CS4349
Ch 3: Growth of Functions

Asymptotic Efficiency of Algorithms

- The ***asymptotic*** efficiency of algorithms focuses on how the running time of an algorithm increases with the size of the input *in the limit*, as the size of the input increases without bound.

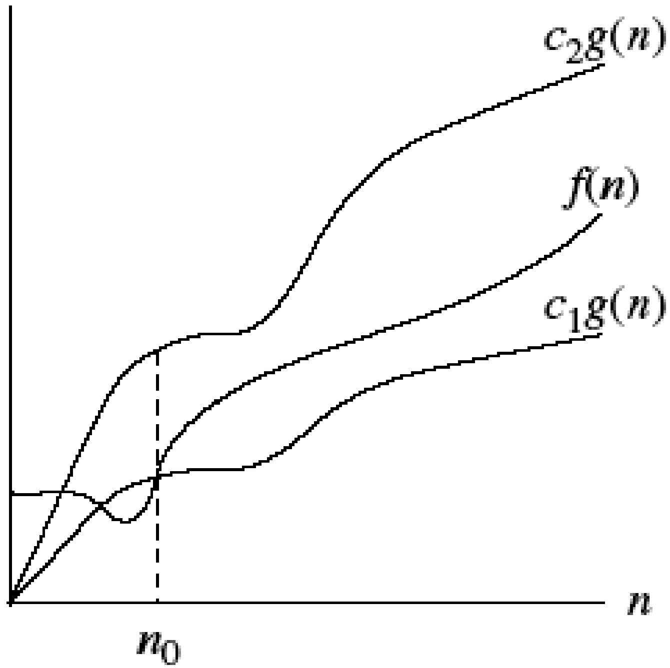
Θ - Notation (theta)

- ***Θ - notation (theta)***
  - $\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2, and n_0 s.t. $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0 \}$
  - $\Theta(g(n))$ is the set of functions
  - $f(n) = \Theta(g(n))$ really means that $f(n)$ belongs to $\Theta(g(n))$
  - *g(n)* is called an ***asymptotically tight bound*** for *f(n)*

Example: $\Theta(1)$ means a constant or a constant function w.r.t. some variable. f(n) = $\Theta(1)$ means f(n) lies sandwiched between two constants c1, c2 > 0 for all sufficiently large n.

Θ - Notation

- ***Θ - notation***



Θ notation bounds a function to within constant factors. We write f(n)=g(Θ) if there exist positive constants n_0, c_1, and c_2 s.t. at and to the right of n_0, the value of f(n) always lies between c_1g(n) and c_2g(n) inclusive.

If *f(n)* is a polynomial of degree *d,* then *f(n)* = $\Theta(n^d)$

$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

Θ - Notation Example 1

- Show that $3n^2 + 4n - 2 = \Theta(n^2)$

  $c_1 n^2 \le 3n^2 + 4n - 2 \le c_2 n^2$

  Left hand inequality: $c_1 \le 3 + 4/n - 2/n^2$
- We can make the left hand inequality hold for any value of $n \ge 1$ by choosing any constant $c_1 \le 5$.

- Right hand inequality: $3 + 4/n - 2/n^2 \le c_2 n^2$
  - We can make the right hand inequality hold for any value of $n \ge 1$ by choosing any constant $c_2 \ge 5$.
  - We can also make the right hand inequality hold for any value of $n \ge 2$ by choosing any constant $c_2 \ge 11/2$.
- The point here is that some choice for c_1, c_2, and n_0 exists, therefore the given function is $\Theta(n^2)$.

Θ - Notation Example 2

- Show that $n^2 - 2n = \Theta(n^2)$

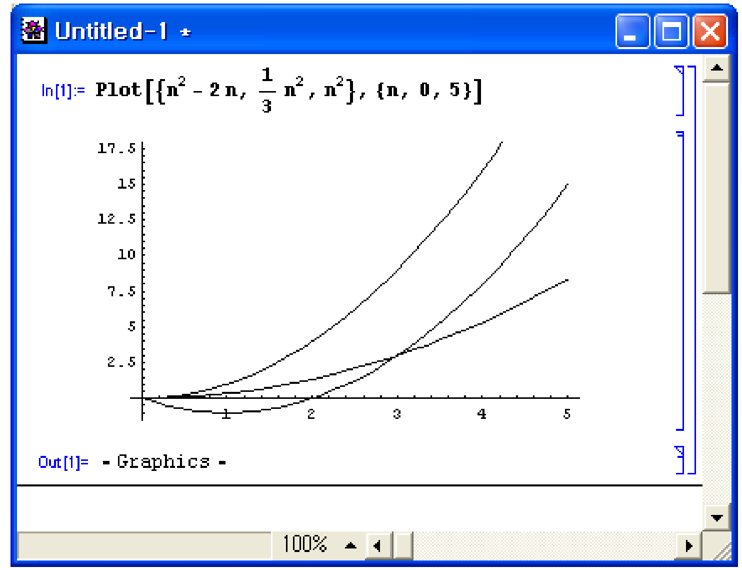
$$c_1 n^2 \le n^2 - 2n \le c_2 n^2$$

$$c_1 \le 1 - \frac{2}{n} \le c_2$$

$$c_1 \le \frac{1}{3} \qquad c_2 \ge 1$$

$$n \ge 3$$

$$n_0 = 3$$



- Some choice for c_1, c_2, and n_0 exists, therefore the function is $\boldsymbol{\Theta}(n^2)$
- **Note:** $\boldsymbol{\Theta}(n^0)$ is equivalent to $\boldsymbol{\Theta}\mathbf{(1)}$

Θ - Notation Example 3

- Show that $200n^2 - 100n = \Theta(n^2)$

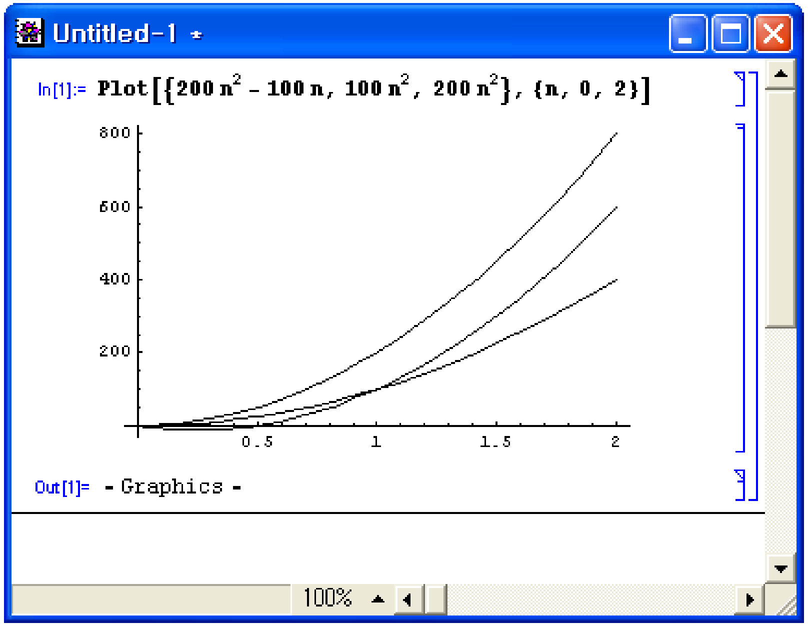
$$c_1 n^2 \le 200n^2 - 100n \le c_2 n^2$$

$$c_1 \le 200 - \frac{100}{n} \le c_2$$

$$c_1 \le 100 \qquad c_2 \ge 200$$

$$n \ge 1 \qquad n \ge 1$$

$$n_0 = 1$$



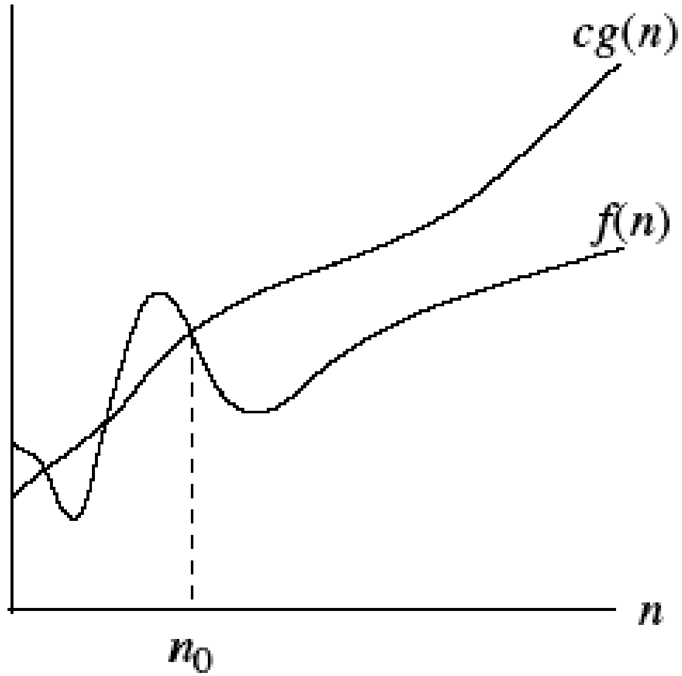
- Some choice for c_1, c_2, and n_0 exists, therefore the function is $\Theta(n^2)$

O – Notation (big-oh)

- **O - notation (big-oh)**
  - $f(n) = \mathbf{O}(g(n))$: $g(n)$ is an asymptotic upper bound for $f(n)$
  - $\mathbf{O}(g(n))$ = { $f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$ }
- $f(n) = \Theta(g(n))$ implies $f(n) = \mathbf{O}(g(n))$,
  but $f(n) = \mathbf{O}(g(n))$ does NOT imply $f(n) = \Theta(g(n))$.
- **Example**:
  - If $f(n) = n^2 - 2n$, then $f(n) = \mathbf{O}(n^2)$ also $f(n)= \mathbf{O}(n^3) = ...$
  - If $f(n) = 200n^2 - 100n$, then $f(n) = \mathbf{O}(n^2)$ also $f(n)= \mathbf{O}(n^3) = ...$
- **O** is good for describing the worst-case running time of an algorithm
- Formally, when we say the running time is O(f(n)), we mean all running times on inputs of length n including the worst-case running time is O(f(n)) even if some inputs have better running times.

O - Notation

- ***O - notation***



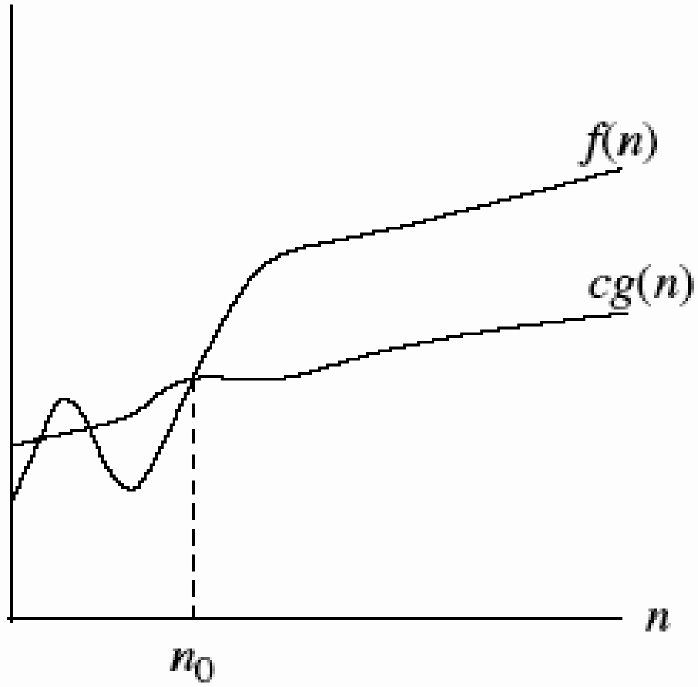
$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

Ω - Notation (big-omega)

- **Ω - notation (big-omega)**
  - f(n) = Ω(g(n)): g(n) is an asymptotic lower bound for f(n)
  - Ω(g(n)) = {f(n): there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$ }
- f(n) = Θ(g(n)) implies f(n) = Ω (g(n)),
  but f(n) = Ω (g(n)) does NOT imply f(n) = Θ(g(n))
- **Example**:
  - n^2 - 2n = Ω (n^2)
  - 200n^2 - 100n = Ω (n^2) = Ω (n) = Ω(1)
  - n^2 = Ω (n)
- Ω is good for describing the best case running time of an algorithm
- **Theorem**
  - f(n) = Θ(g(n)) if and only if f(n) = **O**(g(n)) and f(n) = Ω (g(n))

Ω - Notation

- ***Ω - notation (omega)***



$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

Insertion Sort revisited

- Based on asymptotic notation definitions:
- The running time of Insertion sort belongs to $\Omega(n)$ and $O(n^2)$, as it lies between a linear function of n (for the best case) and a quadratic function of n (for the worst case).

Asymptotic Notation in Equations

- Below "LHS=RHS" means LHS is RHS (but not vice versa). From a set definition point of view, it means LHS belongs to RHS (but not vice versa).
  - $n = \boldsymbol{O}(n^2)$ means "n is $\boldsymbol{O}(n^2)$" or "n belongs to $\boldsymbol{O}(n^2)$"
- In general, asymptotic notation stands for some anonymous function
- ***Example***:
  - $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means
    - there is some function $f(n) \in \Theta(n)$ that makes the equation true, i.e. $f(n) = 3n + 1$
  - $2n^2 + \Theta(n) = \Theta(n^2)$ means
    - for any function $f(n) \in \Theta(n)$, there is some function $g(n) \in \Theta(n^2)$ such that $2n^2 + f(n) = g(n)$ for all n
  - $2n^2 + 3n + 1 = 2n^2 + \Theta(n) = \Theta(n^2)$

o – Notation (little-oh)

- ***o - notation (little-oh)***
  - ***O - notation*** may or may not be asymptotically tight
    - $2n^2=O(n^2)$ is asymptotically tight, but $2n=O(n^2)$ is not.
  - *f(n)* = ***o***(*g(n)*): *g(n)* is an <u>*upper bound*</u> of *f(n)* that is <u>*not asymptotically tight*</u>
  - ***o***(*g(n)*) = { *f(n)*: for any positive constant *c* > 0, there exists a constant n_0 > 0 such that $0 \le f(n) < cg(n)$ for all $n \ge n_0$ }
  - $2n = \boldsymbol{o}(n^2)$, but $2n^2 \neq \boldsymbol{o}(n^2)$
- ***O***: for some constant *c*, ***o***: for all constant *c*
- In the ***o-notation***, the function *f(n)* becomes <u>*insignificant*</u> relative to *g(n)* as *n* approaches infinity
  - i.e., $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

ω - Notation (little-omega)

- ***ω* - notation (little-omega)**
  - $2n^2 = \Omega(n^2)$ is asymptotically tight, but $2n^3 = \Omega(n^2)$ is not
  - *f(n)* = $\boldsymbol{\omega}$ (*g(n)*): *g(n)* is a <u>*lower bound*</u> of *f(n)* that is <u>*not asymptotically tight*</u>
  - ω(*g(n)*) = { *f(n)*: for any positive constant *c* > 0, there exists a constant $n_0 > 0$ such that $0 \le cg(n) < f(n)$ for all $n \ge n_0$ }
  - $2n^2 = \omega(n)$, but $2n^2 \neq \omega(n^2)$
- In the ***ω*-notation**, the function *f(n)* becomes <u>*arbitrarily large*</u> relative to *g(n)* as *n* approaches infinity
  - i.e.,

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

Standard Notations and Common Functions

- ***Monotonicity***: A function *f(n)* is:
  - monotonically increasing if $a \le b$ implies $f(a) \le f(b)$
  - monotonically decreasing if $a \le b$ implies $f(a) \ge f(b)$
  - strictly increasing if $a < b$ implies $f(a) < f(b)$
  - strictly decreasing if $a < b$ implies $f(a) > f(b)$
- ***Floors and ceilings***
  - Floor: $\lfloor x \rfloor$ is the greatest integer $\le x$
  - Ceiling: $\lceil x \rceil$ is the least integer $\ge x$
  - ***Examples***:

$$\lfloor 3 \rfloor = 3 \qquad \lceil 3 \rceil = 3$$

$$\lfloor 3.3 \rfloor = 3 \qquad \lceil 3.3 \rceil = 4$$

$$\lfloor 3.9 \rfloor = 3 \qquad \lceil 3.9 \rceil = 4$$

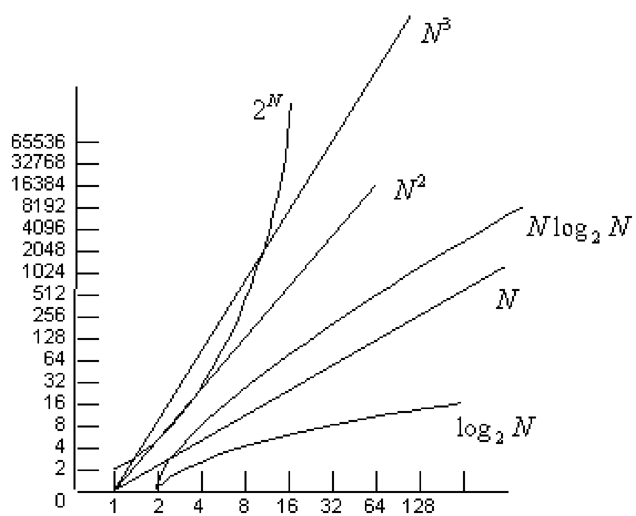
$$\lceil n/2 \rceil + \lfloor n/2 \rfloor = n, \quad \lceil \lceil n/a \rceil / b \rceil = \lceil n/ab \rceil, \quad \lfloor \lfloor n/a \rfloor / b \rfloor = \lfloor n/ab \rfloor$$

Standard Notations and Common Functions

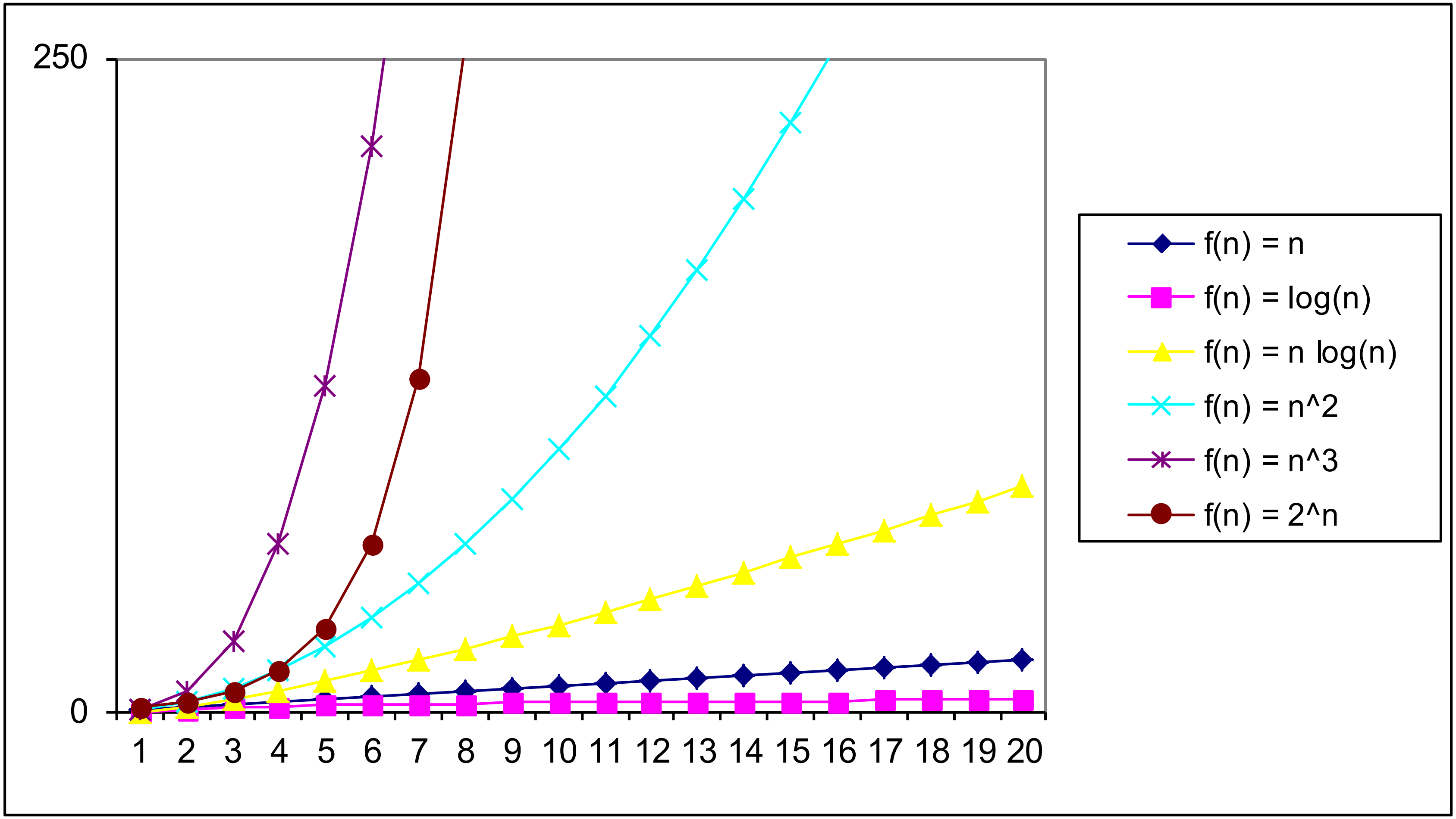
- ***Logarithms***
  - $\log_b n$: logarithm of n base b
  - $\lg n = \log_2 n$ (binary logarithm)
  - $\ln n = \log_e n$ (natural logarithm, $e = 2.7182...$)
- ***Factorials***
  - $n! = \begin{cases} 1 & \text{if } n = 0, \\ n(n-1)! & \text{if } n > 0. \end{cases}$
  - $n! = 1 * 2 * 3 * ... * n$
  - $n! \leq n^n$, thus $\boldsymbol{O}(n^n)$
- Stirling's Approximation
  - $$n! = \sqrt{2 \cdot \pi \cdot n}\left(\frac{n}{e}\right)^n\left(1 + \Theta\left(\frac{1}{n}\right)\right)$$
  - From Stirling's approximation, the following holds:
    - $n! = \boldsymbol{o}(n^n)$
    - $n! = \boldsymbol{\omega}(2^n)$
    - $\lg(n!) = \boldsymbol{\Theta}(n \lg n)$

Comparison of Running Times

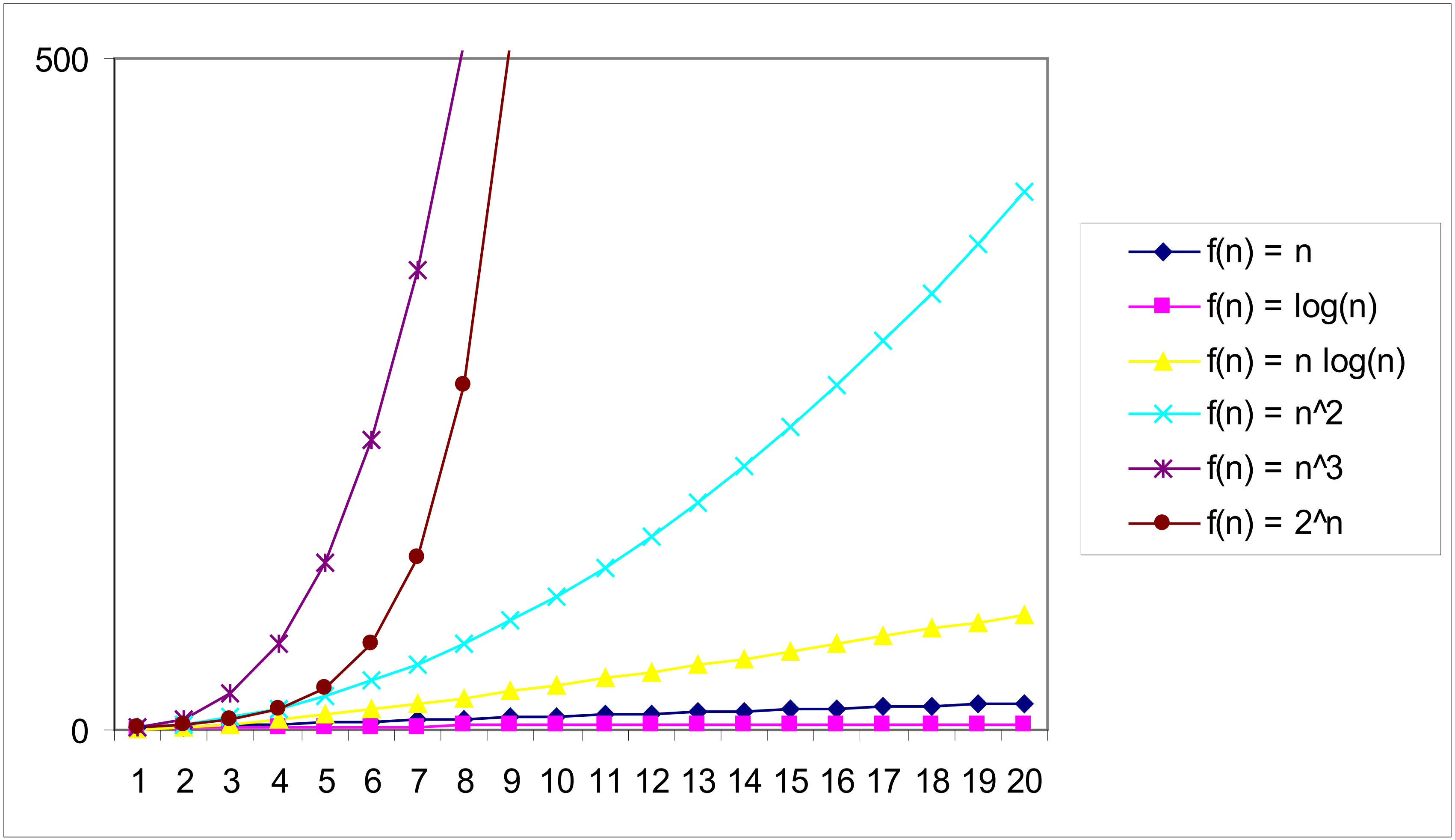
	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long



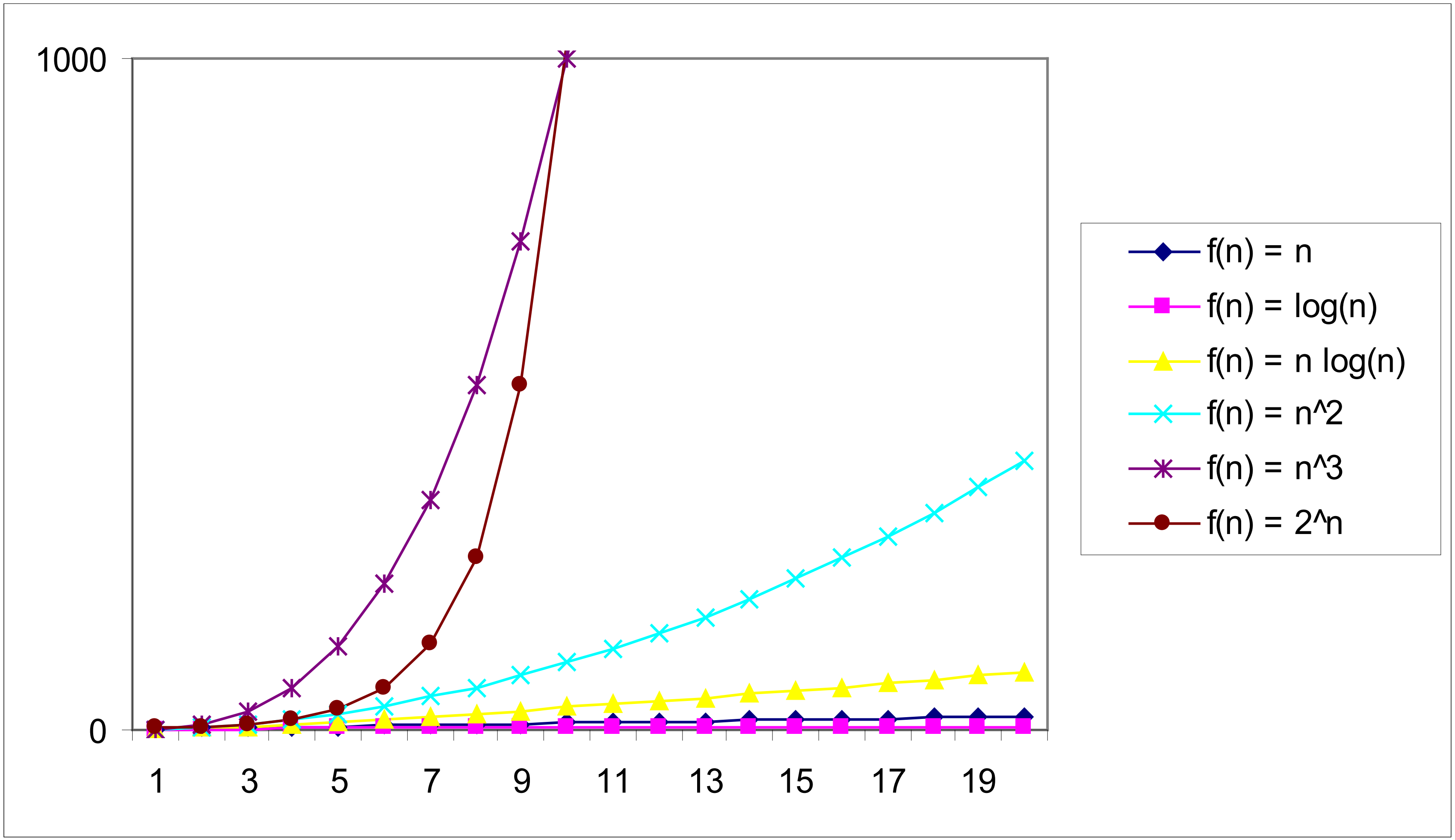
Practical Complexity



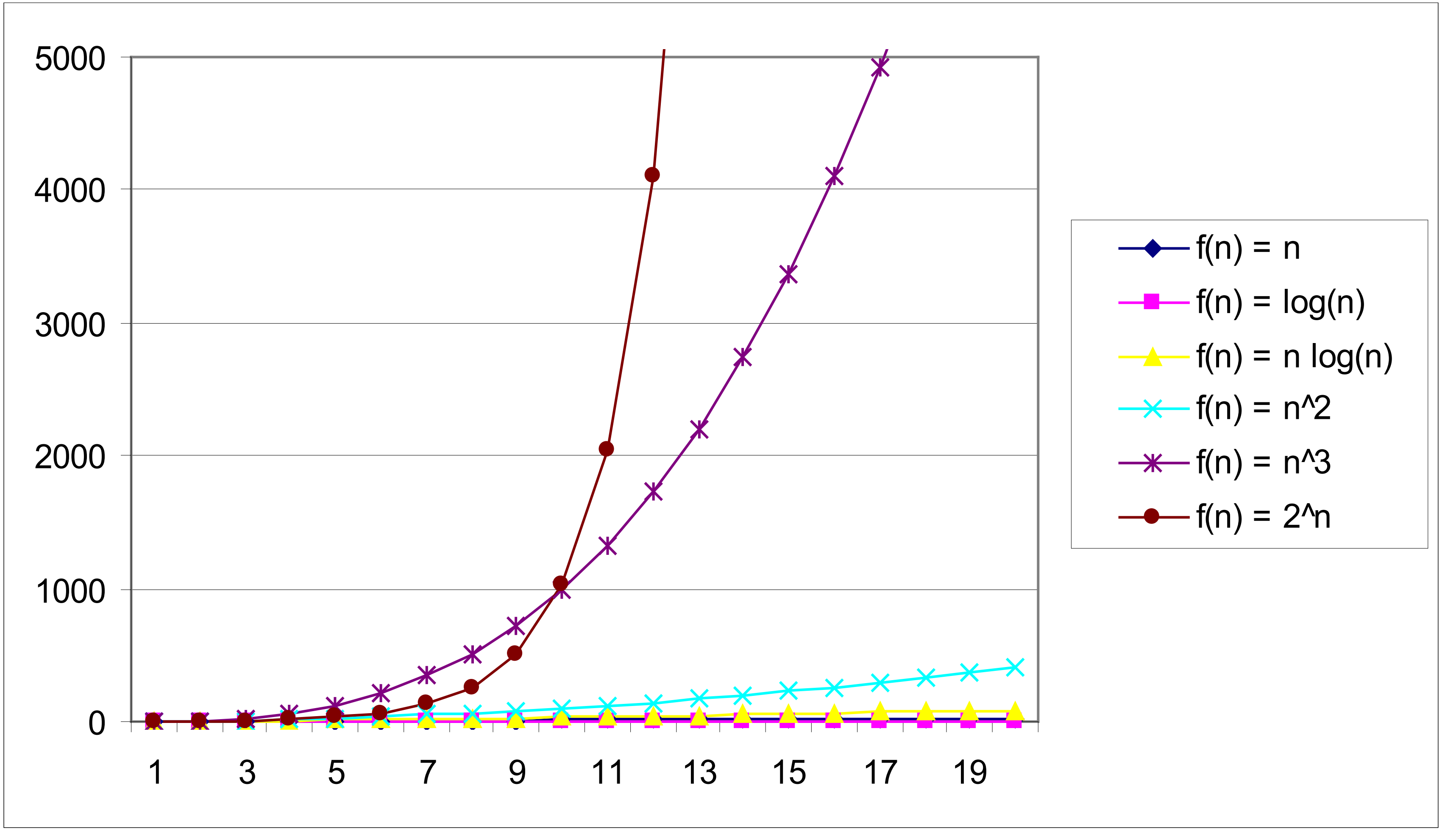
Practical Complexity



Practical Complexity



Practical Complexity



Common Time Complexities

BETTER

$O(1)$	constant time
$O(\log n)$	log time
$O(n)$	linear time
$O(n \log n)$	log linear time
$O(n^2)$	quadratic time
$O(n^3)$	cubic time
$O(n^k)$	polynomial time
$O(2^n)$	exponential time

WORSE