[Chapter 1] (part2)

Algorithms :

Efficiency, Analysis,

And Order

Representative Order Functions

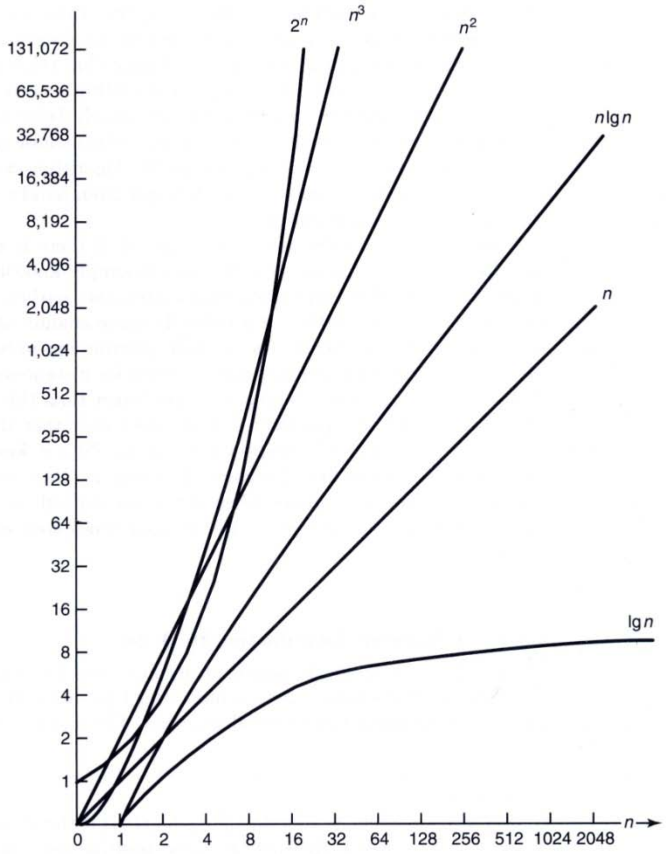
- $\Theta(\lg n)$
- $\Theta(n)$: linear
- $\Theta(n \lg n)$
- $\Theta(n^2)$: quadratic
- $\Theta(n^3)$: cubic
- $\Theta(2^n)$: exponential
- $\Theta(n!)$: combinatorial

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		

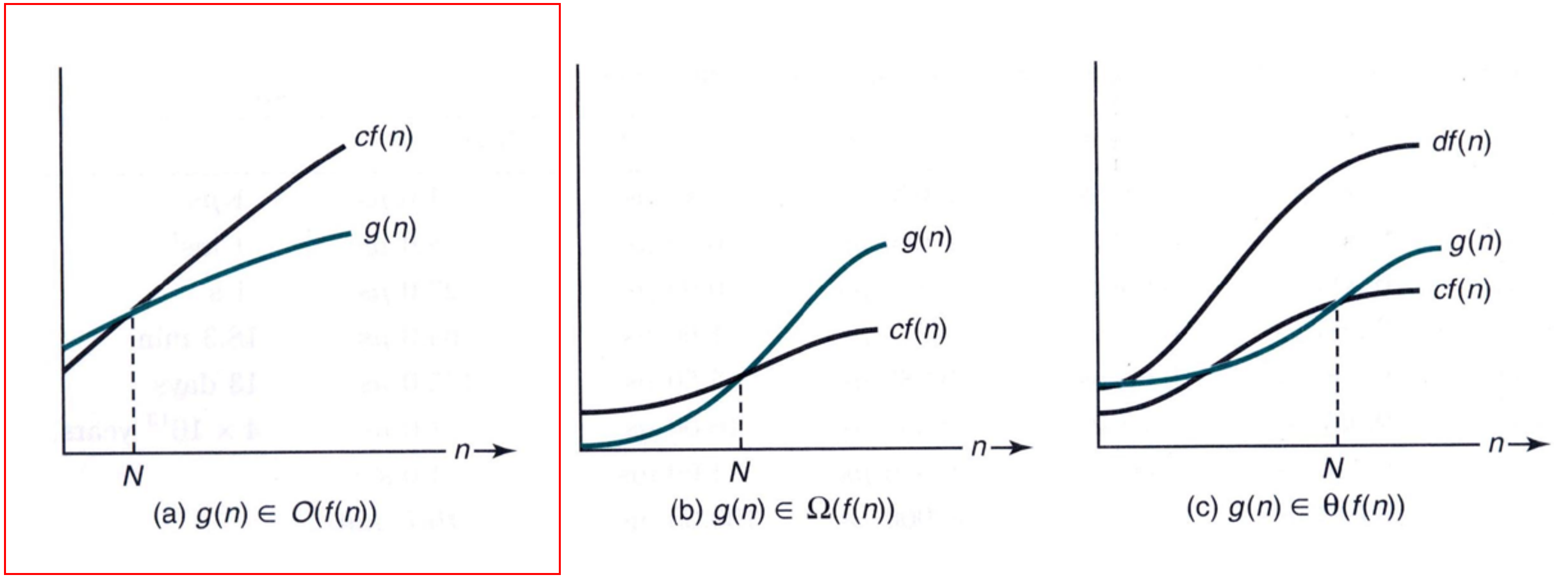
Rigorous Definition to Order: Big O

- 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 $n \geq N$,

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

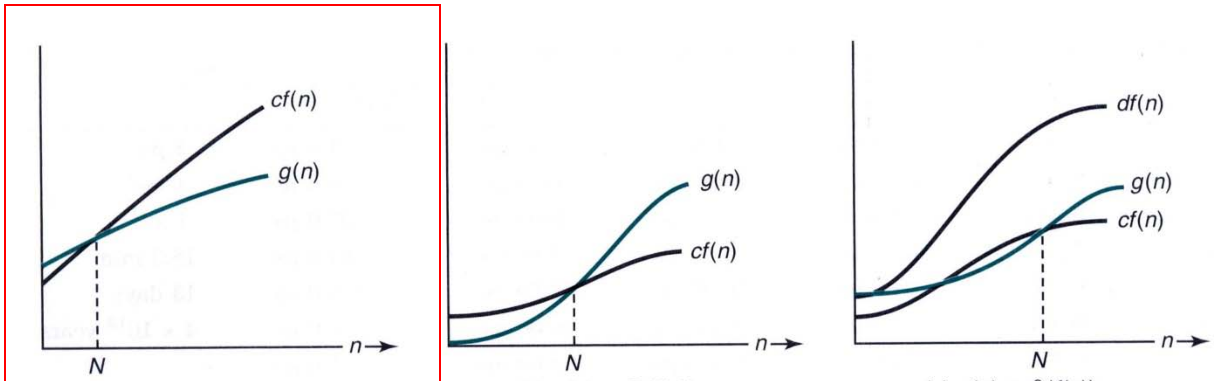
  - $g(n) \in O(f(n))$

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)

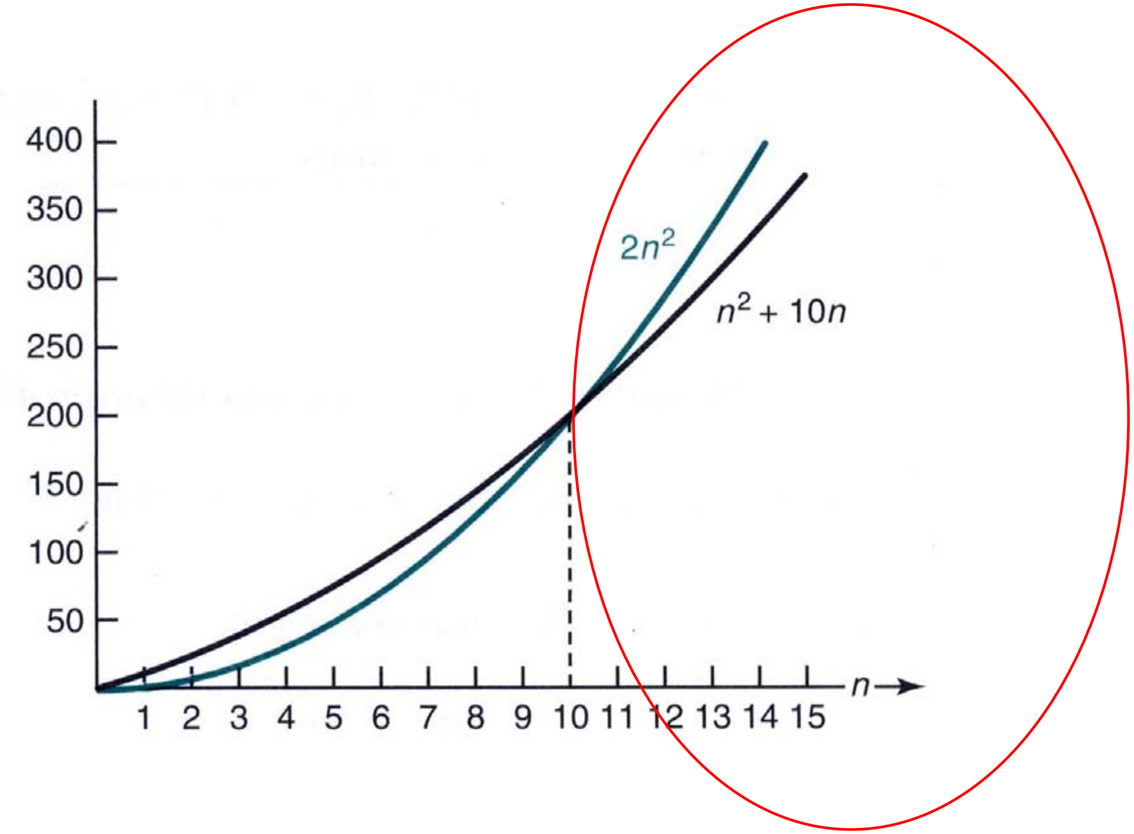


(a) $g(n) \in O(f(n))$ (b) $g(n) \in \Omega(f(n))$ (c) $g(n) \in \Theta(f(n))$

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 $2n^2$.



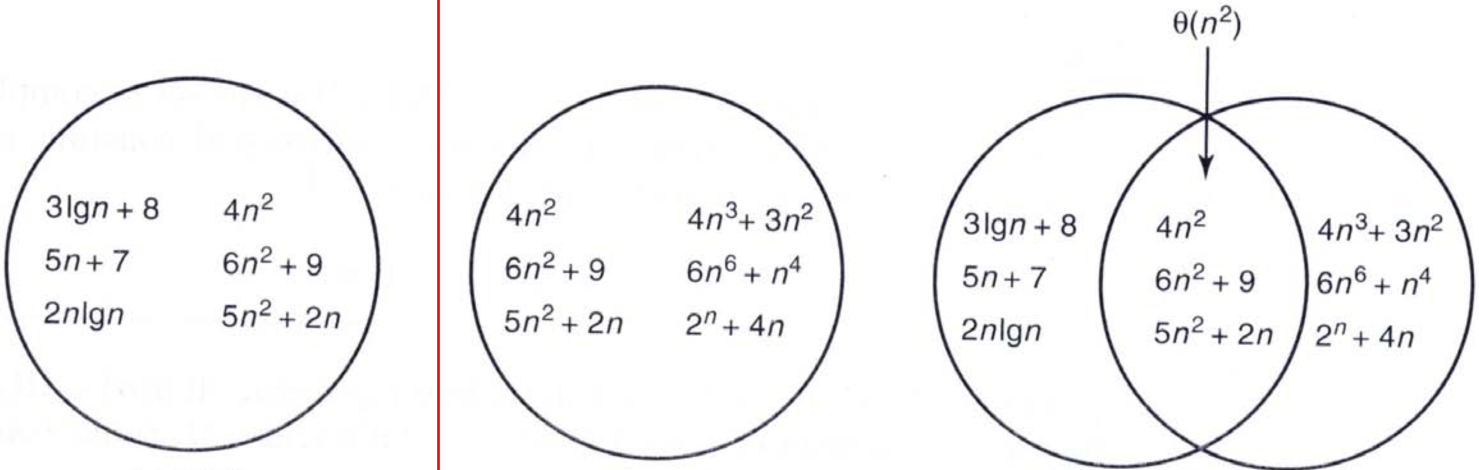
Big O Notation: More Examples

- $5n^2 \in O(n^2)$
  - Take *c = 5* and *N = 0*, then for all n such that $n \geq N$, $5n^2 \leq cn^2$.
- $T(n) = \dfrac{n(n-1)}{2}$
  - Because, for $n \geq 0$, $\dfrac{n(n-1)}{2} \leq \dfrac{n^2}{2}$
  - 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.



(a) $O(n^2)$

(b) $\Omega(n^2)$

(c) $\theta(n^2) = O(n^2) \cap \Omega(n^2)$

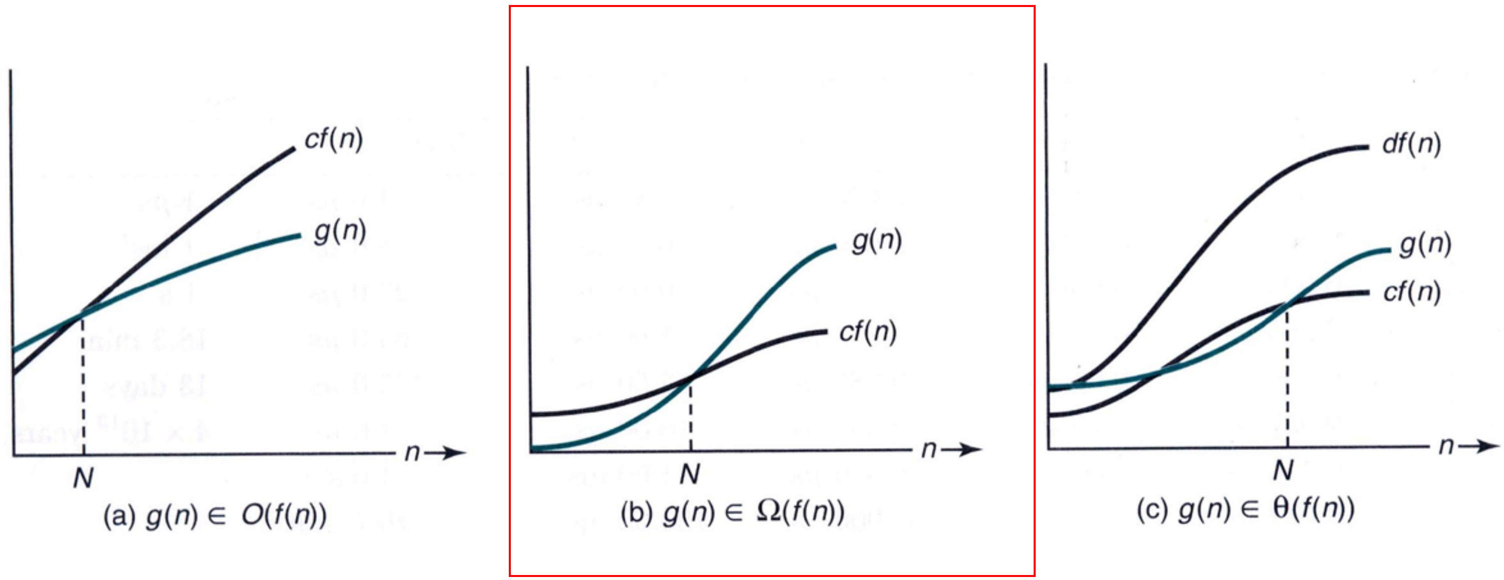
Rigorous Definition to Order: Ω

- 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 $n \geq N$,

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

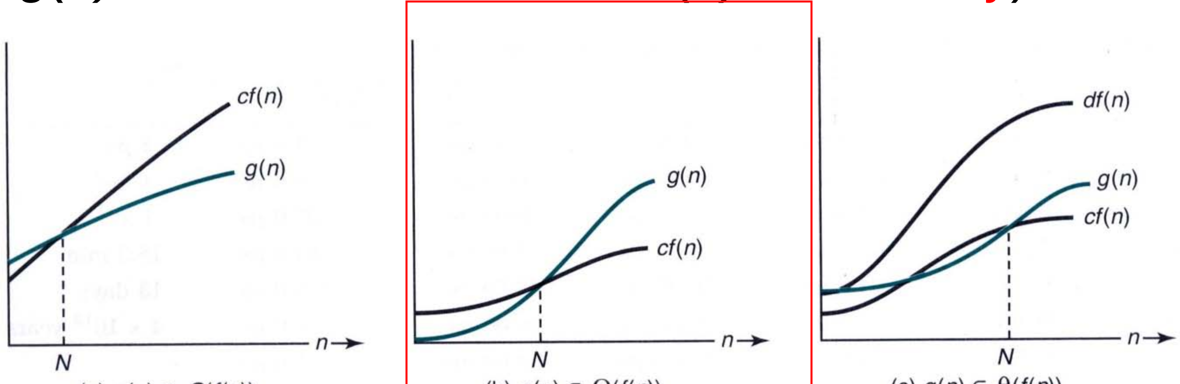
  - $g(n) \in \Omega(f(n))$

Illustrating "big O", Ω, and Θ



Ω Notation: Definition

- Meaning of $g(n) \in \Omega(f(n))$
  - Although *g(n)* starts out below *cf(n)* in the figure, eventually it goes above *cf(n)* and stays there.
  - If *g(n)* is the time complexity for some algorithm, eventually the running time of the algorithm will be at least as bad as *f(n)*
  - *f(n)* is called as an asymptotic lower bound (*of what?*) (i.e. *g(n)* cannot run faster than *f(n)*, eventually)



(a) $g(n) \in O(f(n))$ (b) $g(n) \in \Omega(f(n))$ (c) $g(n) \in \Theta(f(n))$

Ω Notation: Example

- Meaning of $n^2+10n \in \Omega(n^2)$
  - Take *c = 1* and *N = 0*.
  - For all integer $n \geq 0$, it holds that $n^2+10n \geq n^2$
  - Therefore, n^2 is an asymptotic lower bound for the time complexity function of n^2+10n. (I.e., n^2+10n belongs to $\Omega(n^2)$)
- $5n^2 \in \Omega(n^2)$
  - Take *c = 1* and *N = 0*.
  - For all integer $n \geq 0$, it holds that $5n^2 \geq 1 \times n^2$
  - Therefore, n^2 is an asymptotic lower bound for the time complexity function of $5n^2$.

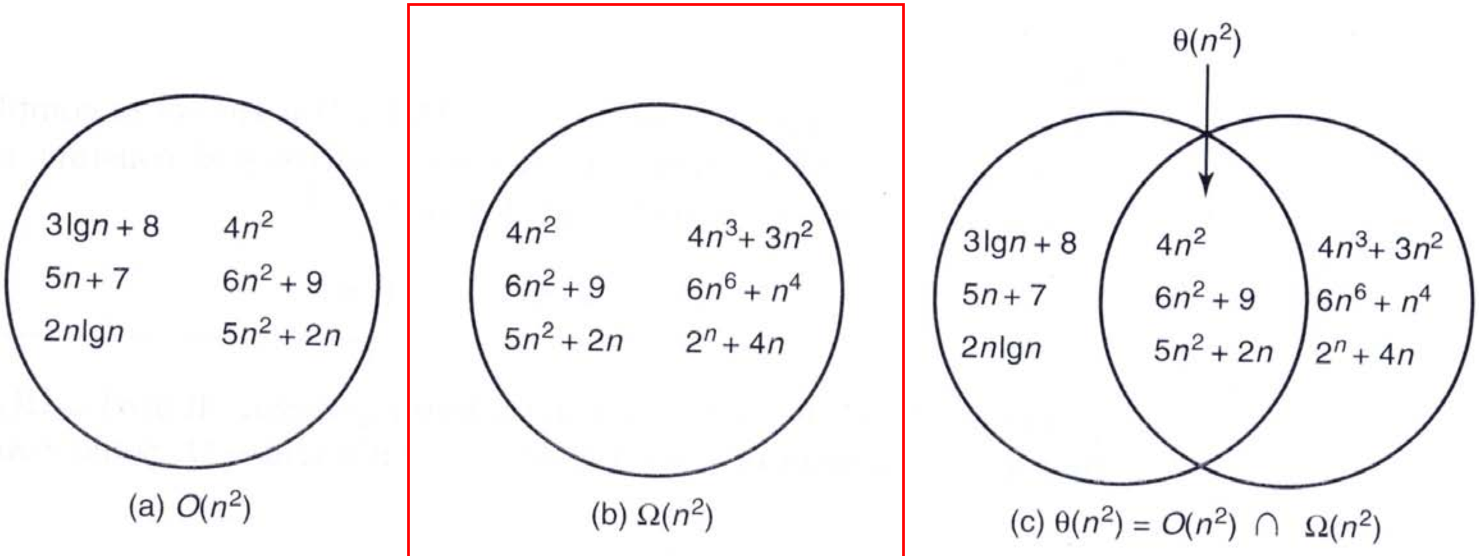
Ω Notation: More Examples

- $T(n) = \frac{n(n-1)}{2}$
  - Because, for $n \geq 2$, $n - 1 \geq n/2$, so it holds that $\frac{n(n-1)}{2} \geq \frac{n}{2} \times \frac{n}{2} = \frac{1}{4} n^2$
  - Therefore, we can take *c = 1/4* and *N = 2*, to conclude that $T(n) \in \Omega(n^2)$.
- $n^3 \in \Omega(n^2)$
  - Because, for $n \geq 1$, it holds that $n^3 \geq 1 \times n^2$
  - Therefore, we can take *c = 1* and *N = 1*, to conclude that $n^3 \in \Omega(n^2)$

Ω Notation: Last Example

- $n \in \Omega(n^2)$
  - Proof by contradiction.
  - Suppose it is true that $n \in \Omega(n^2)$.
  - Then, for all integer $n \geq N$, there must exist some positive real number $c > 0$, and non-negative integer N.
  - Let's divide both sides by cn.
  - Then, we will get $1/c \geq n$, which is impossible.
  - Therefore, n does not belong to $\Omega(n^2)$.

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



Rigorous Definition to Order: Θ

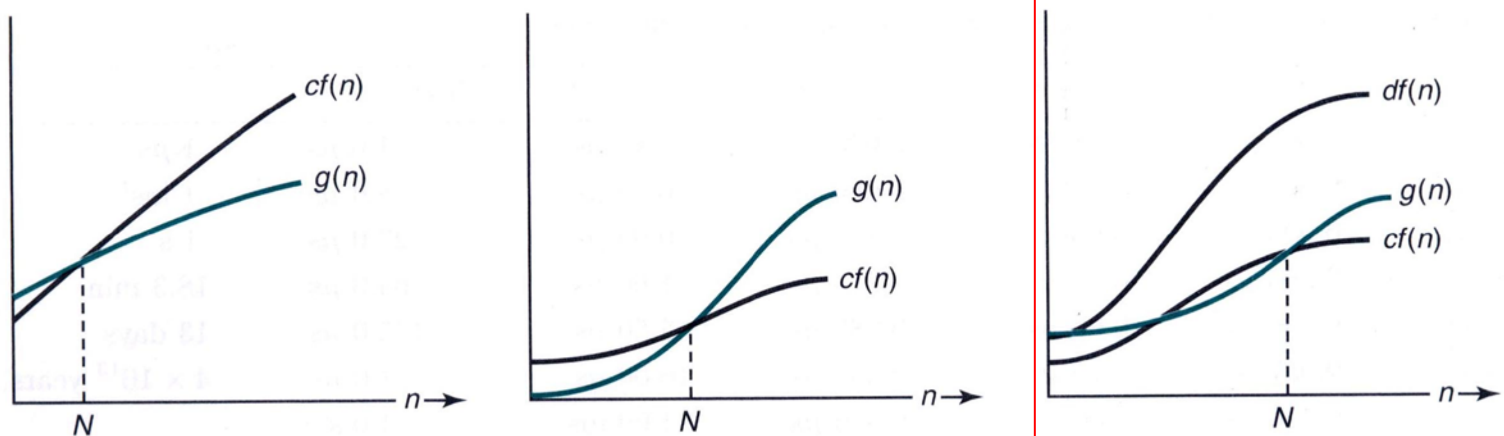
- Definition: (Asymptotic Tight Bound)
  - For a given complexity function $f(n)$, $\Theta(f(n))$ is the set of complexity functions $g(n)$ for which there exists some positive real constants c and d and some non-negative integer N such that for all $n \geq N$,

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

  - $g(n) \in \Theta(f(n))$, we say that $g(n)$ is order of $f(n)$.
  - Example:

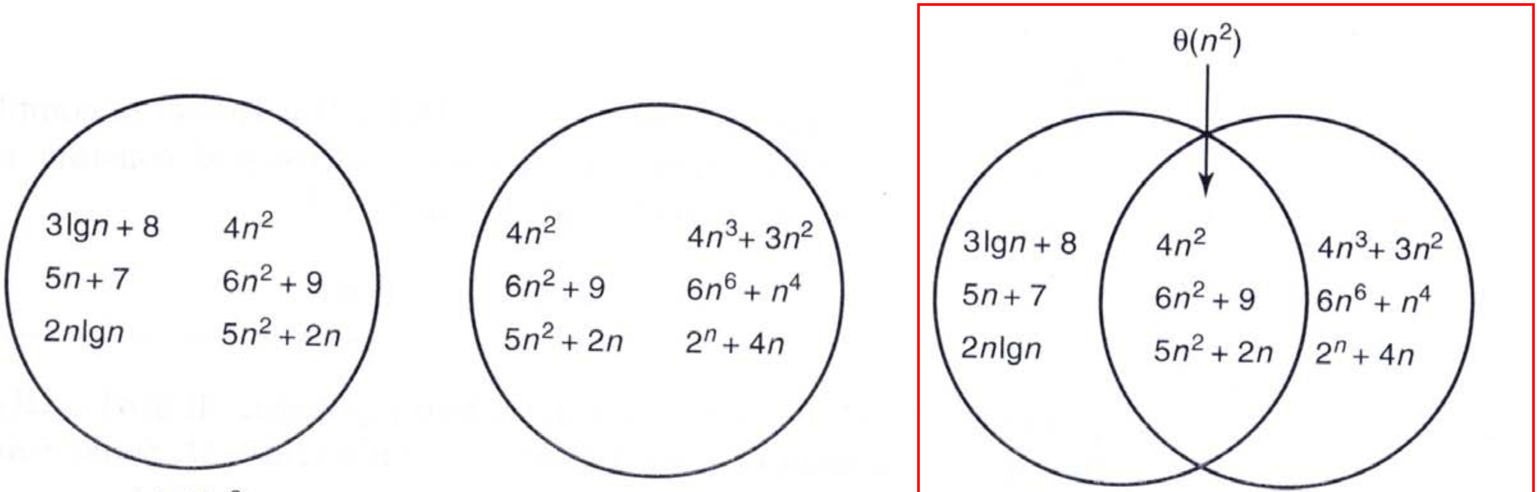
$$T(n) = \frac{n(n-1)}{2} \qquad T(n) \in \Theta\left(n^2\right)$$

Illustrating “big O”, Ω, and Θ



(a) $g(n) \in O(f(n))$ (b) $g(n) \in \Omega(f(n))$ (c) $g(n) \in \theta(f(n))$

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



(a) $O(n^2)$

(b) $\Omega(n^2)$

(c) $\theta(n^2) = O(n^2) \cap \Omega(n^2)$

Rigorous Definition to Order: Small *o*

- Definition:
  - For a given complexity function $f(n)$, $o(f(n))$ is the set of complexity functions $g(n)$ satisfying the following: For ***every*** positive real constant c there exists a non-negative integer N such that for all $n \geq N$,

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

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

Big *O* vs. Small *o*

- Difference
  - Big *O*: 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 $n \geq N$
  - Small *o*: For a given complexity function $f(n)$, $o(f(n))$ is the set of complexity functions $g(n)$ satisfying the following: For ***every*** positive real constant c there exists a non-negative integer N such that for all $n \geq N$,

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

  - If $g(n) \in o(f(n))$, $g(n)$ is eventually much better than $f(n)$.

Small *o* Notation: Example

- $n \in o(n^2)$
  - Suppose $c > 0$. We need to find an N such that, for $n \geq N$, $n \leq cn^2$.
  - If we divide both sides by cn,
  - Then, we get $1/c \leq n$
  - Therefore, it suffice to choose any $N \geq 1/c$.
  - For example, if $c=0.00001$, we must take equal to at least 100,000. That is, for $n \geq 100{,}000$, $n \leq 0.00001n^2$.

Small *o* Notation: Example2

- $n \in o(5n)$?
  - Proof by contradiction.
  - Let c = 1/6. If $n \in o(5n)$, then there must exist some N such that, for $n \geq N$,
    $$n \leq \frac{1}{6} \times 5n = \frac{5}{6} n$$
  - But it is impossible.
  - This contradiction proves that n is not in $o(5n)$.

Properties of Order Functions

- $g(n) \in O(f(n))$ iff $f(n) \in \Omega(g(n))$
- $g(n) \in \Theta(f(n))$ iff $f(n) \in \Theta(g(n))$
- If $b > 1$ and $a > 1$, then $\log_a n \in \Theta(\log_b n)$.
- If $b > a > 0$, then $a^n \in o(b^n)$.
- For all $a > 0$, $a^n \in o(n!)$.
- See the following ordering, where $k>j>2$ and $b>a>1$.

$$\Theta(\lg n), \Theta(n), \Theta(n \lg n), \Theta(n^2), \Theta(n^j), \Theta(n^k), \Theta(a^n), \Theta(b^n), \Theta(n!)$$

- If $c \geq 0$, $\mathrm{d} \geq 0$, $g(n) \in O(f(n))$, and $h(n) \in \Theta(f(n))$, then

$$c \times g(n) + d \times h(n) \in \Theta(f(n))$$

Using a Limit to Determine Order (1/2)

- Theorem 1.3

$$\lim_{n\to\infty}\frac{g(n)}{f(n)} = \begin{cases} c > 0 & g(n)\in\Theta(f(n)) \\ 0 & g(n)\in o(f(n)) \\ \infty & f(n)\in o(g(n)) \end{cases}$$

- Example: Theorem 1.3 implies

$$\frac{n^2}{2}\in o(n^3) \quad \text{because} \quad \lim_{n\to\infty}\frac{n^2/2}{n^3} = \lim_{n\to\infty}\frac{1}{2n} = 0$$

Using a Limit to Determine Order (2/2)

- Theorem 1.4 (L'Hopital's Rule)

  If $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$, then

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \left(\frac{g'(n)}{f'(n)} \right)$$

- Example:
  - $\lg n \in o(n)$ $\quad \lim_{n\to\infty} \frac{\lg n}{n} = \lim_{n\to\infty} \left(\frac{\frac{1}{n \ln 2}}{1} \right) = 0$
  - $\log_a n \in \Theta(\log_b n)$

$$\lim_{n\to\infty} \frac{\log_a n}{\log_b n} = \lim_{n\to\infty} \left(\frac{\frac{1}{n \ln a}}{\frac{1}{n \ln b}} \right) = \frac{\log b}{\log a} > 0$$