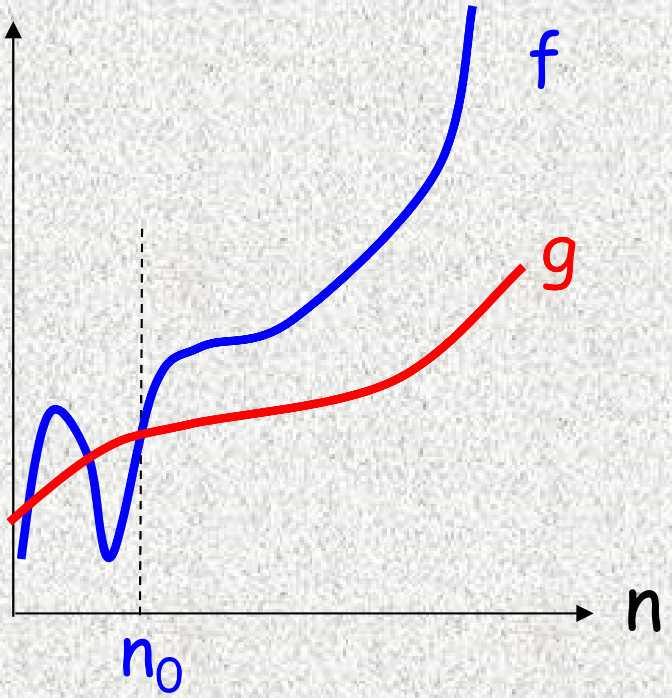


$f(n) = 10n^3 - 43n^2 + n - 100$

$\Rightarrow f(n) = \theta(n^3)$ 成長率大

$g(n) = 2n^2 + 1000000n$

$\Rightarrow g(n) = \theta(n^2)$ 成長率小

the rate of growth of $g(n)$ is n^2

when $n \to \infty$ (sufficiently large)

* 成長率 (rate of growth) 指的是 "最大項"

* f 的成長率比 g 大表示

"$f(n) \geq g(n)$ for $n \geq n_0$"

when n is sufficiently large

* When comparing the complexities of algorithms,
we compare their rates of growth.
"最 大 項"

* "algo. A is better than algo B" means
"A is faster when n is sufficiently large"

insertion sort: $O(n^2)$ merge sort: $O(n\lg n)$

faster for small n better

* Simple

* Constants depend on
-- hardware
-- programming skills

f, g: two functions (假設可以調整係數, when $n \to \infty$) 3-1b

* 如果 f, g 的最大項不一樣, 分出勝負

⟺ 調整係數無法改變大小關係

$f(n) = n^3 + n^2 - 100n$ 大

$g(n) = 30n^2 + 100n$ 小

⇒

$10^{-6} \times f(n)$ 大

$10^{6} \times g(n)$ 小

* 如果 f, g 的最大項一樣, 由最大項係數決定

⟺ 調整係數可以改變大小關係

$f(n) = 4n^3 + n^2 - 100n$ 大

$g(n) = 3n^3 + 5n^2 + 100n$ 小

⇒

$f(n)$ 小

$10^{6} \times g(n)$ 大

f, g: two functions

調整 g 係數可讓 $g \le f$ 也可讓 $g \ge f$

⇨ f 的最大項"="g 的最大項

⇨ f 的成長率"="g 的成長率

調整 g 係數可讓 $f \le g$

⇨ f 的最大項"≤"g 的最大項

⇨ f 的成長率"≤"g 的成長率

調整 g 係數可讓 $g \le f$

⇨ f 的最大項"≥"g 的最大項

⇨ f 的成長率"≥"g 的成長率

method 1. try n_0 = 1, 2, 3, ...

$c_1 \le 3 - 6/n \le c_2$ for $n \ge n_0$

	n_0	n_0	n_0						
n	1	2	3	4	5	6	7	8	• • •
3-6/n	-3	0	1	1.5	1.8	2	2.14	2.25	• • •

c_1 ~$c_1 c_2$~$c_1 c_2$~ c_2

try

- if n_0 = 1, positive c_1 and c_2 exist ?
- if n_0 = 2, positive c_1 and c_2 exist ?
- if n_0 = 3, positive c_1 and c_2 exist ?

• • •

method 2. choose c_1, c_2 and then find n_0

n	1	2	3	4	5	6	7	8	• • •
3-6/n	-3	0	1	1.5	1.8	2	2.14	2.25	• • •

$$c_1 n^2 \le 3n^2 - 6n \le c_2 n^2$$

- choose $c_1 = 2$ and $c_2 = 3$ -----> $n_0 = 6$
- choose $c_1 = 0.5$ and $c_2 = 4$ -----> $n_0 = ???$

• • •

f, g: two functions

調整 g 係數可讓 $g \le f$ 也可讓 $g \ge f$

⇨ f 的最大項 "=" g 的最大項

⇨ f 的成長率 "=" g 的成長率 ⇨ $f = \theta(g)$

調整 g 係數可讓 $f \le g$

⇨ f 的最大項 "$\le$" g 的最大項

⇨ f 的成長率 "$\le$" g 的成長率 ⇨ $f = O(g)$

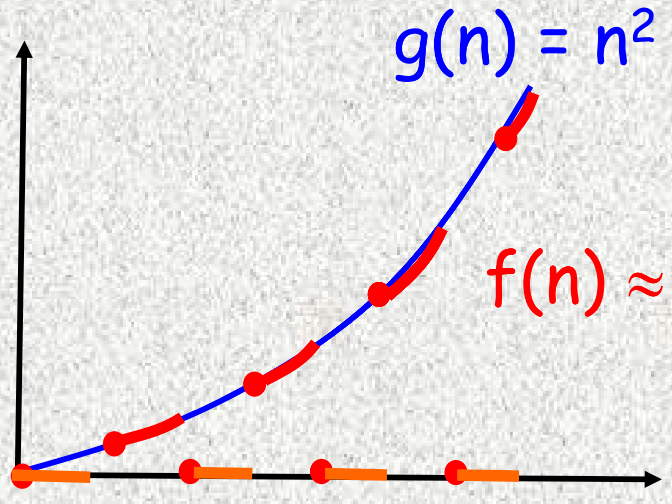
調整 g 係數可讓 $g \le f$

⇨ f 的最大項 "$\ge$" g 的最大項

⇨ f 的成長率 "$\ge$" g 的成長率 ⇨ $f = \Omega(g)$

$o(g(n)) = O(g(n)) \setminus \Theta(g(n))$???

Mathematically, no!



$f = O(g)$, but not o or Θ

$f(n) \approx 0$ if "n is even" and is n^2 otherwise

Algorithmically, yes!

Because, it is reasonable to assume that $T(n)$ is "regular" for large n.

functions:		ω	Ω	Θ	O	o
real numbers:		$>$	$\geq$	$=$	$\leq$	$<$

Transitivity	$a \leq b,\ b \leq c \Rightarrow a \leq c$	$f(n) = O(g(n)),\ g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$
Reflexivity	$a = a$	$f(n) = \Theta(f(n))$
Symmetry	$a = b \Rightarrow b = a$	$f(n) = \Theta(g(n) \Rightarrow g(n) = \Theta(f(n))$
Transpose Symmetry	$a \leq b \Rightarrow b \geq a$	$f(n) = O(g(n) \Rightarrow g(n) = \Omega(f(n))$
	$a > b \Rightarrow b < a$	$f(n) = \omega(g(n) \Rightarrow g(n) = o(f(n))$

time complexities

constant $\quad n^{0.000000000001} \quad$ linear

$$\underbrace{1 \quad \lg^* n \quad \lg\lg n \quad \lg_2 n \quad \sqrt{n} \quad n \quad n^2 \quad n^3 \quad \cdots \quad n^k}_{\text{polynomial}} \quad \underbrace{2^n \quad \cdots}_{\text{exponential}}$$

$$\underbrace{g(n) \times \overset{}{a(n)}_{\text{大}}}_{\text{大}} \qquad \underbrace{g(n) \times b(n)_{\text{小}}}_{\text{小}}$$

$$\underbrace{\cancel{n^2} n^{0.01}_{\text{大}}}_{\text{大}} \qquad \underbrace{\cancel{n^2}\lg n_{\text{小}}}_{\text{小}}$$

All logarithms are base-2 in CS

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \quad (|x|<1)$$

等 比 會 收 斂 ！

$$n + (1/2)n + (1/2)^2 n + (1/2)^3 n + \ \ldots + (1/2)^k n$$
$$= n \times (1 + (1/2) + (1/2)^2 + (1/2)^3 + \ \ldots + (1/2)^k)$$
$$\leq n \times 2$$
$$= O(n)$$

$$n^2 + (2/3)n^2 + (2/3)^2 n^2 + (2/3)^3 n^2 + \ \ldots + (2/3)^k n^2$$
$$= n^2 \times (1 + (2/3) + (2/3)^2 + (2/3)^3 + \ \ldots + (2/3)^k)$$
$$\leq n^2 \times 3$$
$$= O(n^2)$$