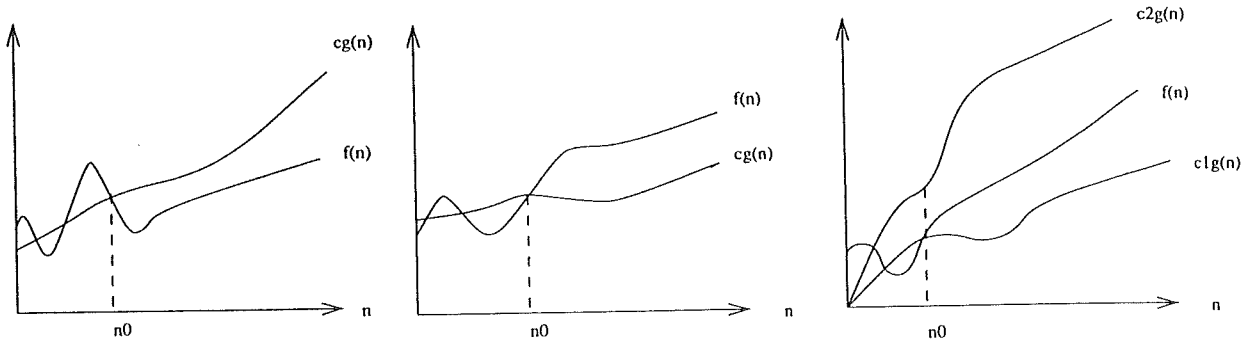

Illustrating the [a] O, [b] Ω, and [c] Θ notations

- $f(n) = O(g(n))$ means $c \cdot g(n)$ is an *upper bound* on $f(n)$. Thus there exists some constant c such that $f(n)$ is always $\leq c \cdot g(n)$, for large enough n.
- $f(n) = \Omega(g(n))$ means $c \cdot g(n)$ is a *lower bound* on $f(n)$. Thus there exists some constant c such that $f(n)$ is always $\geq c \cdot g(n)$, for large enough n.
- $f(n) = \Theta(g(n))$ means $c_1 \cdot g(n)$ is an upper bound on $f(n)$ and $c_2 \cdot g(n)$ is a lower bound on $f(n)$, for large enough n. Thus there exists constants c_1 and c_2 such that $f(n) \leq c_1 \cdot g(n)$ and $f(n) \geq c_2 \cdot g(n)$. This means that $g(n)$ is a nice, tight bound on $f(n)$.

TABLE 3-1 VARIOUS ORDERS OF MAGNITUDE

n	$\log_2 n$	$n \log_2 n$	n^2	n^3	2^n
2	1	2	4	8	4
4	2	8	16	64	16
8	3	24	64	512	256
16	4	64	256	4096	65,536
32	5	160	1024	32,768	4,294,967,296
128	7	896	16,384	2,097,152	3.4×10^{38}
1024	10	10,240	1,048,576	1,073,741,824	1.8×10^{308}
65,536	16	1,048,576	4,294,967,296	2.8×10^{14}	Forget it!

Table 3-1 gives the linear, quadratic, cubic, exponential, and logarithmic orders of magnitude for selected values of n. Clearly, you should avoid cubic and exponential algorithms unless n is small.