Design and Analysis Algorithm

Drs. Achmad Ridok M.Kom
Imam Cholissodin, S.Si., M.Kom
M. Ali Fauzi, S.Kom., M.Kom
Ratih Kartika Dewi, ST, M.Kom

Pertemuan 03

Contents

Asymptotic Notation

Contents

- Asymptotic Notations:
  - O (big oh)
  - Ω (big omega)
  - Θ (big theta)
- Basic Efficiency Classes

In the following discussion…

- t(n) & g(n): any nonnegative functions defined on the set of natural numbers

- t(n) → an algorithm's running time
  - Usually indicated by its basic operation count C(n)
- g(n) → some simple function to compare the count with

$O(g(n))$: Informally

- **$O(g(n))$** is a set of all functions with a **smaller** or **same** order of growth as g(n)
- Examples:
  - $n \in O(n^2)$; $100n + 5 \in O(n^2)$
  - $\frac{1}{2} n (n-1) \in O(n^2)$
  - $n^3 \notin O(n^2)$; $0.0001\, n^3 \notin O(n^2)$; $n^4+n+1 \notin O(n^2)$

$\leq$

$\Omega(g(n))$: Informally

- **$\Omega(g(n))$** is a set of all functions with a **larger** or **same** order of growth as g(n)
- Examples:
  - $n^3 \in \Omega(n^2)$
  - $\frac{1}{2} n (n-1) \in \Omega(n^2)$
  - $100n + 5 \notin \Omega(n^2)$

$\geq$

Θ(g(n)): Informally

- **Θ(g(n))** is a set of all functions with a **same** order of growth as g(n)
- Examples:
  - $an^2+bn+c;\ a>0 \in \Theta(n^2)$; $n^2+\sin n \in \Theta(n^2)$
  - $\frac{1}{2} n\ (n-1) \in \Theta(n^2)$; $n^2+\log n \in \Theta(n^2)$
  - $100n + 5 \notin \Theta(n^2)$; $n^3 \notin \Theta(n^2)$

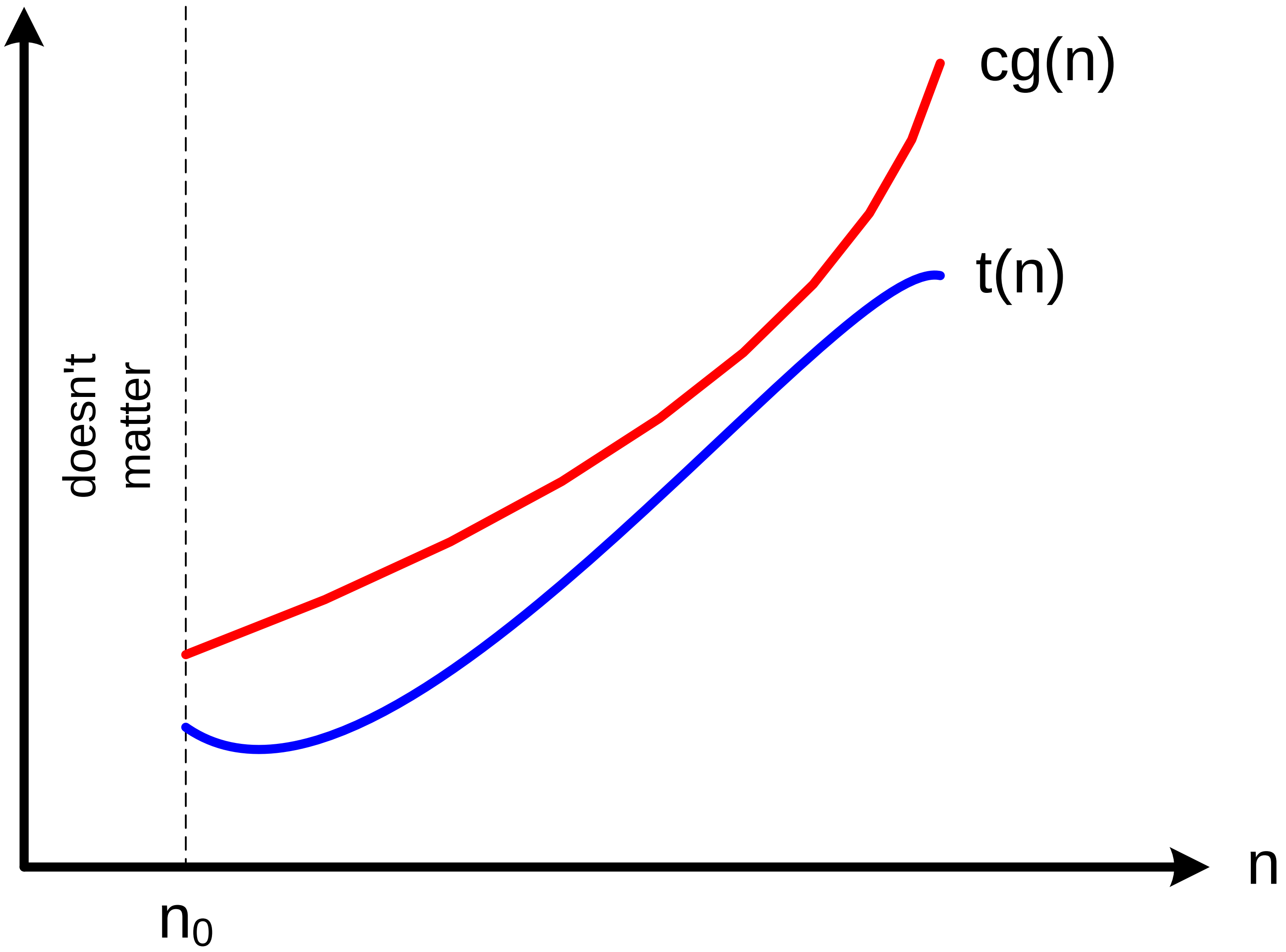
O-notation: Formally

- **DEF1**: A function t(n) is said to be in $O(g(n))$, denoted $t(n) \in O(g(n))$, if t(n) is bounded **above** by some constant multiple of g(n) for all large n

- i.e. there exist some positive constant c and some nonnegative integer n_0, such that

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

t(n) ∈ *O*(g(n)): Illustration



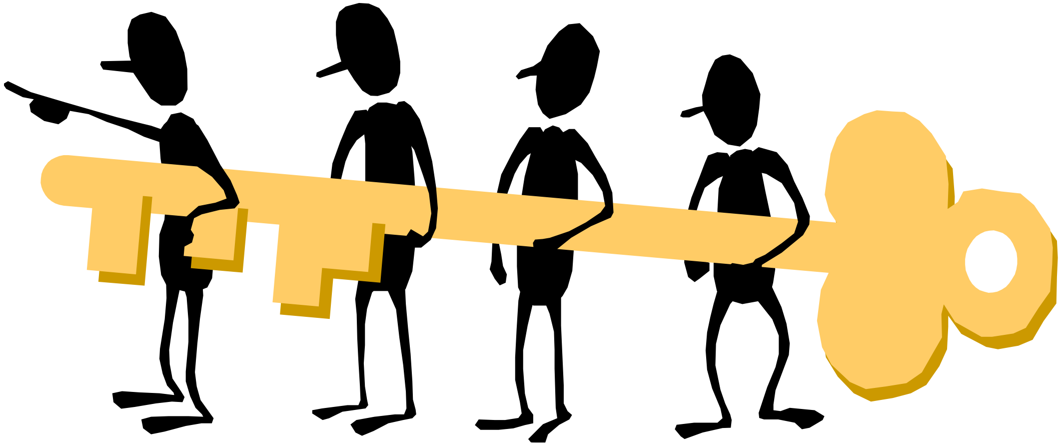
Proving Example: $100n + 5 \in O(n^2)$

- Remember DEF1: find c and n_0, such that $t(n) \leq cg(n)$ for all $n \geq n_0$

- $100n + 5 \leq 100n + n$ (for all $n \geq 5$) $= 101n \leq 101n^2 \rightarrow c=101, n_0=5$
- $100n + 5 \leq 100n + 5n$ (for all $n \geq 1$) $= 105n \leq 105n^2 \rightarrow c=105, n_0=1$
- …

Big-Oh

- The O symbol was introduced in 1927 to indicate relative growth of two functions based on asymptotic behavior of the functions now used to classify functions and families of functions



Upper Bound Notation

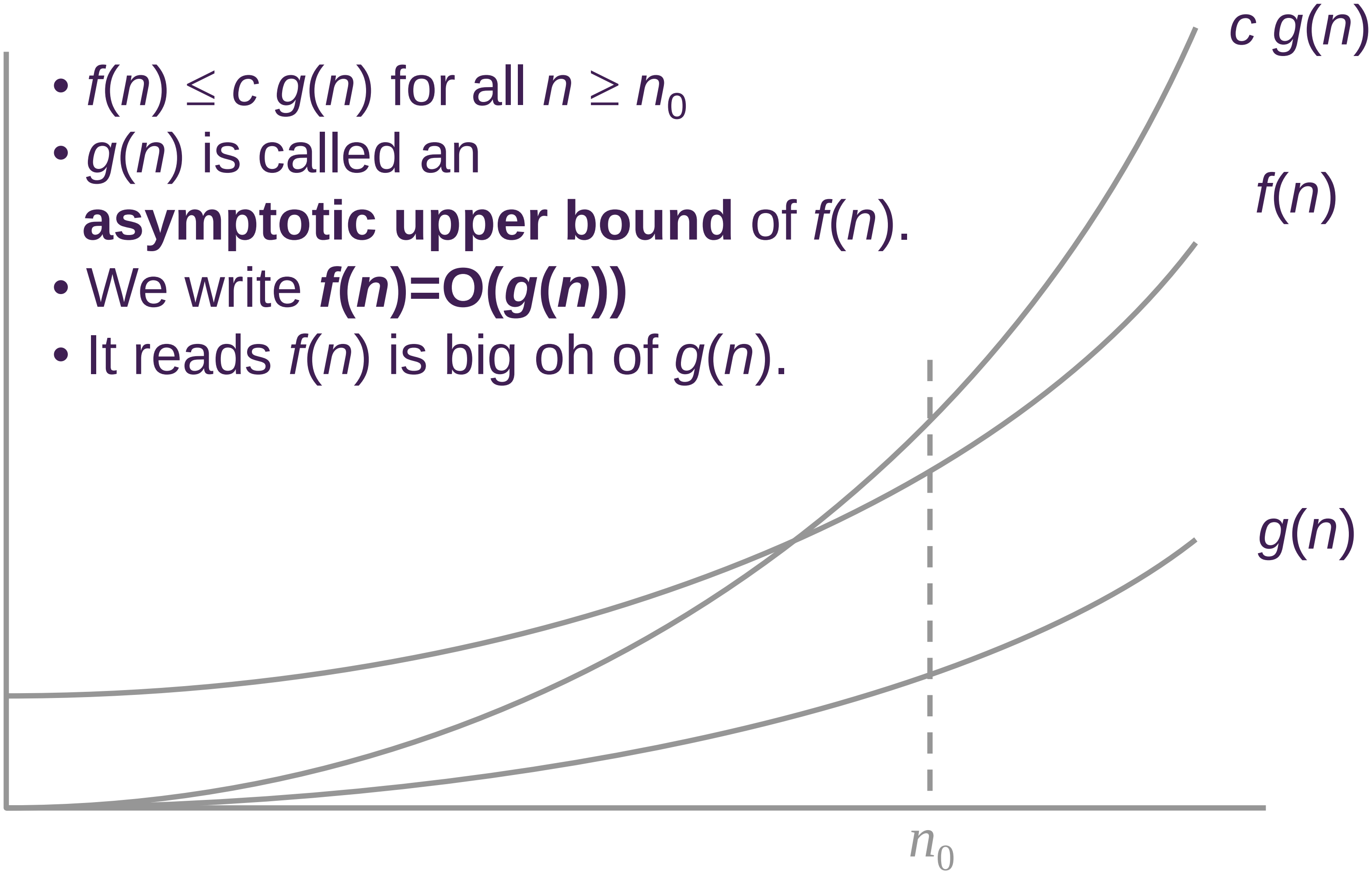
- We say Insertion Sort's run time is $O(n^2)$
  - Properly we should say run time is *in* $O(n^2)$
  - Read O as "Big-O" (you'll also hear it as "order")

- In general a function
  - $f(n)$ is $O(g(n))$ if $\exists$ positive constants c and n_0 such that $f(n) \leq c \cdot g(n)\ \forall\ n \geq n_0$

- e.g. if $f(n)=1000n$ and $g(n)=n^2$, $n_0 \geq 1000$ and $c = 1$ then $f(n_0) \leq 1.g(n_0)$ and we say that $f(n) = O(g(n))$

Asymptotic Upper Bound

- $f(n) \leq c\, g(n)$ for all $n \geq n_0$
- $g(n)$ is called an **asymptotic upper bound** of $f(n)$.
- We write $\boldsymbol{f(n)=O(g(n))}$
- It reads $f(n)$ is big oh of $g(n)$.



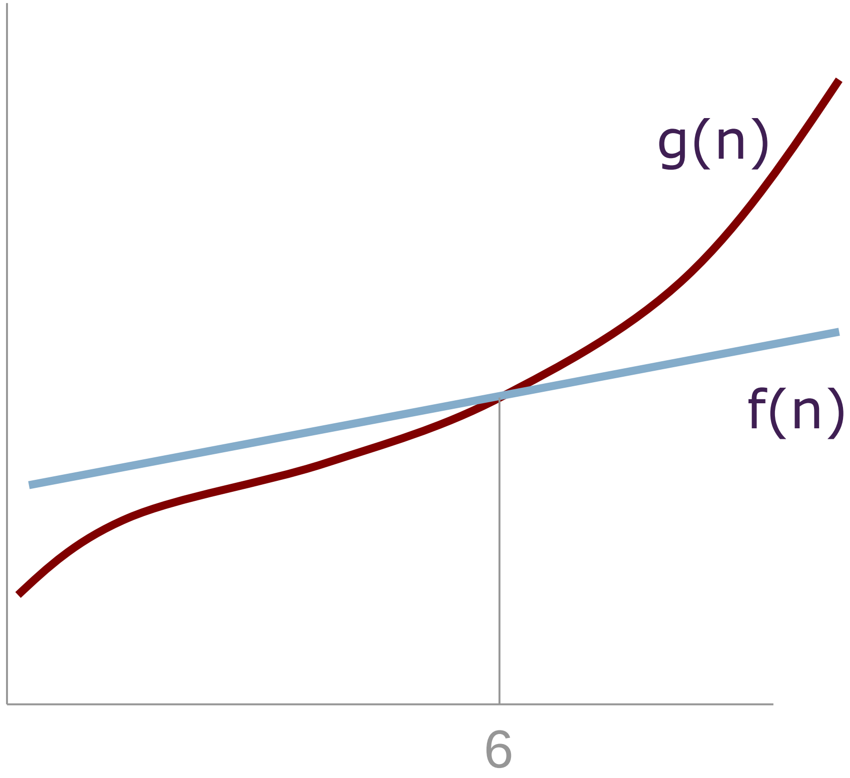
Big-Oh, the Asymptotic Upper Bound

- This is the most popular notation for run time since we're usually looking for **worst case time**.
- If Running Time of Algorithm X is $O(n^2)$, then for any input the running time of algorithm X is at most a quadratic function, for sufficiently large n.

- e.g. $2n^2 = O(n^3)$.
- From the definition using $c = 1$ and $n_0 = 2$. $O(n^2)$ is tighter than $O(n^3)$.

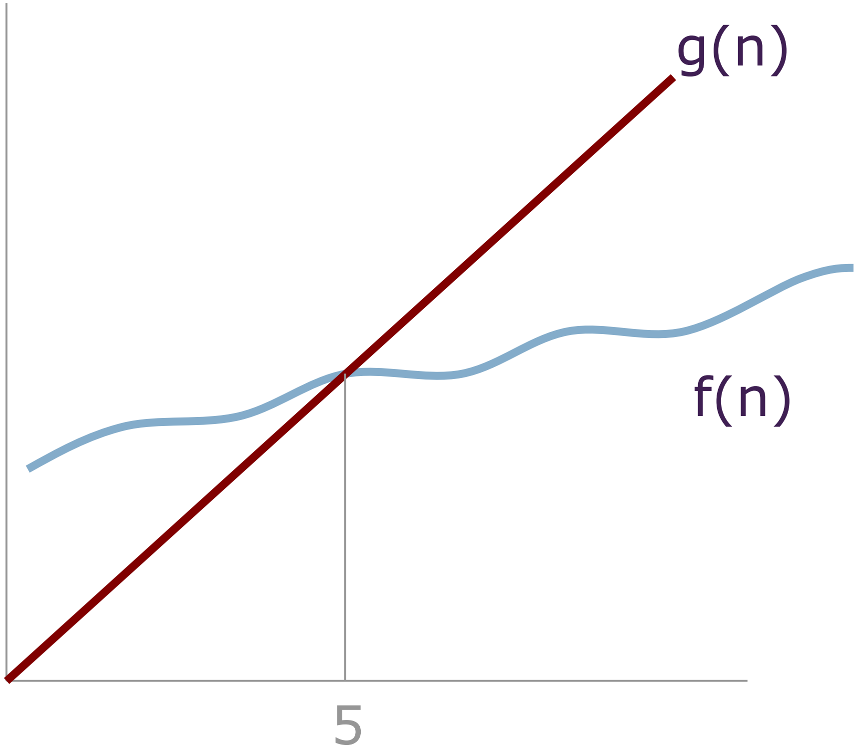
Example 1

for all *n*>6, g(*n*) > 1 f(*n*).
Thus the function *f* is in the big-O of *g*.
that is, *f*(*n*) in O(*g*(*n*)).



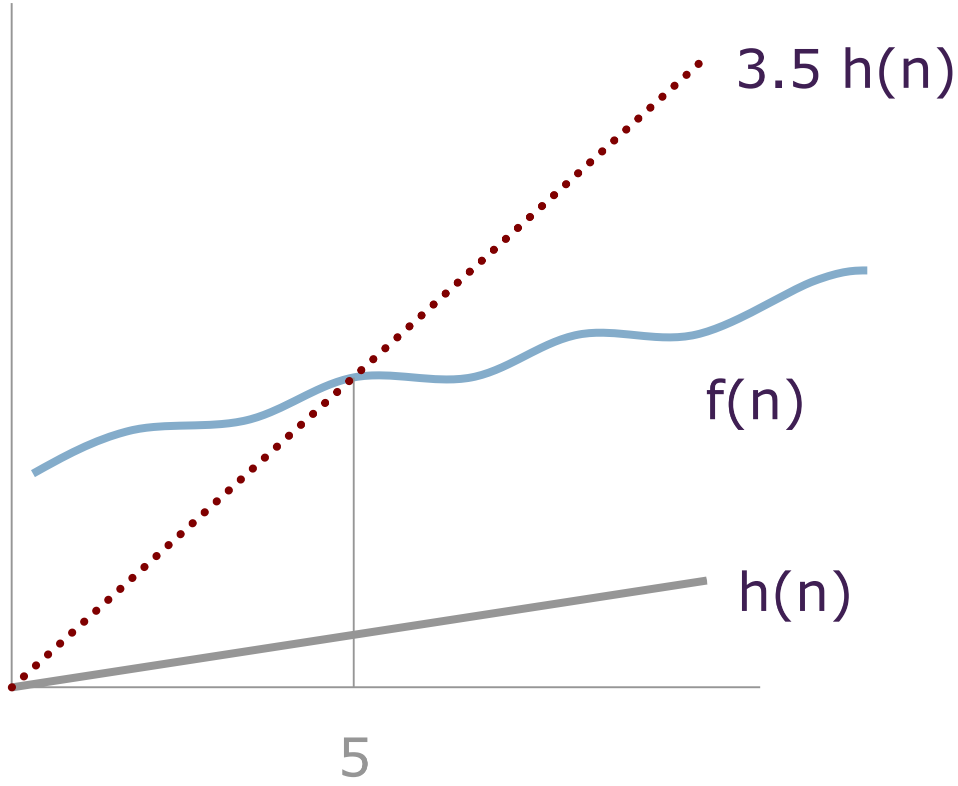
Example 2

There exists a n_0=5 s.t. for all n>n_0, f(n) < 1 g(n).
Thus, f(n) is in O(g(n)).



Example 3

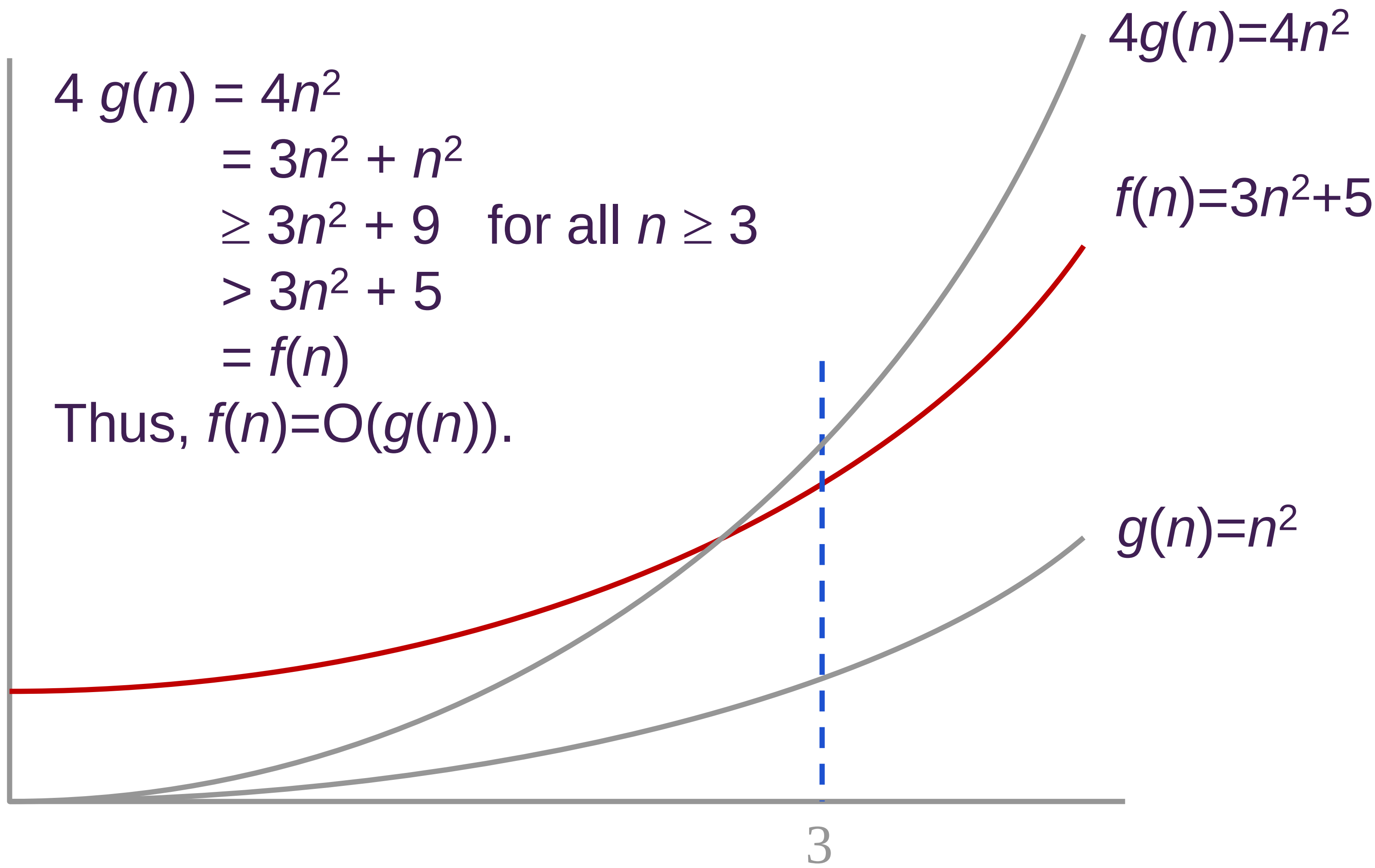
There exists a n_0=5, c=3.5, s.t. for all $n>n_0$, f(n) < c h(n). Thus, f(n) is in O(h(n)).



Example of Asymptotic Upper Bound

$$
\begin{aligned}
4\,g(n) &= 4n^2 \\
&= 3n^2 + n^2 \\
&\geq 3n^2 + 9 \quad \text{for all } n \geq 3 \\
&> 3n^2 + 5 \\
&= f(n)
\end{aligned}
$$

Thus, $f(n)=O(g(n))$.



Exercise on O-notation

- Show that $3n^2+2n+5 = O(n^2)$

$$
\begin{aligned}
10\,n^2 &= 3n^2 + 2n^2 + 5n^2 \\
&\geq 3n^2 + 2n + 5 \text{ for } n \geq 1
\end{aligned}
$$

$c = 10,\ n_0 = 1$

Classification of Function : BIG O (1/2)

- A function f(n) is said to be of at most logarithmic growth if $f(n) = O(\log n)$
- A function f(n) is said to be of at most quadratic growth if $f(n) = O(n^2)$
- A function f(n) is said to be of at most polynomial growth if $f(n) = O(n^k)$, for some natural number $k > 1$
- A function f(n) is said to be of at most exponential growth if there is a constant c, such that $f(n) = O(c^n)$, and $c > 1$
- A function f(n) is said to be of at most factorial growth if $f(n) = O(n!)$.

Classification of Function : BIG O (2/2)

- A function f(n) is said to have constant running time if the size of the input n has no effect on the running time of the algorithm (e.g., assignment of a value to a variable). The equation for this algorithm is $f(n) = c$
- Other logarithmic classifications:
  - $f(n) = O(n \log n)$
  - $f(n) = O(\log \log n)$

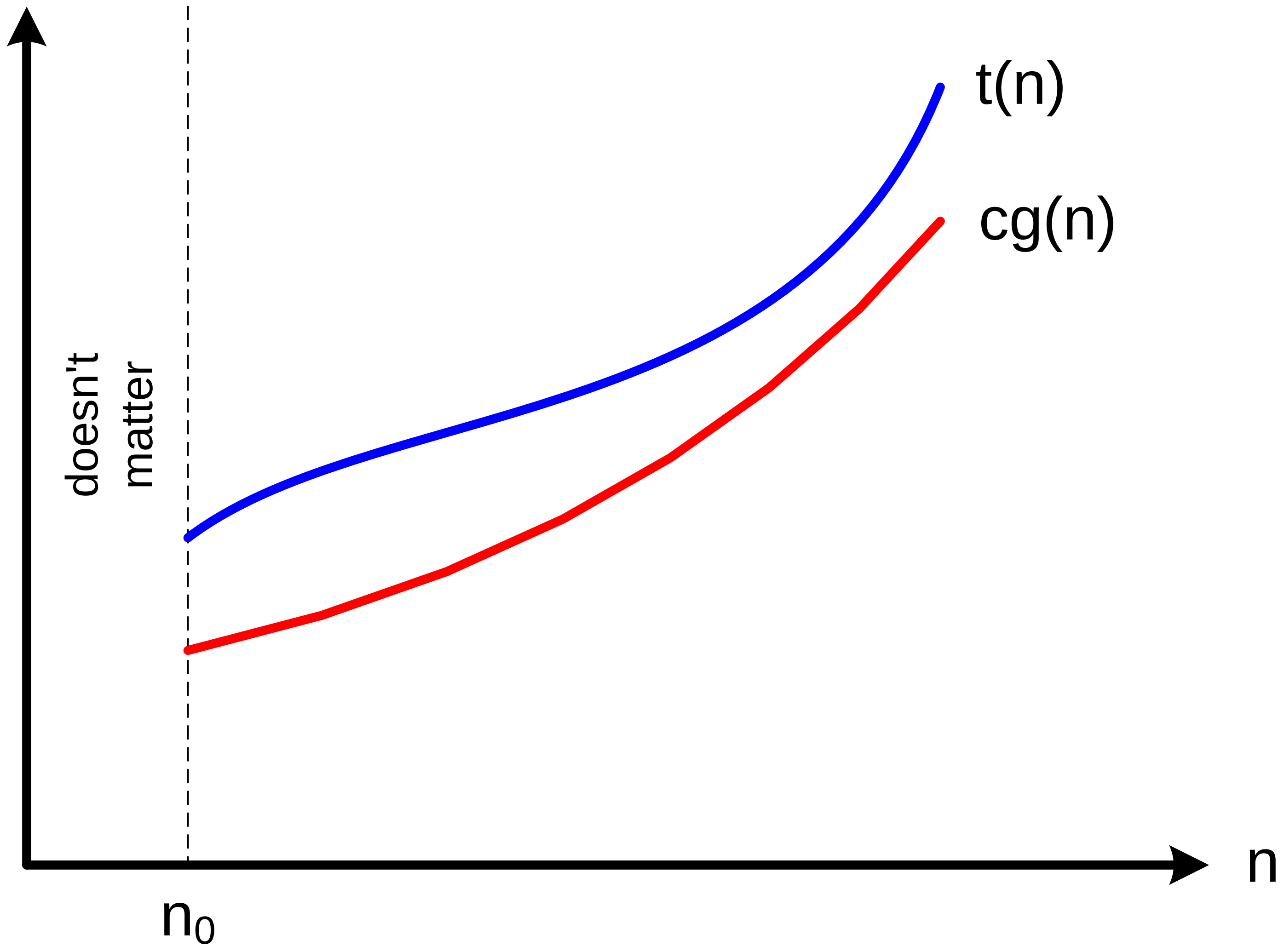
Ω-notation: Formally

- **DEF2**: A function t(n) is said to be in $\Omega(g(n))$, denoted $t(n) \in \Omega(g(n))$, if t(n) is bounded **below** by some constant multiple of g(n) for all large n

- i.e. there exist some positive constant c and some nonnegative integer n_0, such that

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

$t(n) \in \Omega(g(n))$: Illustration



Proving Example: $n^3 \in \Omega(n^2)$

- Remember DEF2: find c and n_0, such that $t(n) \geq cg(n)$ for all $n \geq n_0$

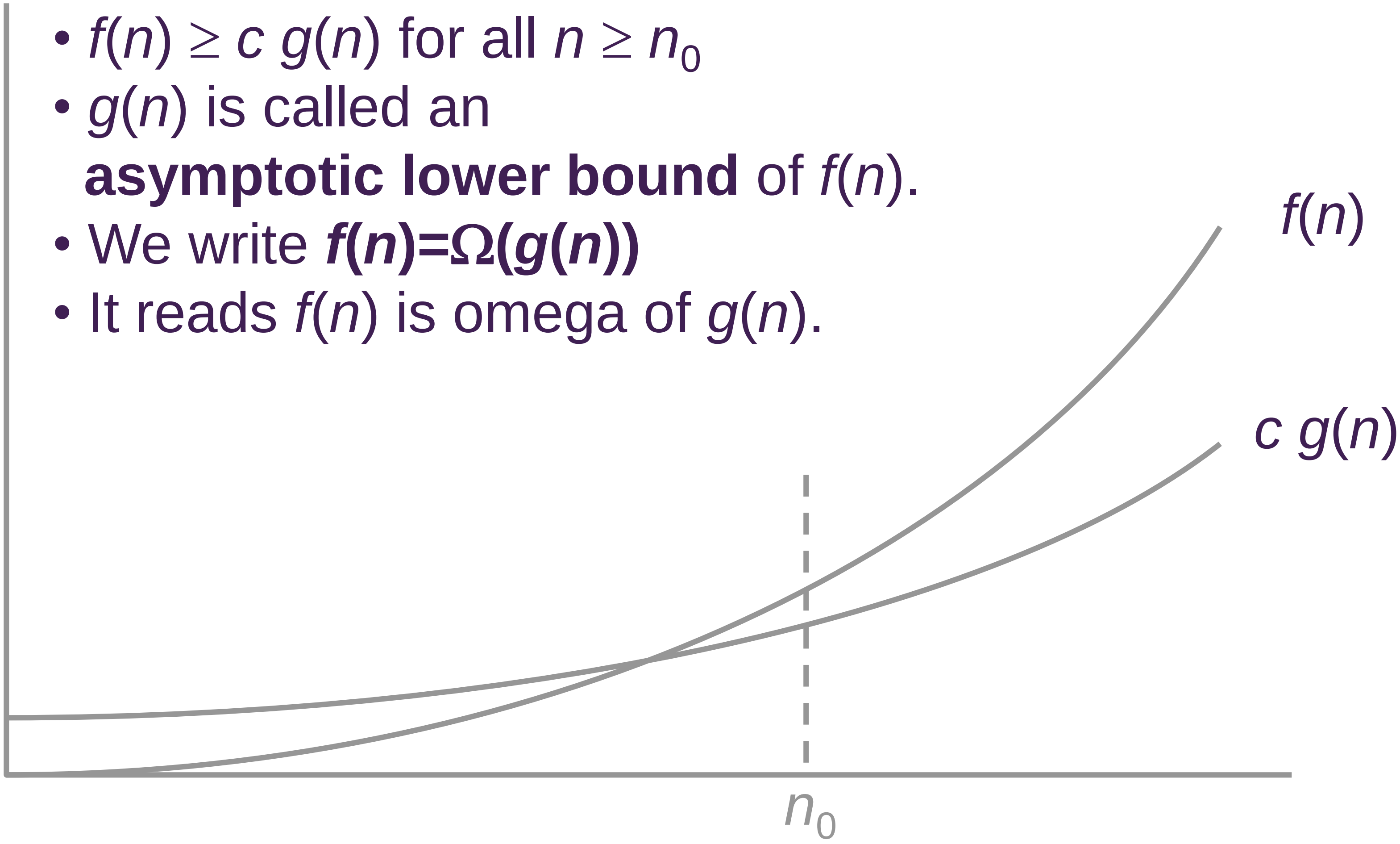
- $n^3 \geq n^2$ (for all $n \geq 0$) $\rightarrow$ $c=1$, $n_0=0$
- …

Lower Bound Notation

- We say InsertionSort's run time is $\Omega(n)$
- In general a function
  - $f(n)$ is $\Omega(g(n))$ if $\exists$ positive constants *c* and n_0 such that $0 \leq c \cdot g(n) \leq f(n) \;\; \forall\, n \geq n_0$
- Proof:
  - Suppose run time is $an + b$
  - $an \leq an + b$

Big Ω Asymptotic Lower Bound

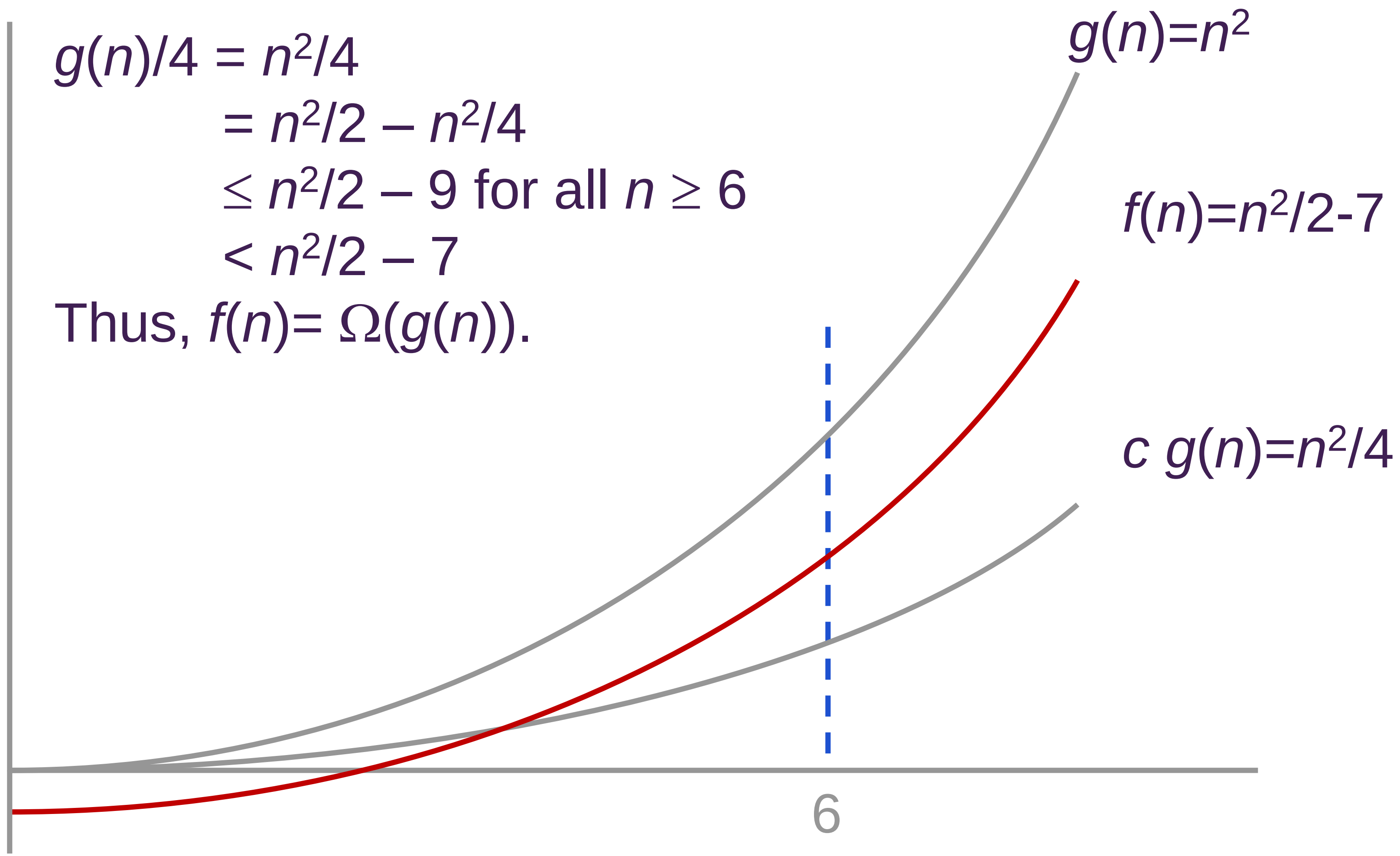
- $f(n) \geq c\, g(n)$ for all $n \geq n_0$
- $g(n)$ is called an **asymptotic lower bound** of $f(n)$.
- We write $\boldsymbol{f(n)=\Omega(g(n))}$
- It reads $f(n)$ is omega of $g(n)$.



Example of Asymptotic Lower Bound

$$
\begin{aligned}
g(n)/4 = n^2/4 & \\
&= n^2/2 - n^2/4 \\
&\le n^2/2 - 9 \text{ for all } n \ge 6 \\
&< n^2/2 - 7
\end{aligned}
$$

Thus, $f(n) = \Omega(g(n))$.



Example: Big Omega

- Example: $n^{1/2} = \Omega(\log n)$.

  Use the definition with $c = 1$ and $n_0 = 16$.

Checks OK.

Let $n \geq 16$: $n^{1/2} \geq (1) \log n$

if and only if $n = (\log n)^2$ by squaring both sides.

This is an example of polynomial vs. log.

Big Theta Notation

- Definition: Two functions f and g are said to be of equal growth, f = *Big Theta*(g) if and only if both

  $f = \Theta(g)$ and $g = \Theta(f)$.

- Definition: $f(n) = \Theta(g(n))$ means $\exists$ positive constants c_1, c_2, and n_0 such that

  $$c_1\, g(n) \leq f(n) \leq c_2\, g(n)\ \forall\ n \geq n_0$$

- If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ then $f(n) = \Theta(g(n))$

  (e.g. $f(n) = n^2$ and $g(n) = 2n^2$)

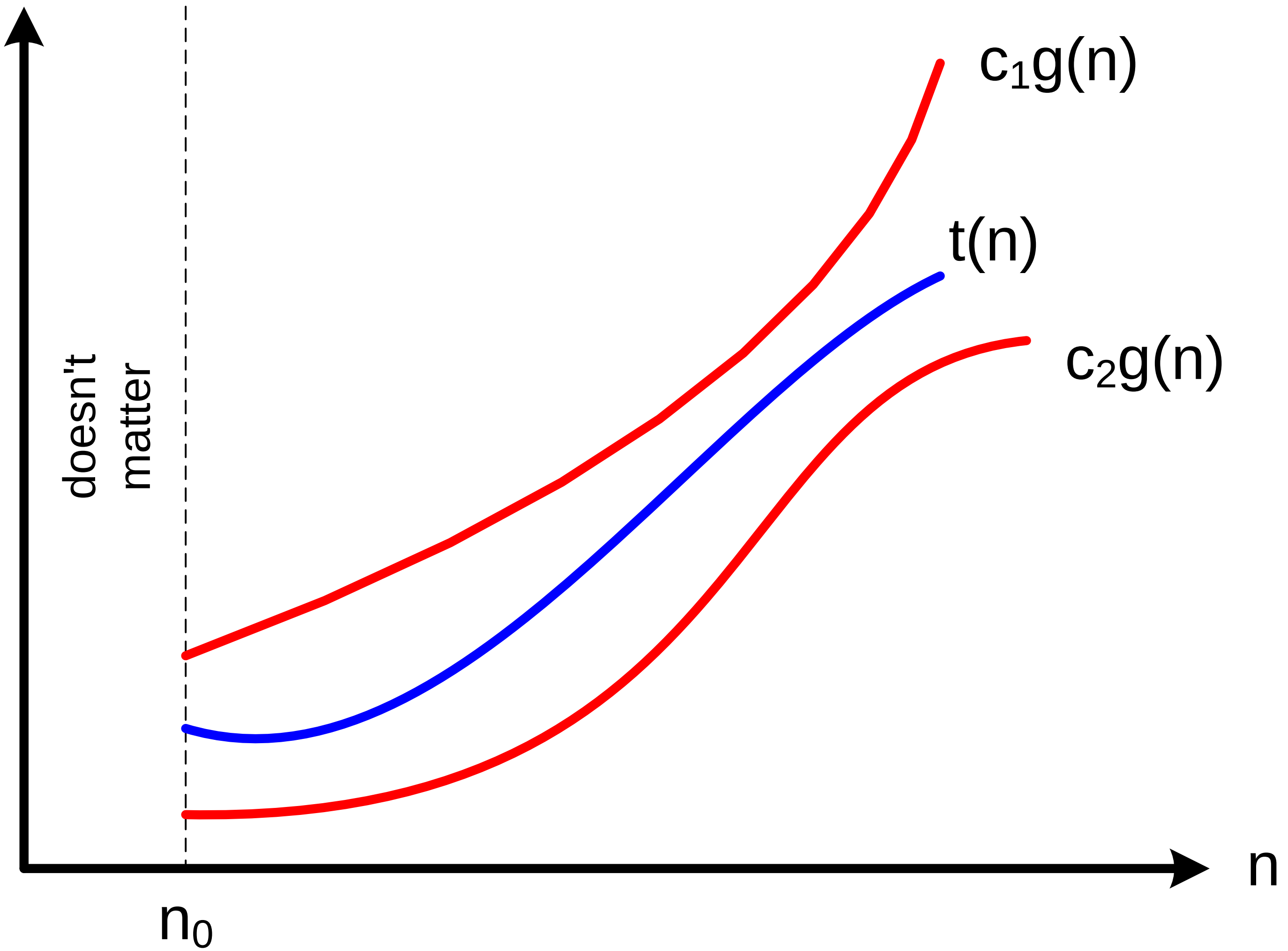
Θ-notation: Formally

- **DEF3**: A function t(n) is said to be in Θ(g(n)), denoted t(n) ∈ Θ(g(n)), if t(n) is bounded both **above** and **below** by some constant multiple of g(n) for all large n

- i.e there exist some positive constant c_1 and c_2 and some nonnegative integer n_0, such that

$$c_2 g(n) \le t(n) \le c_1 g(n) \text{ for all } n \ge n_0$$

$t(n) \in \Theta(g(n))$: Illustration



Proving Example: $\frac{1}{2}n(n-1) \in \Theta(n^2)$

- Remember DEF3: find c_1 and c_2 and some nonnegative integer n_0, such that
  $c_2 g(n) \le t(n) \le c_1 g(n)$ for all $n \ge n_0$

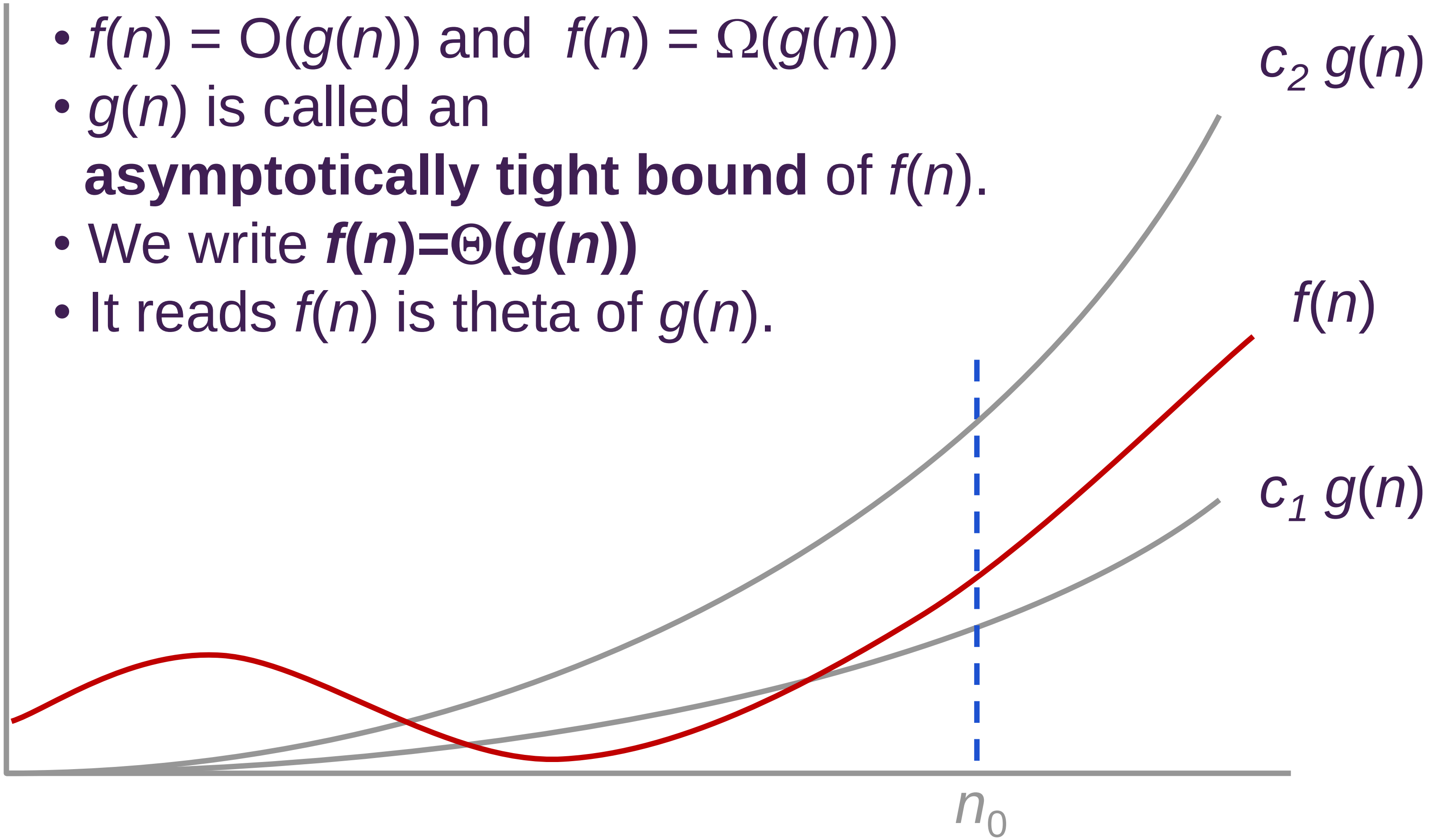
- The upper bound: $\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \le \frac{1}{2}n^2$ (for all $n \ge 0$)
- The lower bound: $\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \ge \frac{1}{2}n^2 - \frac{1}{2}n \frac{1}{2}n$ (for all $n \ge 2$) $= \frac{1}{4}n^2$
- $c_1 = \frac{1}{2}$, $c_2 = \frac{1}{4}$, $n_0 = 2$

Theta, the Asymptotic Tight Bound

- Theta means that f is bounded above and below by g; *BigTheta* implies the "best fit".
- f(n) does not have to be linear itself in order to be of linear growth; it just has to be between two linear functions,

Asymptotically Tight Bound

- $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$
- $g(n)$ is called an **asymptotically tight bound** of $f(n)$.
- We write $\boldsymbol{f(n)=\Theta(g(n))}$
- It reads $f(n)$ is theta of $g(n)$.



Other Asymptotic Notations

- A function f(n) is o(g(n)) if $\exists$ positive constants *c* and n_0 such that

  $f(n) < c\, g(n)\ \forall\, n \geq n_0$

- A function f(n) is ω(g(n)) if $\exists$ positive constants *c* and n_0 such that

  $c\, g(n) < f(n)\ \forall\, n \geq n_0$

- Intuitively,

  - o() is like <
  - O() is like $\leq$
  - ω() is like >
  - Ω() is like $\geq$
  - Θ() is like =

Examples

1. $2n^3 + 3n^2 + n \;=\; 2n^3 + 3n^2 + O(n)$

$$= 2n^3 + O(\,n^2 + n) = 2n^3 + O(\,n^2\,)$$

$$= O(n^3\,) = O(n^4)$$

2. $2n^3 + 3n^2 + n \;=\; 2n^3 + 3n^2 + O(n)$

$$= 2n^3 + \Theta(n^2 + n)$$

$$= 2n^3 + \Theta(n^2) = \Theta(n^3)$$

Example (cont.)

$n^3 = 50^3 * 729$

$n = \sqrt[3]{50^3 * 729}$

$n = \sqrt[3]{50^3}\sqrt[3]{729}$

$n = 50 * 9$

$n = 50 * 9 = 450$

$3^n = 3^{50} * 729$

$n = \log_3 (729 * 3^{50})$

$n = \log_3(729) + \log_3 3^{50}$

$n = 6 + \log_3 3^{50}$

$n = 6 + 50 = 56$

- Improvement: problem size increased by 9 times for n^3 algorithm but only a slight improvement in problem size (+6) for exponential algorithm.

More Examples

(a) $0.5n^2 - 5n + 2 = \Omega(n^2)$.

Let $c = 0.25$ and $n_0 = 25$.

$0.5 n^2 - 5n + 2 = 0.25(n^2)$ for all $n = 25$

(b) $0.5 n^2 - 5n + 2 = O(n^2)$.

Let $c = 0.5$ and $n_0 = 1$.

$0.5(n^2) = 0.5 n^2 - 5n + 2$ for all $n = 1$

(c) $0.5 n^2 - 5n + 2 = \Theta(n^2)$

from (a) and (b) above.

Use $n_0 = 25$, $c_1 = 0.25$, $c_2 = 0.5$ in the definition.

More Examples

(d) $6 * 2^n + n^2 = O(2^n)$.

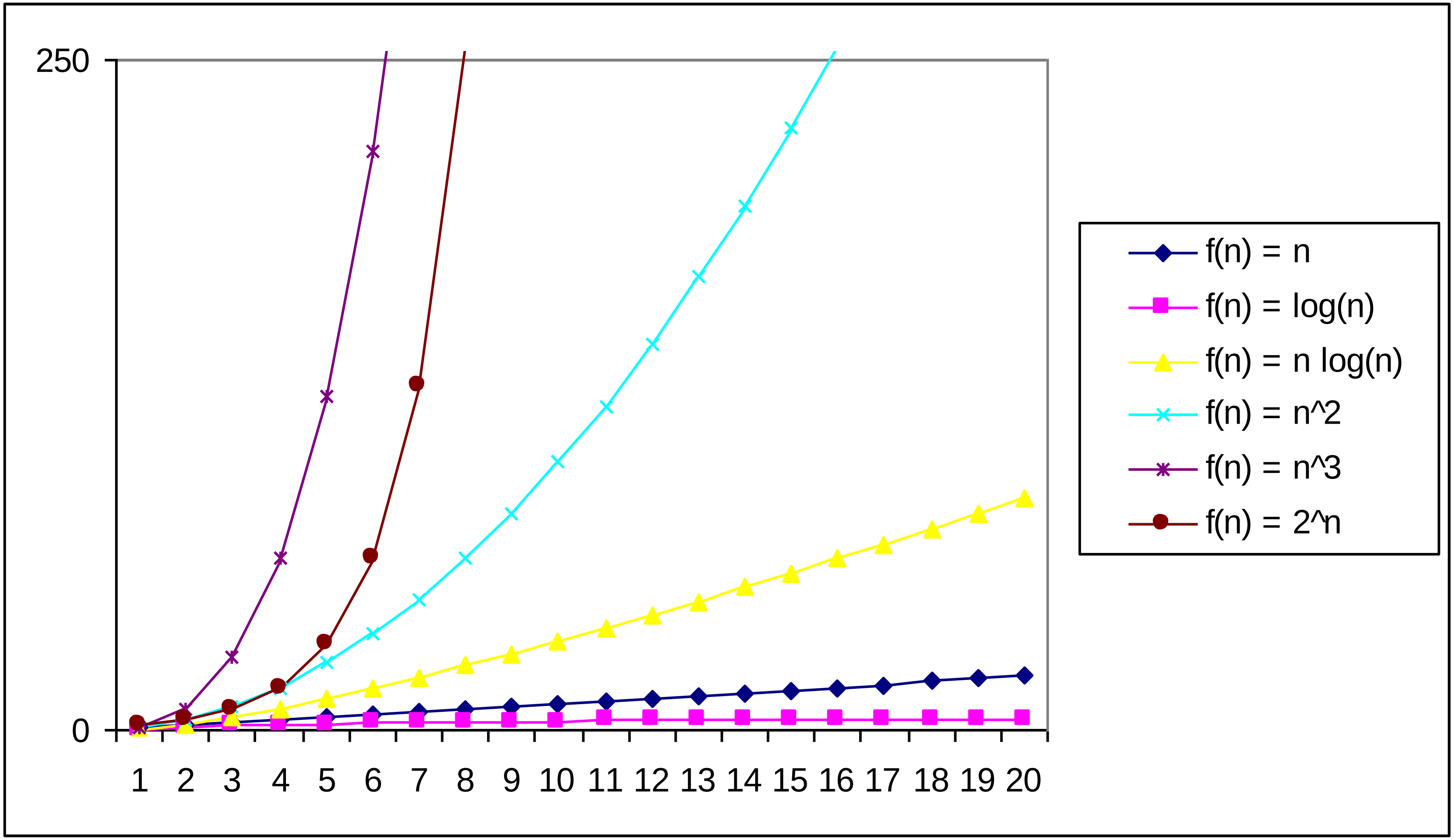
Let $c = 7$ and $n_0 = 4$.

Note that $2^n = n^2$ for $n = 4$. Not a tight upper bound, but it's true.

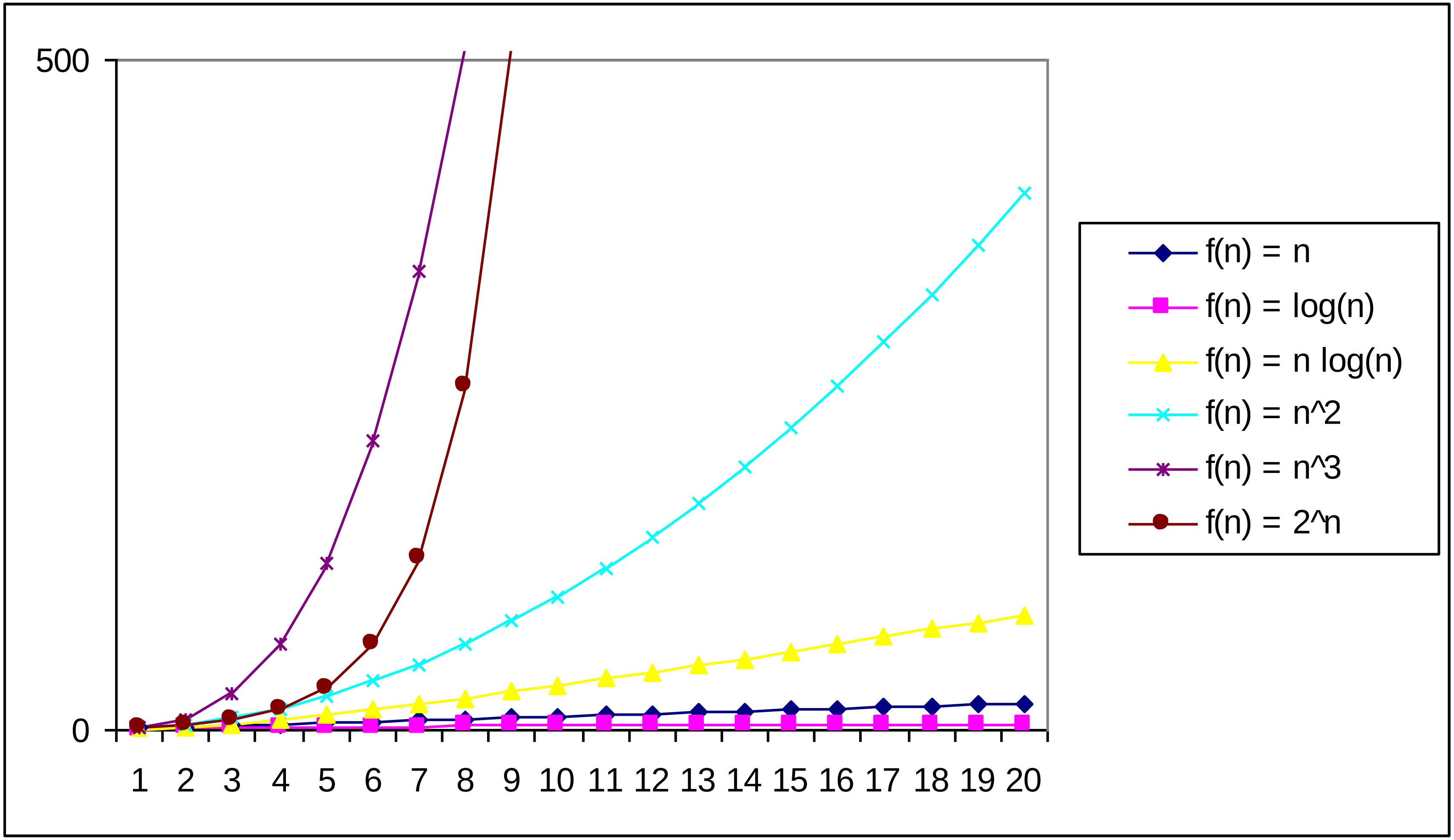
(e) $10\ n^2 + 2 = O(n^4)$.

There's nothing wrong with this, but usually we try to get the closest g(n). Better is to use $O(n^2)$.

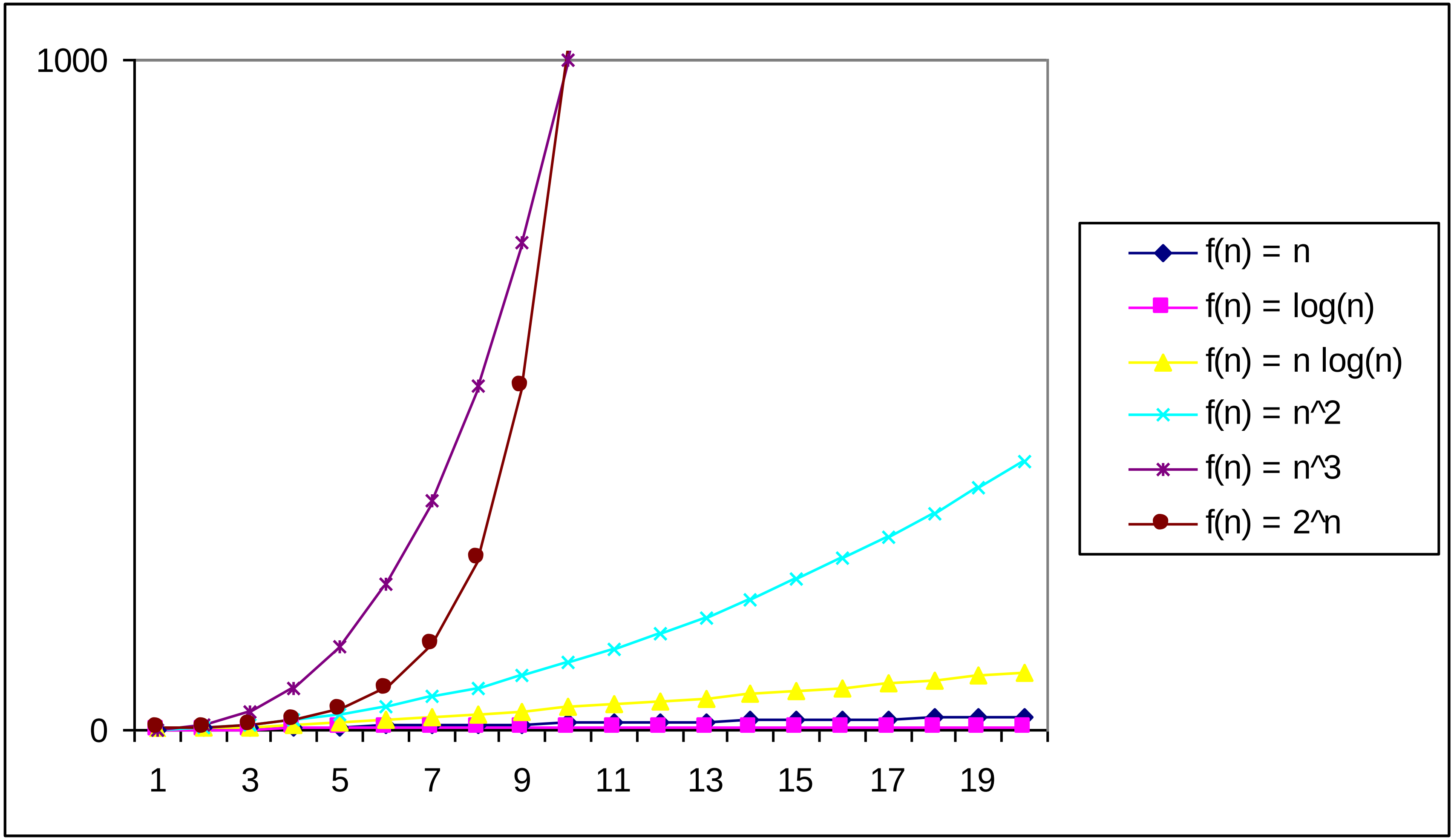
Practical Complexity t < 250



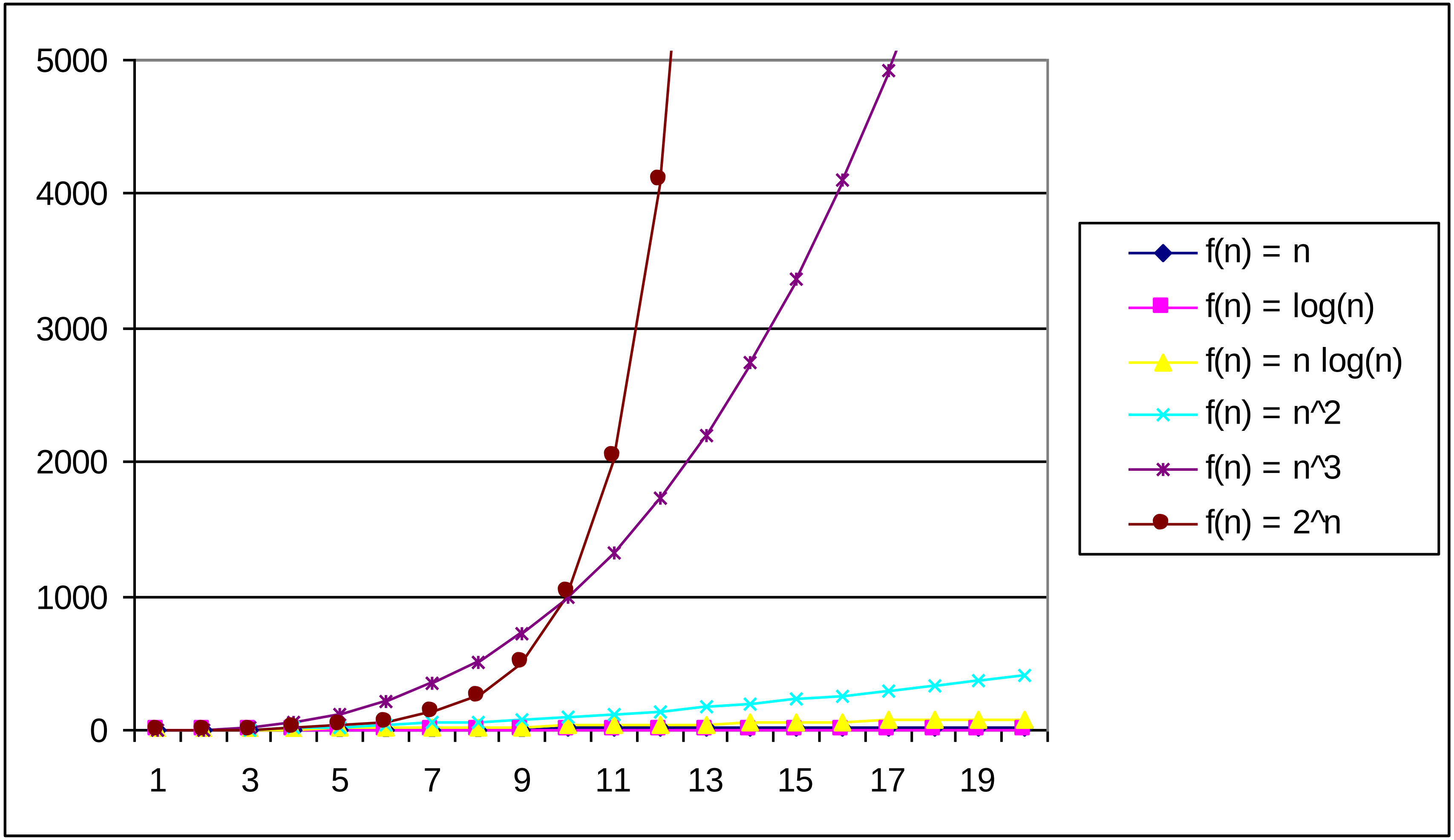
Practical Complexity t < 500



Practical Complexity t < 1000



Practical Complexity t < 5000



Tugas (1)

1. True or false:
   a. $n(n+1)/2 \in O(n^3)$
   b. $n(n+1)/2 \in O(n^2)$
   c. $n(n+1)/2 \in \Theta(n^3)$
   d. $n(n+1)/2 \in \Omega(n)$
2. Indicate the class $\Theta(g(n))$:
   a. $(n^2+1)^{10}$
   b. $(10n^2+7n+3)^{1/2}$
   c. $2n \log (n+2)^2+(n+2)^2 \log (n/2)$

Tugas 1 : O-notation

3. Tentukan OoG dari masing-masing soal

a. $f1(n) = 10\ n + 25\ n^2$
b. $f2(n) = 20\ n \log n + 5\ n$
c. $f3(n) = 12\ n \log n + 0.05\ n^2$
d. $f4(n) = n^{1/2} + 3\ n \log n$

- $O(n^2)$
- $O(n \log n)$
- $O(n^2)$
- $O(n \log n)$

4.. True/false ?

(a) $0.25n^2 - 5n + 2 = \Omega(n^2)$.
(b) $0.25n^2 - 5n + 2 = O(n^2)$.
(c) $0.25n^2 - 5n + 2 = \Theta(n^2)$.

Tugas Kelompok

1. Kerjakan soal di hal 29 no 2.2-1 sd. 2.2-4
2. Tugas 2 s.d Tugas 6 di slide ini
3. Pengumpulan :
   1. Tulis dikertas folio bergaris
   2. Dikumpulkan minggu depan di kelas
   3. KODE TUGAS :

      DAA_A_1_1 (MT DAA, kelas A, Kelompok1, tugas ke-1)

      DAA_D_5_1

Tugas 2

3-2 Relative asymptotic growths

Indicate, for each pair of expressions (A, B) in the table below, whether A is O, o, Ω, ω, or Θ of B. Assume that $k \geq 1$, $\epsilon > 0$, and $c > 1$ are constants. Your answer should be in the form of the table with "yes" or "no" written in each box.

	A	B	O	o	Ω	ω	Θ
a.	$\lg^k n$	n^ϵ					
b.	n^k	c^n					
c.	$\sqrt{n}$	$n^{\sin n}$					
d.	2^n	$2^{n/2}$					
e.	$n^{\lg c}$	$c^{\lg n}$					
f.	$\lg(n!)$	$\lg(n^n)$					

Tugas 3

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(\frac{3}{2})^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg^* n$	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2 \lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

Tugas 4

3-4 ***Asymptotic notation properties***

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.

a. $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.

b. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

c. $f(n) = O(g(n))$ implies $\lg(f(n)) = O(\lg(g(n)))$, where $\lg(g(n)) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n.

d. $f(n) = O(g(n))$ implies $2^{f(n)} = O\left(2^{g(n)}\right)$.

e. $f(n) = O\left((f(n))^2\right)$.

f. $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

g. $f(n) = \Theta(f(n/2))$.

h. $f(n) + o(f(n)) = \Theta(f(n))$.

Tugas 5

5. Prove that every polynomial

   $p(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_0$ with $a_k > 0$

   belongs to $\Theta(n^k)$

6. Prove that exponential functions a^n have different orders of growth for different values of base $a > 0$

Tugas 6: Examples (cont.)

7. Suppose a program P is $O(n^3)$, and a program Q is $O(3^n)$, and that currently both can solve problems of size 50 in 1 hour. If the programs are run on another system that executes exactly 729 times as fast as the original system, what size problems will they be able to solve?

Classifying functions by their Asymptotic Growth Rates (1/2)

- asymptotic growth rate, asymptotic order, or order of functions
  - Comparing and classifying functions that ignores *constant factors* and *small inputs*.

- O(g(n)), Big-Oh of g of n, the Asymptotic Upper Bound;
- Ω(g(n)), Omega of g of n, the Asymptotic Lower Bound.
- Θ(g(n)), Theta of g of n, the Asymptotic Tight Bound; and

Example

- Example: $f(n) = n^2 - 5n + 13$.
- The constant 13 doesn't change as n grows, so it is not crucial. The low order term, -5n, doesn't have much effect on f compared to the quadratic term, n^2.

  We will show that $f(n) = \Theta(n^2)$.

- Q: What does it mean to say $f(n) = \Theta(g(n))$?
- A: Intuitively, it means that function f is the same order of magnitude as g.

Example (cont.)

- Q: What does it mean to say $f_1(n) = \Theta(1)$?
- A: $f_1(n) = \Theta(1)$ means after a few n, f_1 is bounded above & below by a constant.
- Q: What does it mean to say $f_2(n) = \Theta(n \log n)$?
- A: $f_2(n) = \Theta(n \log n)$ means that after a few n, f_2 is bounded above and below by a constant times n log n. In other words, f_2 is the same order of magnitude as n log n.
- More generally, $f(n) = \Theta(g(n))$ means that f(n) is a member of $\Theta(g(n))$ where $\Theta(g(n))$ is a set of functions of the same order of magnitude.

Useful Property

- **Theorem**:

  If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

- The analogous assertions are true for the Ω and Θ notations as well

Example

- Alg to check whether an array has identical elements:
  1. Sort the array
  2. Scan the sorted array to check its consecutive elements for equality
- (1) = ≤ ½n(n-1) comparison → $O(n^2)$
- (2) = ≤ n-1 comparison → $O(n)$
- The efficiency of (1)+(2) = $O(\max\{n^2,n\}) = O(n^2)$

Using Limits for Comparing OoG

- A ‘convenient’ method for comparing order of growth of two specific functions
- Three principal cases:

$$\lim_{n\to\infty}\frac{t(n)}{g(n)}\begin{cases}0 & \text{implies that } t(n) \text{ has a smaller OoG than } g(n)\\ c & \text{implies that } t(n) \text{ has the same OoG as } g(n)\\ \infty & \text{implies that } t(n) \text{ has a larger OoG than } g(n)\end{cases}$$

- The first two cases → $t(n) \in O(g(n))$; the last two cases → $t(n) \in \Omega(g(n))$; the second case alone → $t(n) \in \Theta(g(n))$

Limit-based: why convenient?

- It can take advantage of the powerful calculus techniques developed for computing limits, such as
  - L'Hopital's rule

$$\lim_{n\to\infty}\frac{t(n)}{g(n)}=\lim_{n\to\infty}\frac{t'(n)}{g'(n)}$$

  - Stirling's formula

$$n!\approx\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \text{ for large value of n}$$

Example (1)

- Compare OoG of ½n(n-1) and n^2.

$$\lim_{n\to\infty}\frac{\frac{1}{2}n(n-1)}{n^2}=\frac{1}{2}\lim_{n\to\infty}\frac{n^2-n}{n^2}=\frac{1}{2}\lim_{n\to\infty}\left(1-\tfrac{1}{n}\right)=\frac{1}{2}$$

  - The limit = c → ½n(n-1) ∈ Θ (n^2)

- Compare OoG of $\log_2 n$ and √n

$$\lim_{n\to\infty}\frac{\log_2 n}{\sqrt{n}}=\lim_{n\to\infty}\frac{(\log_2 n)'}{(\sqrt{n})'}=\lim_{n\to\infty}\frac{(\log_2 e)\frac{1}{n}}{\frac{1}{2\sqrt{n}}}=2\log_2 e\lim_{n\to\infty}\frac{\sqrt{n}}{n}=0$$

  - The limit = 0 → $\log_2 n$ has smaller order of √n

Example (2)

- Compare OoG of n! and 2^n.

$$\lim_{n\to\infty}\frac{n!}{2^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n}=\lim_{n\to\infty}\sqrt{2\pi n}\,\frac{n^n}{2^n e^n}=\lim_{n\to\infty}\sqrt{2\pi n}\left(\frac{n}{2e}\right)^n=\infty$$

  - The limit = ∞ → $n! \in \Omega(2^n)$

Review Tugas n!

- Menghitung kompleksitas pada Faktorial

```
Function Faktorial (input n : integer) → integer
{menghasilkan nilai n!, n ≥ 0}
Algoritma
   If n=0 then
        Return 1
   Else
        Return n*faktorial (n-1)
Endif
```

- Kompleksitas waktu :
  - untuk kasus basis, tidak ada operasi perkalian → (0)
  - untuk kasus rekurens, kompleksitas waktu diukur dari jumlah perkalian (1) ditambah kompleksitas waktu untuk faktorial (n-1)

$$T(n) = \begin{cases} 0 & , n = 0 \\ T(n-1)+1 & , n > 0 \end{cases}$$

Review Tugas n! (Lanjutan)

Kompleksitas waktu n! :

$T(n)=1+T(n-1)$

$=T(n)=1+1+T(n-2)=2+T(n-2)$

$=T(n)=2+1+T(n-3)=3+T(n-3)$

$= \ldots$

$= \ldots$

$= n+T(0)$

$= n + 0$

Jadi $T(n) = n$

$T(n) \in O(n)$

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

Click to edit subtitle style

Thank You !