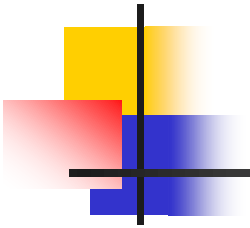
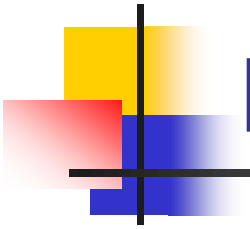


[Chapter 1] (part2)

Algorithms : Efficiency, Analysis, And Order



Representative Order Functions

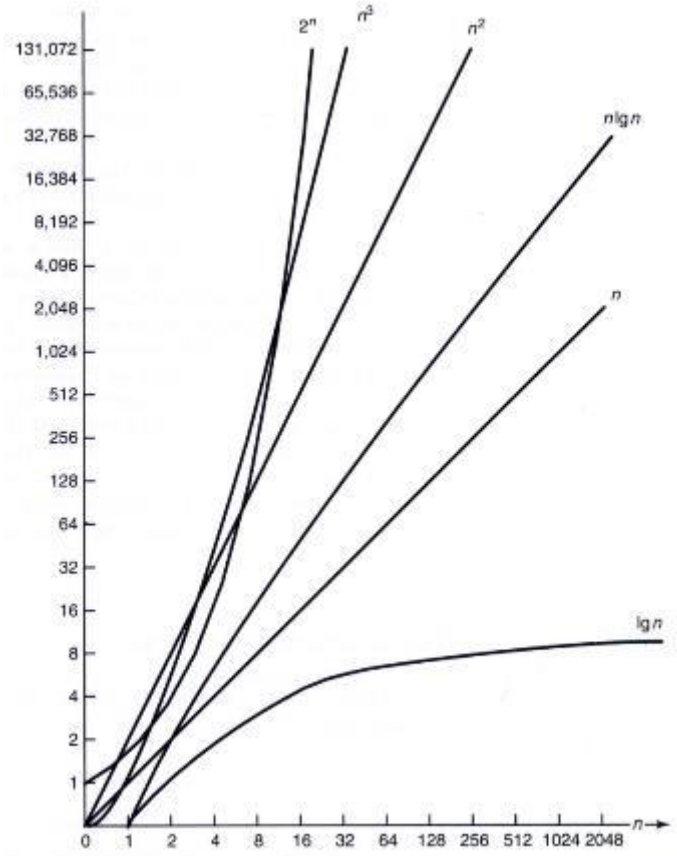
- $\Theta(\lg n)$
- $\Theta(n)$: linear
- $\Theta(n \lg n)$
- $\Theta(n^2)$: quadratic
- $\Theta(n^3)$: cubic
- $\Theta(2^n)$: exponential
- $\Theta(n!)$: combinatorial

Example

- The quadratic term eventually determines

n	$0.1n^2$	$0.1n^2 + n + 100$
10	10	120
20	40	160
50	250	400
100	1,000	1,200
1,000	100,000	101,100

Growth Rates of Some Complexity Functions



Execution Times for Algorithms with the Given Time Complexities

n	$f(n) = lg\, n$	$f(n) = n$	$f(n) = n\, lg\, n$	$f(n) = n^2$	$f(n) = n^3$	$f(n) = 2^n$
10	0.003 μs*	0.01 μs	0.033 μs	0.10 μs	1.0 μs	1 μs
20	0.004 μs	0.02 μs	0.086 μs	0.40 μs	8.0 μs	1 ms†
30	0.005 μs	0.03 μs	0.147 μs	0.90 μs	27.0 μs	1 s
40	0.005 μs	0.04 μs	0.213 μs	1.60 μs	64.0 μs	18.3 min
50	0.006 μs	0.05 μs	0.282 μs	2.50 μs	125.0 μs	13 days
10^2	0.007 μs	0.10 μs	0.664 μs	10.00 μs	1.0 ms	4×10^{13} years
10^3	0.010 μs	1.00 μs	9.966 μs	1.00 ms	1.0 s	
10^4	0.013 μs	10.00 μs	130.000 μs	100.00 ms	16.7 min	
10^5	0.017 μs	0.10 ms	1.670 ms	10.00 s	11.6 days	
10^6	0.020 μs	1.00 ms	19.930 ms	16.70 min	31.7 years	
10^7	0.023 μs	0.01 s	2.660 s	1.16 days	31,709 years	
10^8	0.027 μs	0.10 s	2.660 s	115.70 days	3.17×10^7 years	
10^9	0.030 μs	1.00 s	29.900 s	31.70 years		

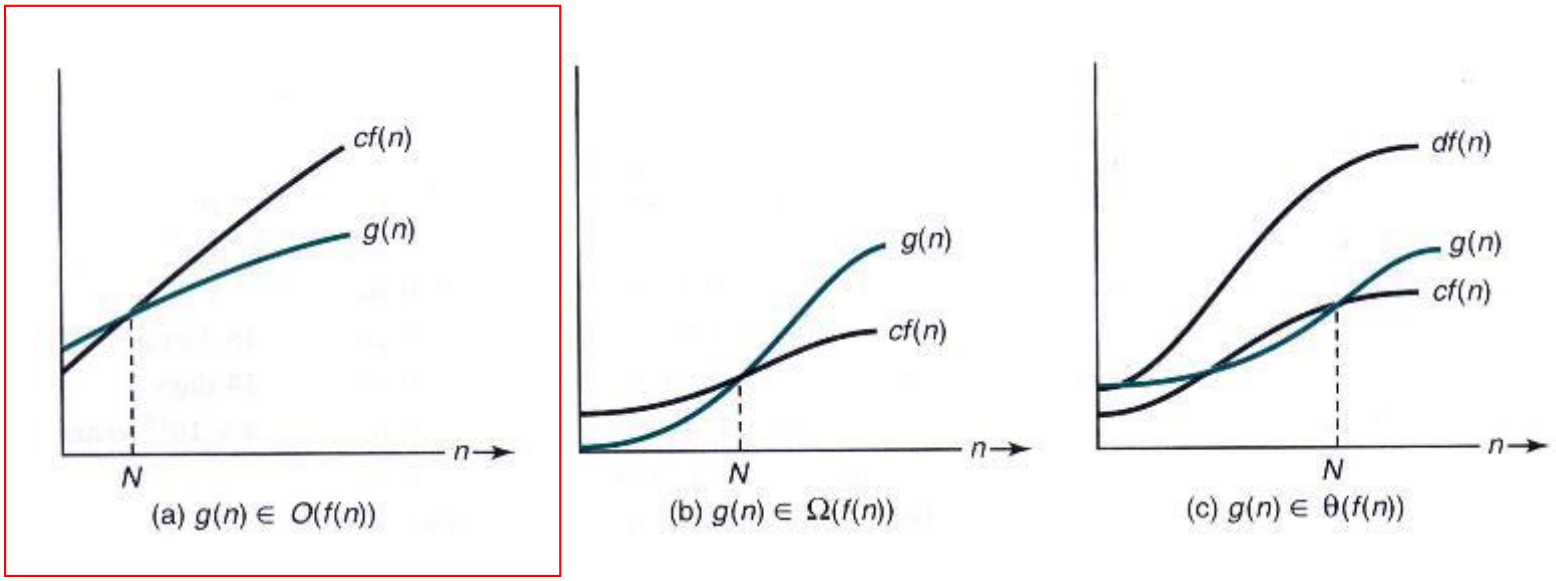
Rigorous Definition to Order: Big O

- Definition: (Asymptotic Upper Bound)
  - For a given complexity function $f(n)$, $O(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constant c and some non-negative integer N such that for all $n \geq N$,

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

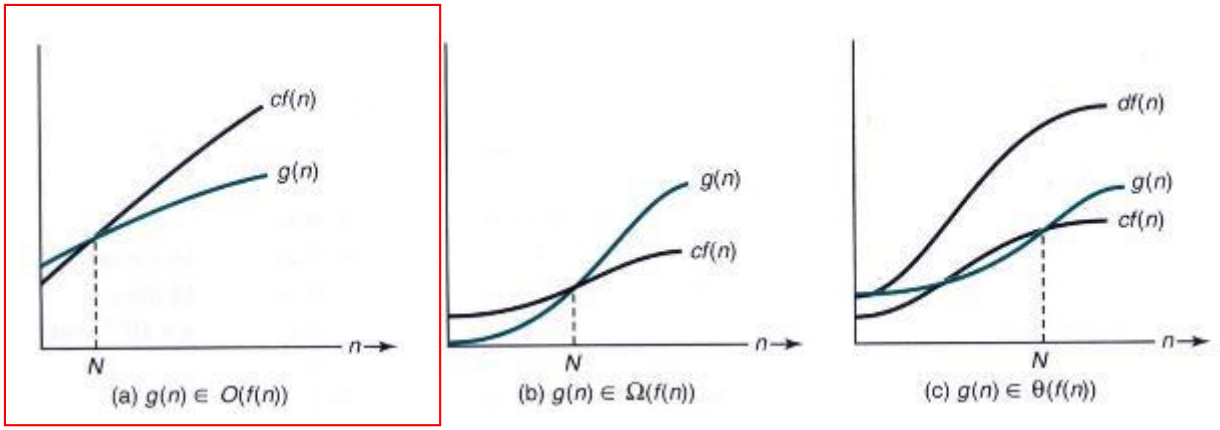
  - $g(n) \in O(f(n))$

Illustrating "big O", Ω, and Θ



Big O Notation: Definition

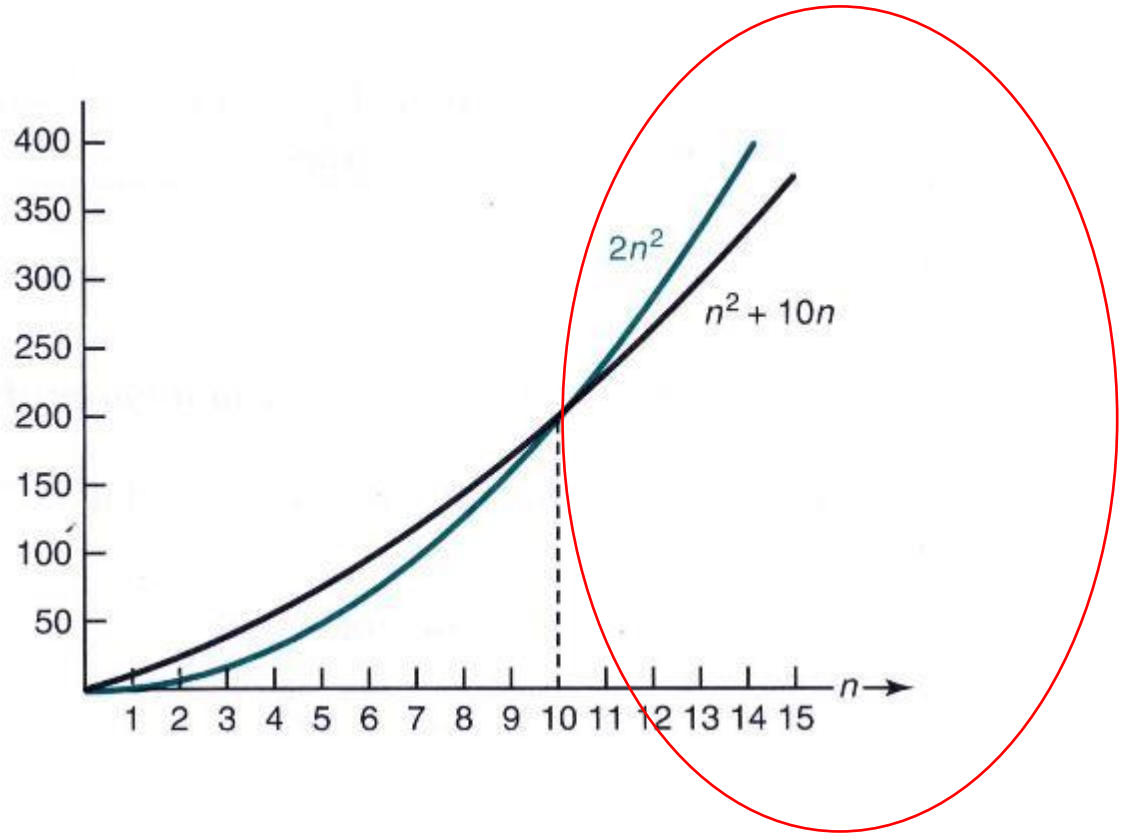
- Meaning of $g(n) \in O(f(n))$
  - Although *g(n)* starts out above *cf(n)* in the figure, eventually it falls beneath *cf(n)* and stays there.
  - If *g(n)* is the time complexity for an algorithm, eventually the running time of the algorithm will be at least as good as *f(n)*
  - *f(n)* is called as an asymptotic upper bound (*of what?*) (i.e. *g(n)* cannot run slower than *f(n)*, eventually)



Big O Notation: Example

- Meaning of $n^2+10n \in O(n^2)$
  - Take *c* = *11* and *N* = *1*.
  - Take *c* = *2* and *N* = *10*.
  - If n^2+10n is the time complexity for some algorithm, eventually the running time of the algorithm will be at least as fast (good) as n^2
  - $11n^2$ is an asymptotic upper bound for the time complexity function of n^2+10n.

Figure 1.5 The function $n^2 + 10n$ eventually stays beneath the function $2n^2$.



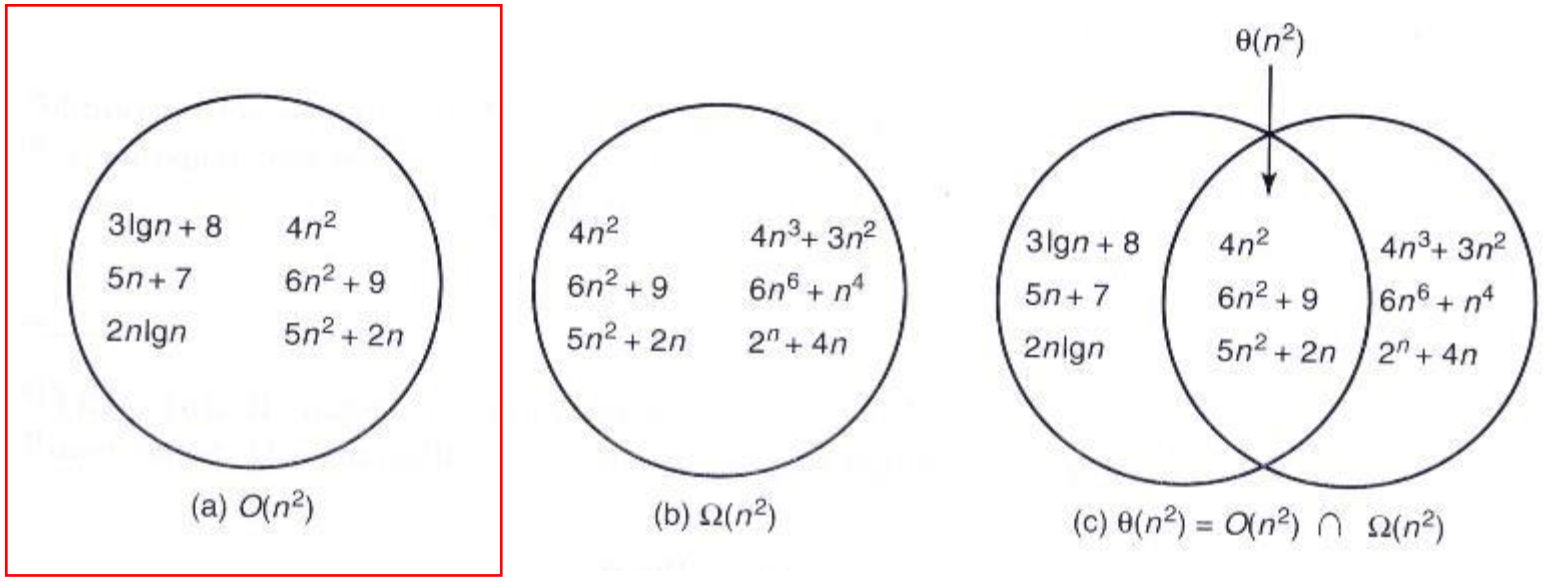
Big O Notation: More Examples

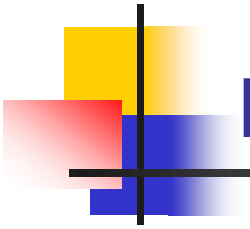
- $5n^2 \in O(n^2)$
  - Take *c* = *5* and *N* = *0*, then for all n such that $n \geq N$, $5n^2 \leq cn^2$.
- $T(n) = \frac{n(n-1)}{2}$
  - Because, for $n \geq 0$, $\frac{n(n-1)}{2} \leq \frac{n^2}{2}$
  - Therefore, we can take *c* = ½ and *N* = *0*, to conclude that $T(n) \in O(n^2)$.
- $n^2 \in O(n^2+10n)$
  - Because, for $n \geq 0$, $n^2 \leq 1 \times (n^2+10n)$
  - Therefore, we can take *c* = *1* and *N* = *0*, to conclude that $n^2 \in O(n^2+10n)$

Big O Notation: More Examples (Cont'd)

- $n \in O(n^2)$
  - Take *c* = *1* and *N* = *1*, then for all n such that $n \geq N$, $n \leq 1 \times n^2$.
- $n^3 \in O(n^2)$?
  - Divide both sides by n^2
  - Then, we can obtain $n \leq c$
  - But it's impossible there exists a constant *c* that is large enough than a variable *n*.
  - Therefore, n^3 does not belong to $O(n^2)$.

Figure 1.6 The sets $O(n^2)$, $\Omega(n^2)$, $\Theta(n^2)$. Some exemplary members are shown.





Rigorous Definition to Order: Ω

- Definition: (Asymptotic Lower Bound)
  - For a given complexity function $f(n)$, $\Omega(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constant c and some non-negative integer N such that for all $n \geq N$,

$$g(n) \geq c \times f(n)$$

  - $g(n) \in \Omega(f(n))$