1.3.2 Applying the Theory

Alg.	**A**	**B**
$T(n)$	n	n^2
b.o. execution time	$1000t$	t

$$n \times 1000\,t \begin{Bmatrix} > \\ = \\ < \\ ? \end{Bmatrix} n^2 \times t$$

1.4 Order

- **1.4.1 An Intuitive Introduction to Order**
  - **a, b, c, d: constants**

$$an + b \in \Theta(n) \quad \text{linear}$$

$$an^2 + bn + c \in \Theta(n^2) \quad \text{quadratic}$$

$$an^3 + bn^2 + cn + d \in \Theta(n^3) \quad \text{cubic}$$

ignore low order terms – see table 1.3

$$\Theta(\log n) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^3) \quad \Theta(2^n)$$

  - **See figure 1.3**
  - **See table 1.4**

Figure 1.3 Growth rates of some common complexity functions.

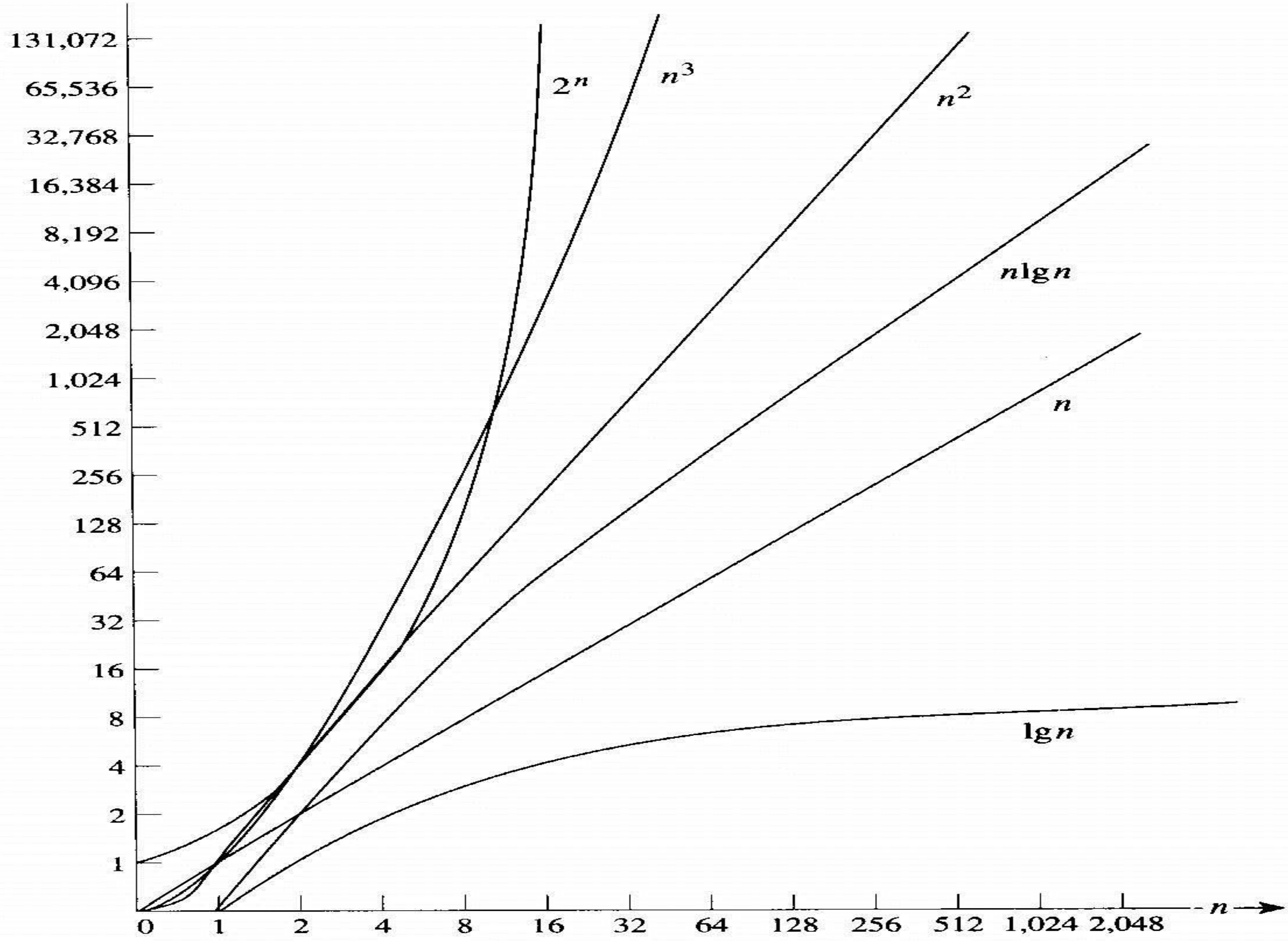


Table 1.4 Execution times for algorithms with the given time complexities

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	0.1 μs	1 μs	1 μs
20	0.004 μs	0.02 μs	0.086 μs	0.4 μs	8 μs	1 ms†
30	0.005 μs	0.03 μs	0.147 μs	0.9 μs	27 μs	1 s
40	0.005 μs	0.04 μs	0.213 μs	1.6 μs	64 μs	18.3 min
50	0.006 μs	0.05 μs	0.282 μs	2.5 μs	125 μs	13 days
10^2	0.007 μs	0.10 μs	0.664 μs	10 μs	1 ms	4×10^{13} years
10^3	0.010 μs	1.00 μs	9.966 μs	1 ms	1 s	
10^4	0.013 μs	10 μs	130 μs	100 ms	16.7 min	
10^5	0.017 μs	0.10 ms	1.67 ms	10 s	11.6 days	
10^6	0.020 μs	1 ms	19.93 ms	16.7 min	31.7 years	
10^7	0.023 μs	0.01 s	0.23 s	1.16 days	31,709 years	
10^8	0.027 μs	0.10 s	2.66 s	115.7 days	3.17×10^7 years	
10^9	0.030 μs	1 s	29.90 s	31.7 years		

*1 μs = 10^{-6} second.

†1 ms = 10^{-3} second.

1.4.2 A Rigorous Introduction to Order

- **Def. 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 nonnegative integer *N* s.t. for all *n* ≥ *N*,**

$$g(n) \le c \times f(n): \text{ asymptotic upper bound}$$

$$g(n) \in O(f(n)) \qquad g(n) \text{ is big } O \text{ of } f(n)$$

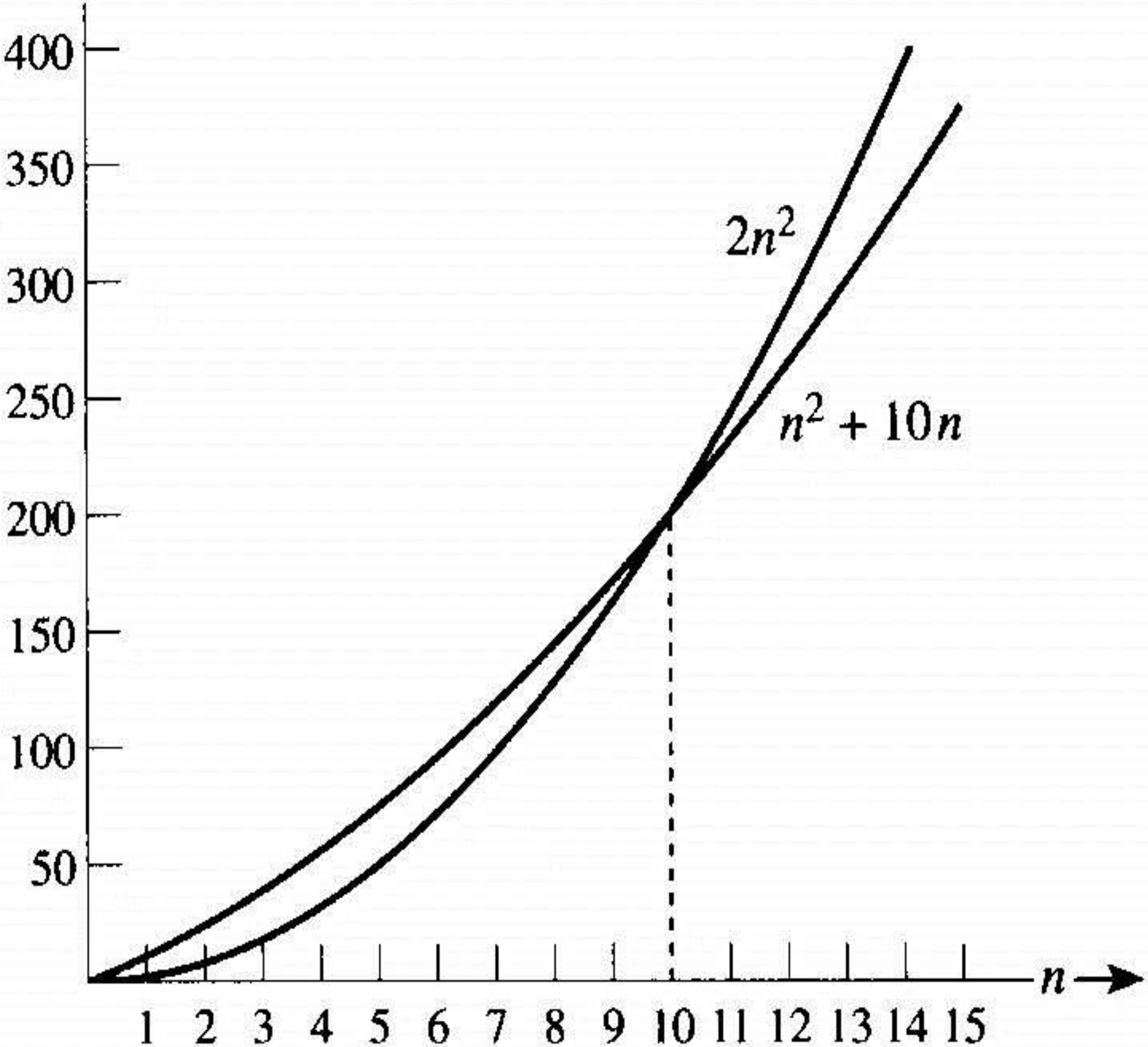
- **Ex)** $\underbrace{n^2 + 10n}_{g(n)} \le \underbrace{2}_{c} \cdot \underbrace{n^2}_{f(n)} \quad n \ge \underbrace{10}_{N} \quad n^2 + 10n \in O(n^2)$

$$n \le 1 \cdot n^2 \quad n \ge 1 \quad n \in O(n^2)$$

$$n(n-1)/2 \le \frac{1}{2} n^2 \quad n \ge 0 \quad n(n-1)/2 \in O(n^2)$$

- **See Figure 1.5 and 1.4**

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



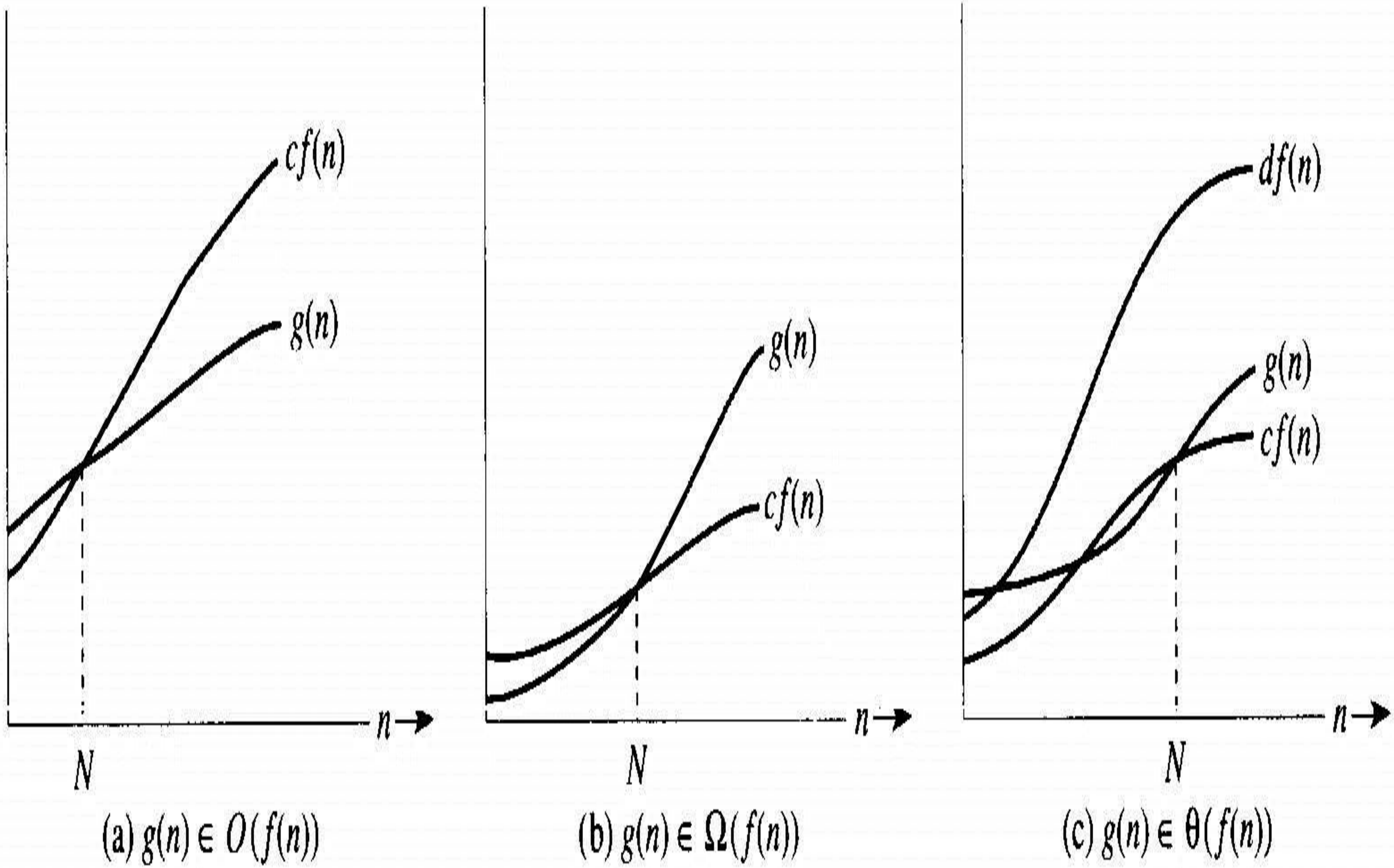


Figure 1.4 Illustrating "big O," Ω, and Θ.

Omega

- **Def. Omega: 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 nonnegative integer N s.t. for all $n \geq N$,**

$$g(n) \geq c \times f(n): \text{ asymptotic lower bound}$$

$$g(n) \in \Omega(f(n)) \qquad g(n) \text{ is omega of } f(n)$$

- **Ex)** $n^2 + 10n \geq n^2 \quad n \geq 0 \quad n^2 + 10n \in \Omega(n^2)$

$$n(n-1)/2 \geq \frac{1}{4}n^2 \quad n \geq 2 \quad n(n-1)/2 \in \Omega(n^2)$$

$$n^3 \geq 1 \cdot n^2 \quad n \geq 1 \quad n^3 \in \Omega(n^2)$$

Order

- **Def. Order: for a given complexity function *f(n)***

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

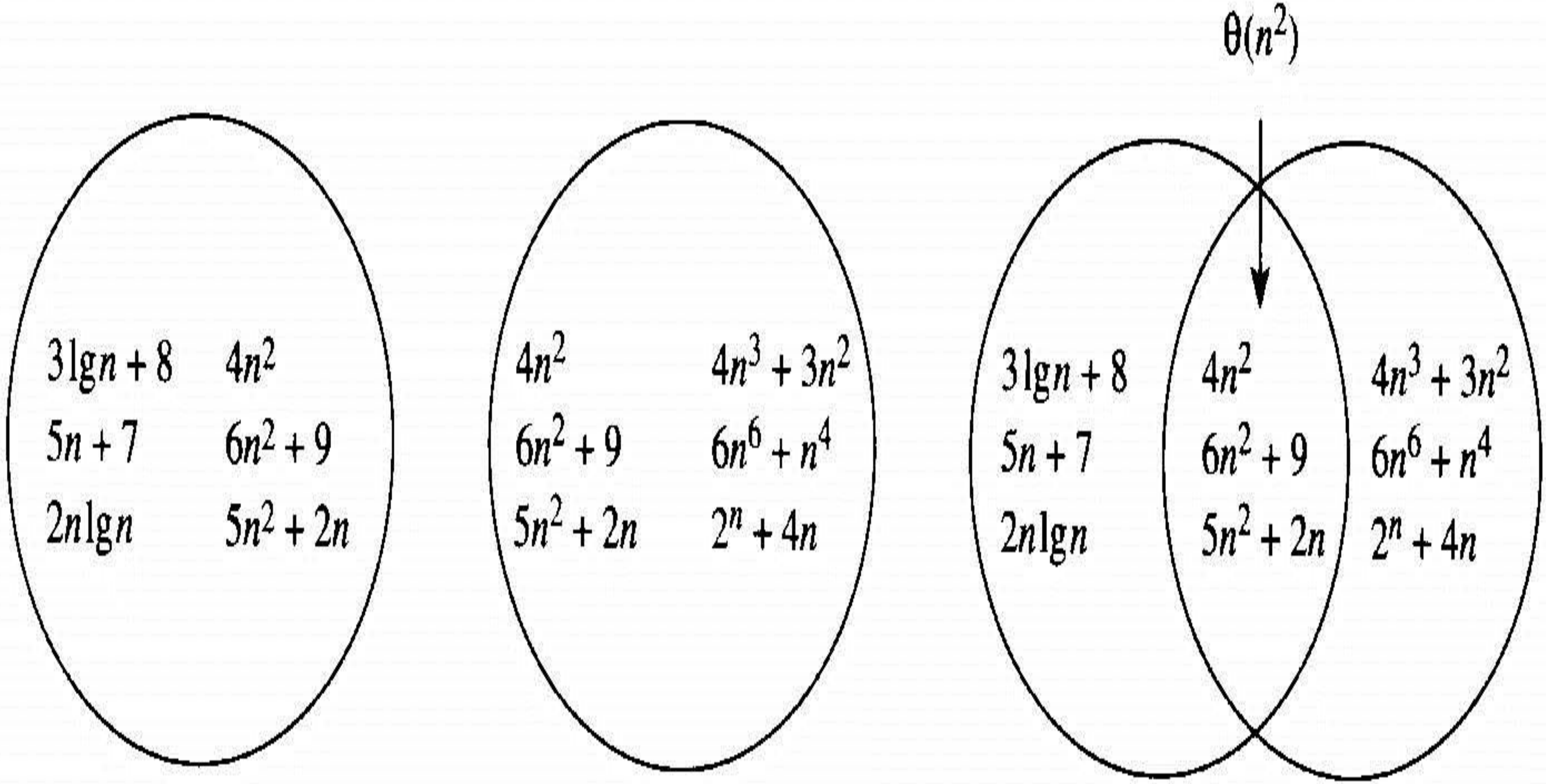
$\Theta(f(n))$ **is the set of complexity functions *g(n)* for which there exists some positive real constant c and d and some nonnegative integer *N* s.t. for all $n \geq N$,**

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

$g(n) \in \Theta(f(n))$ $\quad$ $g(n)$ **is order of** $f(n)$

- **Ex)** $n(n-1)/2 \in \Theta(n^2)$
- **See Figure 1.6**

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



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

Small o

- **Def. 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 nonnegative integer *N* s.t. for all $n \geq N$,**

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

$g(n) \in o(f(n))$ $\quad$ *g(n)* is small o of *f(n)*

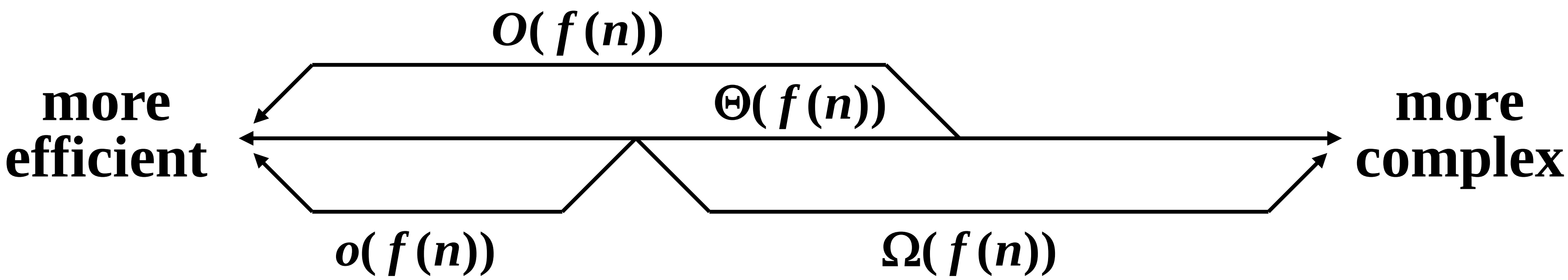
g(n) is eventually much better function than *f(n)*

- **Ex)** $\underbrace{n}_{g(n)} \leq c \underbrace{n^2}_{f(n)}$ for $n \geq \underbrace{\frac{1}{c}}_{N}$

Theorem 1.2

If $g(n) \in o(f(n))$ **then** $g(n) \in O(f(n)) - \Omega(f(n))$

That is, $g(n)$ **is in** $O(f(n))$ **but not in** $\Omega(f(n))$



- **Note that** $o(f(n)) \neq O(f(n)) - \Omega(f(n))$

  → **see ex. 1.20.**

  But equality holds for the time complexities of actual algorithms.

Properties of Order

1. $g(n) \in O(f(n))$ if and only if $f(n) \in \Omega(g(n))$

2. $g(n) \in \Theta(f(n))$ if and only if $f(n) \in \Theta(g(n))$

3. If $b > 1$ and $a > 1$, then $\log_a n \in \Theta(\log_b n)$

4. If $b > a > 0$, then $a^n \in o(b^n)$

5. For all $a > 0$, $a^n \in o(n!)$

6. Assume $k > j > 2$ and $b > a > 1$.

   $\Theta(\lg n)$ $\Theta(\mathrm{n})$ $\Theta(n \lg n)$ $\Theta(n^2)$ $\Theta(n^j)$ $\Theta(n^k)$ $\Theta(a^n)$ $\Theta(b^n)$ $\Theta(n!)$

7. If $c \geq 0$, $d > 0$, $g(n) \in O(f(n))$, $h(n) \in \Theta(f(n))$, then

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