $n \geq 1$.

$$0 \leq \underbrace{n^2 + 5n + 6}_{f(n)} \leq n^2 + 5n^2 + 6n^2 \leq \underbrace{12n^2}_{c \bullet g(n)} \quad \text{for all } n \geq 1.$$

Exercise # 1

For each of the following functions, show that f is $O(g)$.

A. $f(n) = 3n + 3$ and $g(n) = n$

Solution.

If we pick $c = 6$, we can show that $0 \leq f(n) \leq c \bullet g(n)$ for all $n \geq 1$.

$0 \leq 3n + 3 \leq 3n + 3n \leq 6n$ for all $n \geq 1$.

B. $f(n) = n^2 + 5n + 6$ and $g(n) = n^2$

Solution.

If we pick $c = 12$, we can show that $0 \leq n^2 + 5n + 6 \leq 12n^2$ for all $n \geq 1$.

$0 \leq n^2 + 5n + 6 \leq n^2 + 5n^2 + 6n^2 \leq 12n^2$ for all $n \geq 1$.

C. $f(n) = n^2$ and $g(n) = n^3$

Solution.

If we pick $c = 1$, It is clear that $0 \leq f(n) \leq c \bullet g(n)$ for all $n \geq 1$.

Dividing $0 \leq n^2 \leq n^3$ by n^2 makes the equation: $0 \leq 1 \leq n$

Back to Quiz # 1

Assume $T(n)$ is the order of growth of the running time of Bubble Sort as a function of the input size n. Which of the following is *true* about $T(n)$?

A. $T(n) = O(n^2)$

B. $T(n) = O(n^3)$

C. $T(n) = O(n^4)$

$\boxed{\text{D. All of the above.}}$

E. None of the above.

$$
\begin{aligned}
T(n) = \tfrac{1}{2}n^2 - \tfrac{1}{2}n \quad &\leq c \bullet n^2 \\
&\leq c \bullet n^3 \\
&\leq c \bullet n^4 \qquad \text{for all } n \geq 1, \text{ assuming } c = 1
\end{aligned}
$$

Quiz # 2

Assume $T(n)$ is the order of growth of the running time of Selection Sort as a function of the input size n. Which of the following best describes $T(n)$?

A. $T(n) = O(n^2)$

B. $T(n) = O(n^6)$

C. $T(n) = O(n^n)$

D. All of the above.

E. None of the above.

Exercise # 1

For each of the following functions, show that f is $O(g)$.

D. $f(n) = 2^n$ and $g(n) = 3^n$

Solution.

We need to show that: $0 \leq 2^n \leq c \bullet 3^n$ for all $n \geq n_o$.

Divide by 2^n: $0 \leq 1 \leq c \bullet (\frac{3}{2})^n$ for all $n \geq n_o$.

We can pick $c = 1$ which makes the statement true for all $n \geq 1$.

!

Note that we don't always need to explicitly find c and n_o.
It is enough to show that they exist. For example, a valid answer for the above example would be:

Since 1 is constant and $(\frac{3}{2})^n$ is a strictly increasing function, there must be some c and $n_o \geq 1$ such that $0 \leq 1 \leq c \bullet (\frac{3}{2})^n$ for all $n \geq n_o$.

Exercise # 1

For each of the following functions, show that f is $O(g)$.

E. $f(n) = An + B$ and $g(n) = n$ where A and B are positive integers

Solution.

We need to show that: $0 \leq An + B \leq c \bullet n$ for all $n \geq n_o$.

Because A, B and n are positive integers.

1. $0 \leq An + B$ for all $n \geq 1$
2. $An + B \leq An + Bn$ for all $n \geq 1$

Exercise # 1

For each of the following functions, show that f is $O(g)$.

E. $f(n) = An + B$ and $g(n) = n$ where A and B are positive integers

Solution.

We need to show that: $0 \leq An + B \leq c \bullet n$ for all $n \geq n_o$.

Because A, B and n are positive integers.

1. $0 \leq An + B$ for all $n \geq 1$
2. $\underbrace{An + B}_{f(n)} \leq \underbrace{(A + B)n}_{c \bullet g(n)}$ for all $n \geq 1$

Exercise # 1

For each of the following functions, show that f is $O(g)$.

E. $f(n) = An + B$ and $g(n) = n$ where A and B are positive integers

Solution.

We need to show that: $0 \leq An + B \leq c \bullet n$ for all $n \geq n_o$.

Because A, B and n are positive integers.

1. $0 \leq An + B$ for all $n \geq 1$
2. $An + B \leq (A + B)n$ for all $n \geq 1$

Pick $c = A + B$ and $n_o = 1$

Big-*O*

Relatives

Big-Ω

Definition. Let $f(n)$ and $g(n)$ be two functions that are always positive, $f(n)$ is said to be $\Omega(g)$ if and only if :

> There are two constants $c > 0$ and $n_o \geq 0$,
> such that $0 \leq c \bullet g(n) \leq f(n)$ for all $n \geq n_o$

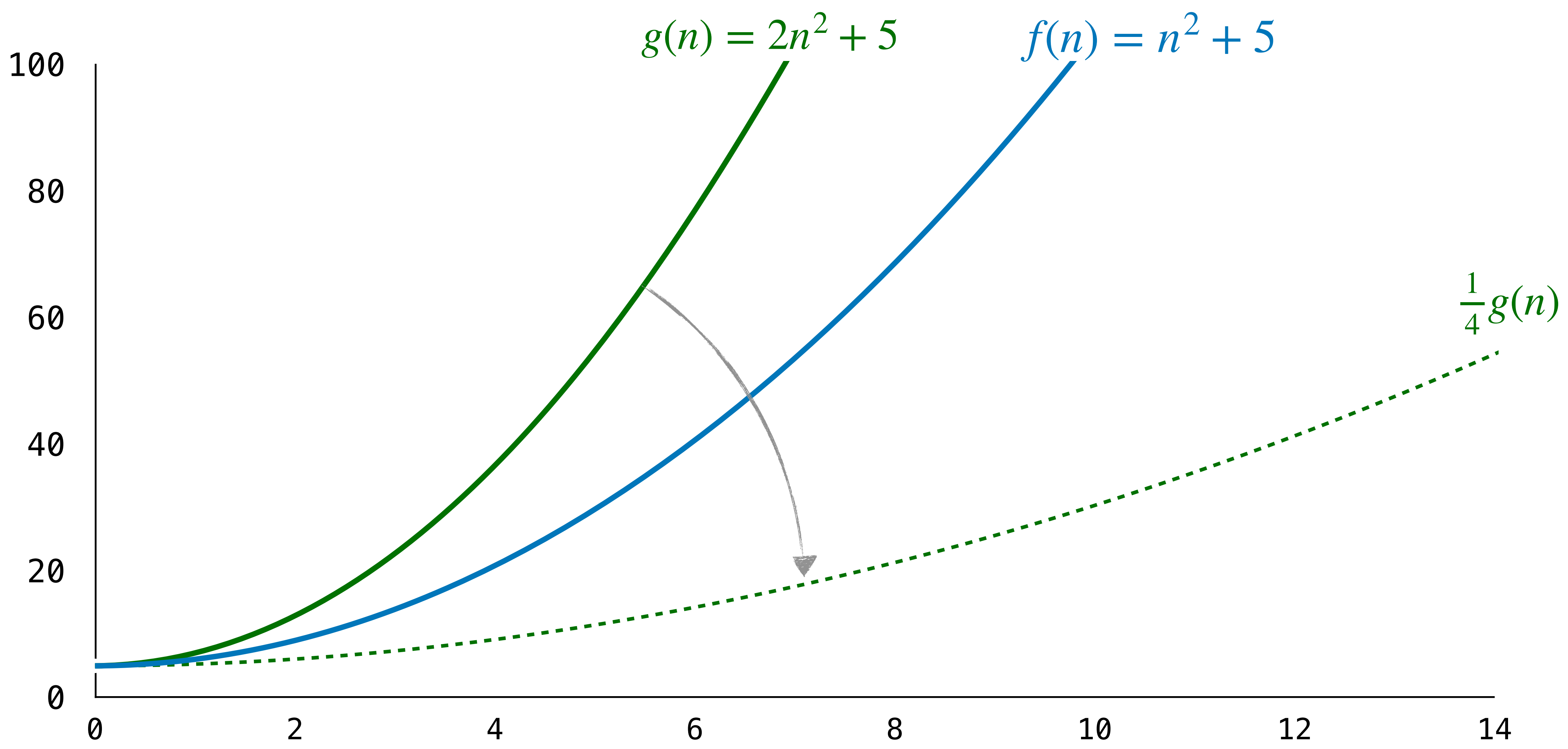
Less formally: If multiplying $g(n)$ by a constant makes it a lower bound for $f(n)$ after some point, then f is $\Omega(g)$.

Big-Ω Example

Assume $f(n) = n^2 + 5$ and $g(n) = 2n^2 + 5$.

f is $\Omega(g)$ because there are c and n_o such that $0 \le c \bullet g(n) \le f(n)$ for all $n \ge n_o$:

If $c = \frac{1}{4}$, then $0 \le \frac{1}{4} \bullet g(n) \le f(n)$ for all $n \ge 1$

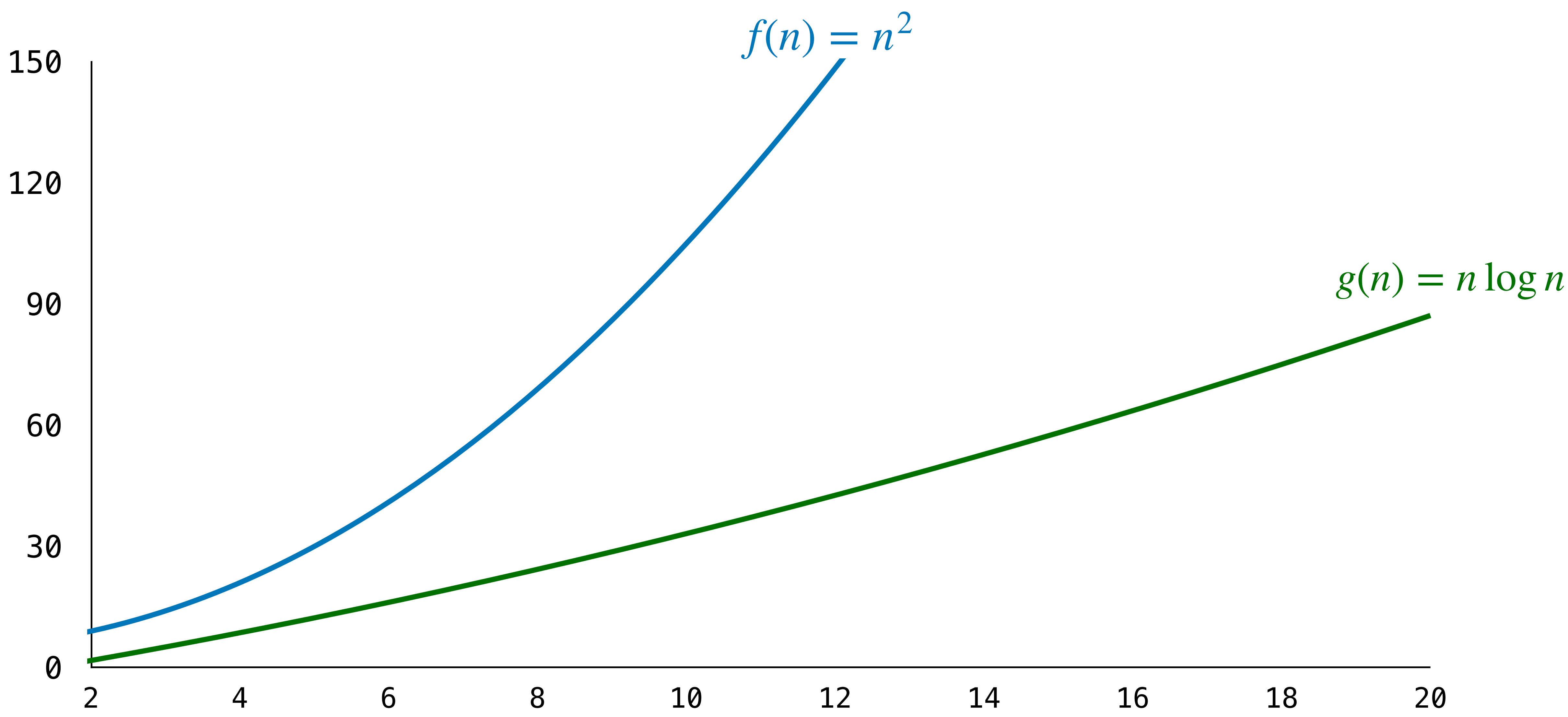


Big-Ω Example

Assume $f(n) = n^2$ and $g(n) = n \log n$.

f is $\Omega(g)$ because there are c and n_o such that $0 \leq c \bullet g(n) \leq f(n)$ for all $n \geq n_o$:

If $c = 1$, then $g(n) \leq f(n)$ for all $n \geq 1$



Quiz # 3

Mark each of the following as **True** or **False**.

The worst case running time for checking if an array of size n is sorted is:

- **A.** $\Omega(\log n)$. **True**: $n = \Omega(\log n)$
- **B.** $\Omega(n)$ **True**: $n = \Omega(n)$
- **C.** $\Omega(n \log n)$ **False**: $n \neq \Omega(n \log n)$
- **D.** $O(\log n)$ **False**: $n \neq O(\log n)$
- **E.** $O(n)$ **True**: $n = O(n)$
- **F.** $O(n \log n)$ **True**: $n = O(n \log n)$

Good and Bad Uses of Big-Ω

Bubble Sort is $\Omega(1)$.

Yea right! All algorithms are $\Omega(1)$!!

An example from the Jordanian market for the weird use of lower bounds!

(Translation: "The mall of burned prices: Everything is for 0.5 Dinars *or more*")

Every comparison-based sorting algorithm performs $\Omega(n \log n)$ comparisons in the worst-case.

Interesting!

In other words. There is no use of trying to find a comparison-based sorting algorithm whose running time in the worst case is *better than* $n \log n$.

Stay tuned for a proof in a couple of weeks from now!

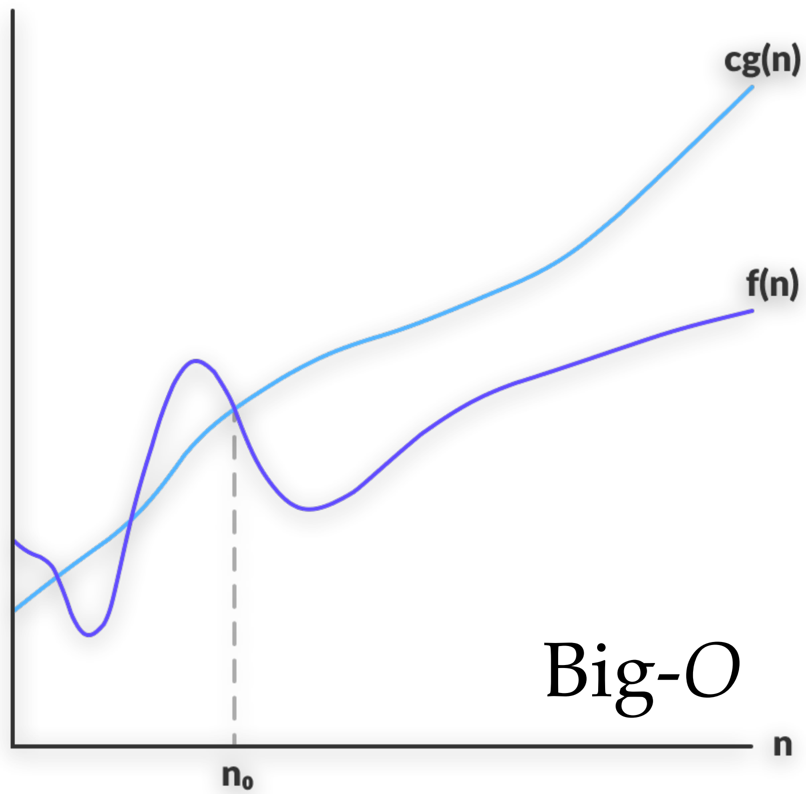
Big-Θ

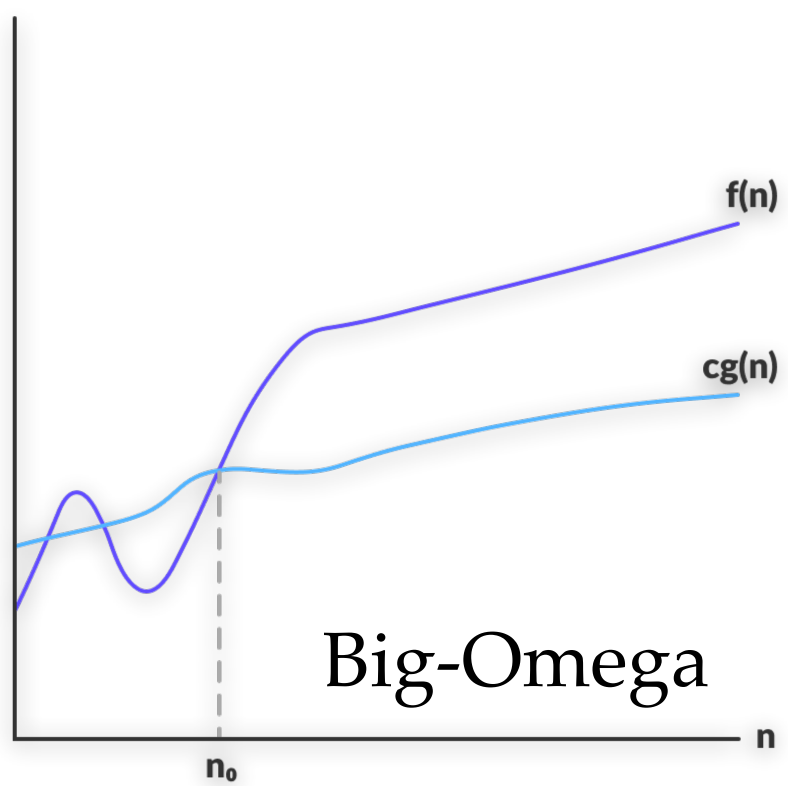
Definition. Let $f(n)$ and $g(n)$ be two functions that are always positive, $f(n)$ is said to be $\Theta(g)$ if and only if :

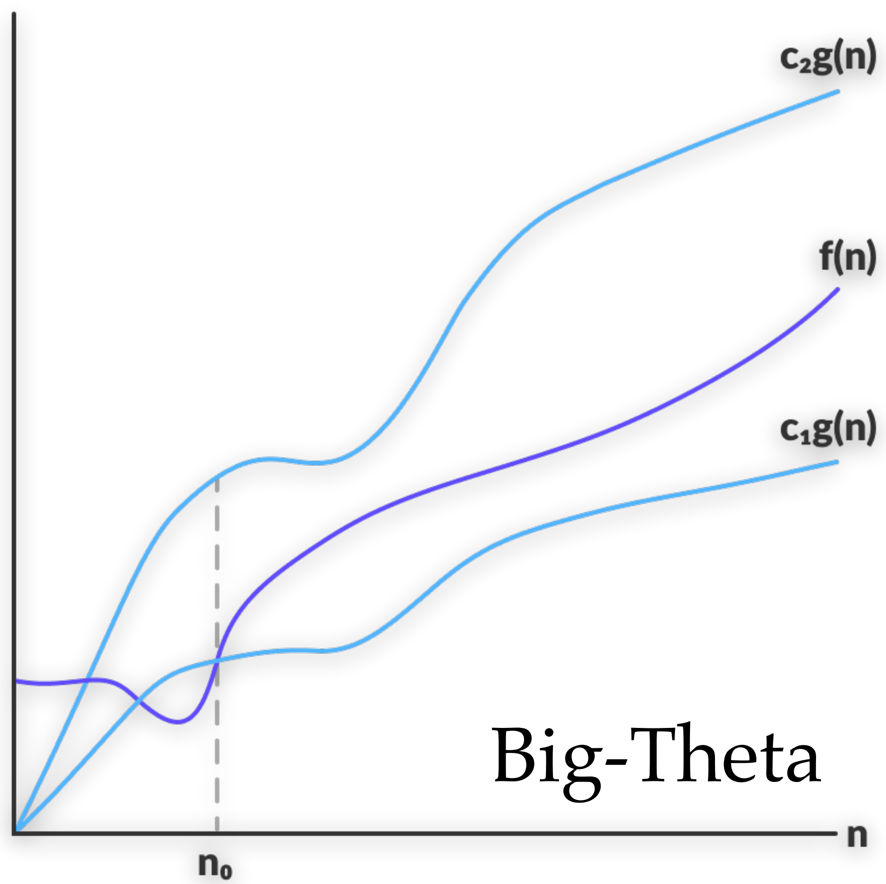
> f is $O(g)$ and f is also $\Omega(g)$

Less formally: If multiplying $g(n)$ by a constant makes it an upper bound for $f(n)$ after some point and also multiplying $g(n)$ by another constant makes it a lower bound for $f(n)$ after some point, then f is $\Theta(g)$.

Big-Θ







Exercises

For each of the following functions, show that f is $\Theta(g)$.

A. $f(n) = 4n + 8$ and $g(n) = n$

Solution.

We need to show that:

$4n + 8 = O(n) \longrightarrow$ pick $c = 12$ and $n_o = 1$

$4n + 8 = \Omega(n) \longrightarrow$ pick $c = 1$ and $n_o = 1$

B. $f(n) = \log_2 n$ and $g(n) = \log_3 n$

Solution.

We need to show that:

$$\log_2 n = O\left(\frac{\log_2 n}{\log_2 3}\right)$$

$$\log_2 n = \Omega\left(\frac{\log_2 n}{\log_2 3}\right)$$

Remember:

$$\log_b(a) = \frac{\log_x(a)}{\log_x(b)}$$

Exercises

For each of the following functions, show that f is $\Theta(g)$.

A. $f(n) = 4n + 8$ and $g(n) = n$

Solution.

We need to show that:

$4n + 8 = O(n)$ → pick $c = 12$ and $n_o = 1$

$4n + 8 = \Omega(n)$ → pick $c = 1$ and $n_o = 1$

B. $f(n) = \log_2 n$ and $g(n) = \log_3 n$

Solution.

We need to show that:

$\log_2 n = O(\frac{\log_2 n}{\log_2 3})$ → pick $c \geq \log_2 3$ and $n_o = 1$

$\log_2 n = \Omega(\frac{\log_2 n}{\log_2 3})$ → pick $c = 1$ and $n_o = 1$

Exercises

Show that $n^3 + n$ is **not** $O(n^2)$.

Solution.

Assume for the sake of contradiction that there exist two constants c and n_o such that $0 \leq n^3 + n \leq c \bullet n^2$ for all $n \geq n_o$.

Divide by n^2: $0 \leq n + \frac{1}{n} \leq c$

This is clearly false because $n + \frac{1}{n}$ is strictly increasing while the right hand side is constant.

Exercises

Show that n^2 is **not** $\Theta(n^3)$.

Solution.

Assume for the sake of contradiction that $n^2 = \Omega(n^3)$, then there exist two constants c and n_o such that $0 \leq c \bullet n^3 \leq n^2$ for all $n \geq n_o$.

Divide by n^2: $0 \leq c \bullet n \leq 1$

This is clearly false because $c \bullet n$ is strictly increasing while the right hand side is constant.

Quiz # 4

Which of the following is true about the running time of **insertion sort**?

A. The running time is $O(n^2)$

B. The running time is $\Omega(n)$

C. The best case is $\Theta(n)$.

D. The worst case is $\Theta(n^2)$.

E. All of the above.

Small-o and Small-ω

Informal Definition. f is said to be $o(g)$ if it grows strictly slower than g.

Informal Definition. f is said to be $\omega(g)$ if it grows strictly faster than g.

Notation	Order of Growth Relation	Example
$f = O(g)$	$f \preceq g$	If $f = O(n^2)$, examples for f could be: $n^2,\ 3n^2 + n,\ 5n - 1,\ 7n \log n + 5n,\ \sqrt{n}$
$f = o(g)$	$f \prec g$	If $f = o(n^2)$, examples for f could be: $n^{1.9},\ 5n - 1,\ 7n \log n + 5n,\ \sqrt{n}$
$f = \Omega(g)$	$f \succeq g$	If $f = \Omega(n^2)$, examples for f could be: $n^2,\ 3n^2 + n,\ 5n^3,\ 7n^5,\ 2^n$
$f = \omega(g)$	$f \succ g$	If $f = \omega(n^2)$, examples for f could be: $n^{2.01},\ n^2 \log n,\ 5n^3,\ 7n^5,\ 2^n$
$f = \Theta(g)$	$f \asymp g$	If $f = \Theta(n^2)$, examples for f could be: $n^2,\ 3n^2,\ 5n^2 - n,\ 7n^2 + n \log n + 100$

Quiz # 5

Assume that a function f is known to be $o(n^2)$ and also known to be $\Omega(\log n)$, which of the following functions can f possibly be?
Choose all that applies.

A. n^n

B. 2^n

C. n^3

D. $n^2 \log n$

E. n^2

F. $n\sqrt{n}$

G. $n^{1.1}$

H. $n \log n$

I. n

J. $\sqrt{n}$

K. $\log^2 n$

L. $\log n$

M. $\log(\log n)$

M. 100

Small-o and Small-ω

Informal Definition. f is said to be $o(g)$ if it grows strictly slower than g.

Informal Definition. f is said to be $\omega(g)$ if it grows strictly faster than g.

Examples.

$3n^2$ vs n^2	$3n^2$ vs n^3	$3n^3$ vs n^2
$3n^2 = O(n^2)$	$3n^2 = O(n^3)$	$3n^3 \neq O(n^2)$
$3n^2 = \Omega(n^2)$	$3n^2 \neq \Omega(n^3)$	$3n^3 = \Omega(n^2)$
$3n^2 = \Theta(n^2)$	$3n^2 \neq \Theta(n^3)$	$3n^3 \neq \Theta(n^2)$
$3n^2 \neq o(n^2)$	$3n^2 = o(n^3)$	$3n^3 \neq o(n^2)$
$3n^2 \neq \omega(n^2)$	$3n^2 \neq \omega(n^3)$	$3n^3 = \omega(n^2)$

Quiz # 6

Consider $f(n) = O(g(n))$. Which of the following is definitely true?
Choose all that applies.

A. $f = \Theta(g)$

B. $f = o(g)$

C. $g = \Omega(f)$

D. $g = \omega(f)$

Properties

- Reflexivity. f is $\Theta(f)$ and $O(f)$ and $\Omega(f)$ but not $o(f)$ or $\omega(f)$

- Constants. If f is $\Theta(g)$ and $c > 0$, then $c \bullet f$ is $\Theta(g)$.

Properties

- Reflexivity. f is $\Theta(f)$.

- Constants. If f is $\Theta(g)$ and $c > 0$, then $c \bullet f$ is $\Theta(g)$.

  Example: $4n^2 + 5$ is $\Theta(n^2)$ and $4 \times (4n^2 + 5)$ is also $\Theta(n^2)$.

  Similarly: If f is $O(g)$ and $c > 0$, then $c \bullet f$ is $O(g)$.

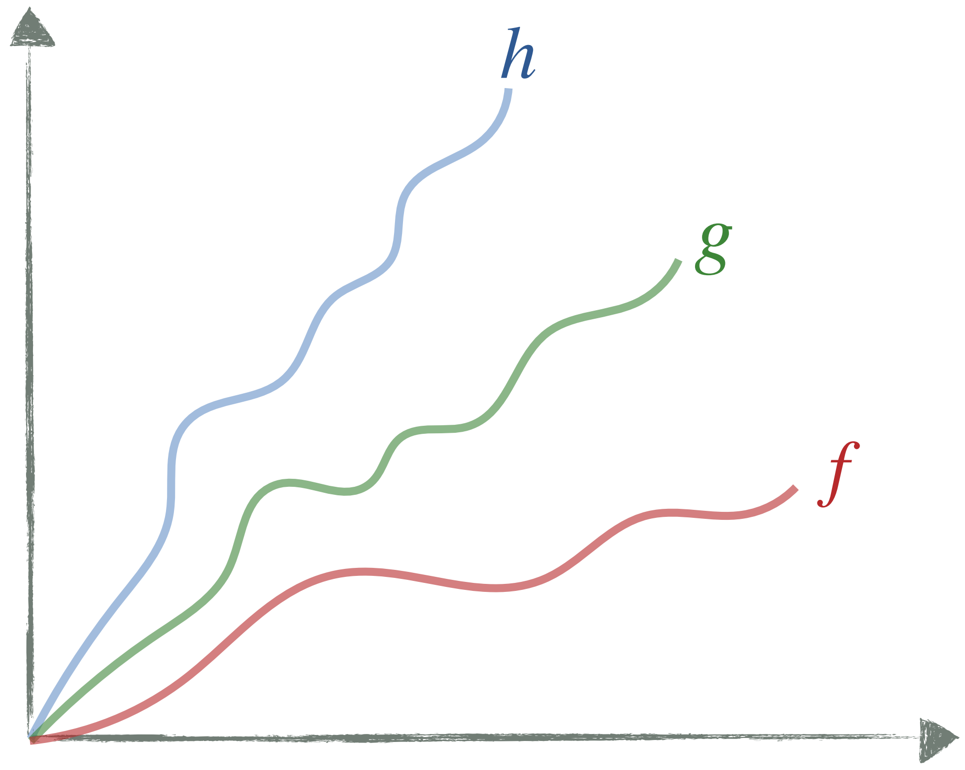
  If f is $\Omega(g)$ and $c > 0$, then $c \bullet f$ is $\Omega(g)$.

  If f is $o(g)$ and $c > 0$, then $c \bullet f$ is $o(g)$.

  If f is $\omega(g)$ and $c > 0$, then $c \bullet f$ is $\omega(g)$.

Properties

- Reflexivity. f is $\Theta(f)$.
- Constants. If f is $\Theta(g)$ and $c > 0$, then $c \bullet f$ is $\Theta(g)$.
- Transitivity. If f is $O(g)$ and g is $O(h)$ then f is $O(h)$.



h is an upper bound for both g and f

Properties

- Reflexivity. f is $\Theta(f)$.
- Constants. If f is $\Theta(g)$ and $c > 0$, then $c \bullet f$ is $\Theta(g)$.
- Transitivity. If f is $O(g)$ and g is $O(h)$ then f is $O(h)$.

Similarly: If f is $\Theta(g)$ and g is $\Theta(h)$ then f is $\Theta(h)$.

If f is $\Omega(g)$ and g is $\Omega(h)$ then f is $\Omega(h)$.

If f is $o(g)$ and g is $o(h)$ then f is $o(h)$.

If f is $\omega(g)$ and g is $\omega(h)$ then f is $\omega(h)$.

Properties

- Reflexivity. f is $\Theta(f)$.
- Constants. If f is $\Theta(g)$ and $c > 0$, then $c \bullet f$ is $\Theta(g)$.
- Transitivity. If f is $\Theta(g)$ and g is $\Theta(h)$ then f is $\Theta(h)$.
- Sums. If f_1 is $\Theta(g_1)$ and f_2 is $\Theta(g_2)$, then $f_1 + f_2$ is $\Theta(\max\{g_1, g_2\})$.

  Example: If $f_1(n)$ is $\Theta(n^2)$ and $f_2(n)$ is $\Theta(n^3)$ then $f_1 + f_2$ is $\Theta(n^3)$.

  Similarly: If f_1 is $O(g_1)$ and f_2 is $O(g_2)$, then $f_1 + f_2$ is $O(\max\{g_1, g_2\})$.

  If f_1 is $\Omega(g_1)$ and f_2 is $\Omega(g_2)$, then $f_1 + f_2$ is $\Omega(\max\{g_1, g_2\})$.

  If f_1 is $o(g_1)$ and f_2 is $o(g_2)$, then $f_1 + f_2$ is $o(\max\{g_1, g_2\})$.

  If f_1 is $\omega(g_1)$ and f_2 is $\omega(g_2)$, then $f_1 + f_2$ is $\omega(\max\{g_1, g_2\})$.

قل ولا تقل

- Don't say: "My algorithm is $O(n^2)$"

  Say: "The running time of my algorithm" is $O(n^2)$ or "My algorithm runs in $O(n^2)$".

  Explanation. An algorithm is not a function, its running time is.

قل ولا تقل

- Don't say: "My algorithm is $O(n^2)$"

  Say: "The running time of my algorithm" is $O(n^2)$ or "My algorithm runs in $O(n^2)$".

- Don't say: "Your algorithm runs in at least $O(n^2)$"

  Say: "Your algorithm runs in $\Omega(n^2)$" or "Your algorithm runs in at least $\Theta(n^2)$"

Explanation. $O(n^2)$ describes all the functions whose order of growth is n^2 or less (e.g. $\log(n)$, $\sqrt{n}$, n, $n\log(n)$, etc.)
Saying that the running time is *at least* one of these functions means that the running time could be anything!

قل ولا تقل

- Don't say: "My algorithm is $O(n^2)$"

  Say: "The running time of my algorithm" is $O(n^2)$ or "My algorithm runs in $O(n^2)$".

- Don't say: "Your algorithm runs in at least $O(n^2)$"

  Say: "Your algorithm runs in $\Omega(n^2)$" or "Your algorithm runs in at least $\Theta(n^2)$"

- Avoid saying: "The worst case running time of Bubble Sort is $O(n^2)$"

  Say: "The worst case running time of Bubble Sort is $\Theta(n^2)$"

If you meant to say: *I don't know the order of growth of the running time of Bubble Sort in the worst case, but it should not be more than n^2*, then using Big-O is appropriate.

Remember. $O(n^2)$ means: in the order of n^2 *or less*
$\Theta(n^2)$ means: in the order of n^2

قل ولا تقل

- Don't say: "My algorithm is $O(n^2)$"

  Say: "The running time of my algorithm" is $O(n^2)$ or "My algorithm runs in $O(n^2)$".

- Don't say: "Your algorithm runs in at least $O(n^2)$"

  Say: "Your algorithm runs in $\Omega(n^2)$" or "Your algorithm runs in at least $\Theta(n^2)$"

- Avoid saying: "The worst case running time of Bubble Sort is $O(n^2)$"

  Say: "The worst case running time of Bubble Sort is $\Theta(n^2)$"

! $O(g(n))$ is a set of functions, but computer scientists often *abuse* the notation by writing $f(n) = O(g(n))$ instead of $f(n) \in O(g(n))$.

Alternative Definitions

order of growth relationship

if	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$	*then*	$f = o(g)$	$f(n) \prec g(n)$
if	$0 \leq \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$	*then*	$f = O(g)$	$f(n) \preceq g(n)$
if	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$	*then*	$f = \omega(g)$	$f(n) \succ g(n)$
if	$0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq \infty$	*then*	$f = \Omega(g)$	$f(n) \succeq g(n)$
if	$0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$	*then*	$f = \Theta(g)$	$f(n) \asymp g(n)$

Optional Example

Show that $\log_2(n) \times \log_2(n) = O(n)$

Solution.

This is equivalent to showing that $\log_2(n) = O(\sqrt{n})$

We need to show that: $$0 \leq \lim_{n\to\infty} \frac{\log_2(n)}{\sqrt{n}} < \infty$$

Using L'Hôpital's rule:

$$\lim_{x\to c} \frac{f(x)}{g(x)} = \lim_{x\to c} \frac{f'(x)}{g'(x)}.$$

$$\lim_{n\to\infty} \frac{\log_2(n)}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{1}{n \bullet \ln 2}}{\frac{1}{2\sqrt{n}}}$$

$$= \lim_{n\to\infty} \frac{2\sqrt{n}}{n \bullet \ln 2} = \lim_{n\to\infty} \frac{2\sqrt{n}}{\sqrt{n}\sqrt{n} \bullet \ln 2}$$

$$= \lim_{n\to\infty} \frac{2}{\sqrt{n} \bullet \ln 2} = 0$$

Optional Example

Show that $\log_2(n) \times \log_2(n) = O(n)$

Solution.

This is equivalent to showing that $\log_2(n) = O(\sqrt{n})$

We need to show that: $$0 \leq \lim_{n\to\infty} \frac{\log_2(n)}{\sqrt{n}} < \infty$$

Using L'Hôpital's rule: $$\lim_{n\to\infty} \frac{\log_2(n)}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{1}{n \bullet \ln 2}}{\frac{1}{2\sqrt{n}}}$$

$$\lim_{x\to c} \frac{f(x)}{g(x)} = \lim_{x\to c} \frac{f'(x)}{g'(x)}.$$

$$= \lim_{n\to\infty} \frac{2\sqrt{n}}{n \bullet \ln 2} = \lim_{n\to\infty} \frac{2\sqrt{n}}{\sqrt{n}\sqrt{n} \bullet \ln 2}$$

$$= \lim_{n\to\infty} \frac{2}{\sqrt{n} \bullet \ln 2} = 0$$

Remember. $\log^c n = o(n^d)$
where $c > 0$ and $d > 0$ are positive constants.

A MUST KNOW !

Optional Example

Prove by induction that $2^n = O(n!)$

Solution.

We need to show that there exist two constants c and n_o such that
$0 \leq 2^n \leq c \bullet n!$ for all $n \geq n_o$.

Assume $c = 1$

i. When $n = 4$, $2^n = 16$ while $n! = 24$. Therefore, the inequality holds for $n = 4$.

ii. Assuming that $0 \leq 2^k \leq k!$ is true for all $4 \leq k \leq m$,
we will show that $0 \leq 2^{m+1} \leq (m+1)!$ is also true.

Rewriting the equation: $0 \leq 2^1 \bullet 2^m \leq (m+1) \bullet m!$

This is clearly true, since $2^1 \leq (m+1)$ since $m \geq 4$ and
we know from the induction hypothesis that $2^m \leq m!$

Optional Example

Prove by induction that $2^n = O(n!)$

Solution.

We need to show that there exist two constants c and n_o such that
$0 \leq 2^n \leq c \bullet n!$ for all $n \geq n_o$.

Assume $c = 1$

i. When $n = 4$, $2^n = 16$ while $n! = 24$. Therefore, the inequality holds for $n = 4$.

ii. Assuming that $0 \leq 2^k \leq k!$ is true for all $4 \leq k \leq m$,
we will show that $0 \leq 2^{m+1} \leq (m+1)!$ is also true.

Rewriting the equation: $0 \leq 2^1 \bullet 2^m \leq (m+1) \bullet m!$

This is clearly true, since $2^1 \leq (m+1)$ since $m \geq 4$ and
we know from the induction hypothesis that $2^m \leq m!$

Therefore, $0 \leq 2^n \leq c \bullet n!$ for all $n \geq n_o$ is true if we pick $c = 1$ and $n_o = 4$.

Exercises

Stirling's Approximation states that:

$$\log_2(n!) \quad = \quad n\log_2 n - n\log_2 e + r\log_2 n \qquad (r \text{ is a positive constant})$$

Show that $\log_2(n!) = \Theta(n \log n)$ without using Stirling's Approximation.

Solution.

1. $$\begin{aligned} \log(1\times 2\times 3\times \ldots \times n) &\leq \log(n\times n\times n\times \ldots\times n) && \text{for all } n\geq 1\\ \log(1\times 2\times 3\times \ldots \times n) &\leq \log(n^n) && \text{for all } n\geq 1\\ \log(1\times 2\times 3\times \ldots \times n) &\leq n\log(n) && \text{for all } n\geq 1 \end{aligned}$$

   Therefore $\log_2(n!) = O(n\log n)$ because $0 \leq \log(n!) \leq 1 \bullet n\log n$ for all $n \geq 1$

2. $$\begin{aligned} \log_2(n!) &= \log_2(1\times 2\times 3\times\ldots\times \tfrac{n}{2}\times(\tfrac{n}{2}+1)\times(\tfrac{n}{2}+2)\times\ldots\times n)\\ &= \log(1)+\log(2)+\log(3)+\ldots+\log(\tfrac{n}{2})+\log(\tfrac{n}{2}+1)+\log(\tfrac{n}{2}+2)+\ldots+\log(n)\\ &\geq \cancel{\log(1)+\log(2)+\log(3)+\ldots+}\log(\tfrac{n}{2})+\log(\tfrac{n}{2}+1)+\log(\tfrac{n}{2}+2)+\ldots+\log(n)\\ &\geq \cancel{\log(1)+\log(2)+\log(3)+\ldots+}\log(\tfrac{n}{2})+\log(\tfrac{n}{2})+\log(\tfrac{n}{2})+\ldots+\log(\tfrac{n}{2})\\ &\geq \tfrac{n}{2}\log(\tfrac{n}{2}) \geq \tfrac{n}{2}(\log(n)-\log(2)) \geq \tfrac{n}{2}(\log(n)-1) \geq \tfrac{n}{2}(\log(n)-\tfrac{1}{2}\log(n)) \end{aligned}$$

   Therefore $\log_2(n!) = \Omega(n\log n)$ because $0 \leq \frac{1}{4}\bullet n\log n \leq \log(n!)$ for all $n \geq 4$

Optional Examples

We know that $\sum_{i=0}^{n} i^2$ can be computed using the formula: $\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$

Show that $\sum_{i=0}^{n} i^2 = \Theta(n^3)$ without using the above formula.

Solution.

1. $1^2 + 2^2 + 3^2 + \ldots + n^2 \leq n^2 + n^2 + n^2 + \ldots + n^2$ for all $n \geq 1$

   $1^2 + 2^2 + 3^2 + \ldots + n^2 \leq n \times n^2$ for all $n \geq 1$

   Therefore, $1^2 + 2^2 + 3^2 + \ldots + n^2 = O(n^3)$ pick $c = 1$ and $n_o = 1$

2. $1^2 + 2^2 + 3^2 + \ldots + (\frac{n}{2})^2 + (\frac{n}{2}+1)^2 + (\frac{n}{2}+2)^2 + \ldots + n^2$

   $\geq \cancel{1^2 + 2^2 + 3^2 + \ldots +} (\frac{n}{2})^2 + (\frac{n}{2}+1)^2 + (\frac{n}{2}+2)^2 + \ldots + n^2$ for all $n \geq 1$

   $\geq \cancel{1^2 + 2^2 + 3^2 + \ldots +} (\frac{n}{2})^2 + (\frac{n}{2})^2 + (\frac{n}{2})^2 + \ldots + (\frac{n}{2})^2$ for all $n \geq 1$

   $\geq \frac{n}{2} \times (\frac{n}{2})^2 \geq \frac{n}{2} \times \frac{n^2}{4} \geq \frac{n^3}{8}$ for all $n \geq 1$

   Therefore, $1^2 + 2^2 + 3^2 + \ldots + n^2 = \Omega(n^3)$ pick $c = \frac{1}{8}$ and $n_o = 1$

Yet Another Asymptotic Notation: The Tilde ~ Notation

If $\dfrac{f(n)}{g(n)}$ approaches 1 when n approaches infinity, we say: $f(n) \sim g(n)$

Less formally. $f(n) \sim g(n)$ if the two functions are equal after dropping the lower order terms (keeping the coefficients)

Example 1. $\underbrace{5n^3 + n^2 - 5}_{} \sim \underbrace{5n^3}_{}$

the two functions are
asymptotically equivalent
(equal when n approaches infinity)

Yet Another Asymptotic Notation: The Tilde ~ Notation

If $\dfrac{f(n)}{g(n)}$ approaches 1 when n approaches infinity, we say: $f(n) \sim g(n)$

Less formally. $f(n) \sim g(n)$ if the two functions are equal after dropping the lower order terms (keeping the coefficients)

Example 1. $5n^3 + n^2 - 5 \sim 5n^3$

Example 2. $\sum_{i=0}^{n} i \sim \frac{1}{2}n^2$

Explanation. $\sum_{i=0}^{n} i = \dfrac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$

Dropping the lower order terms,
we are left with $\frac{1}{2}n^2$

Example 3. $\log(n!) \sim n \log n$

Explanation. $\log_2(n!) = n\log_2 n - n\log_2 e + r\log_2 n$

Using Stirling's approximation.
Dropping the lower order terms,
we are left with $n\log_2(n)$