BIG OH NOTATION

Classification of Algorithms

The running time of most algorithms is proportional to one of the following functions :

`constant`	instructions run only once
`log N`	solve a big problem by transforming it into a smaller problem
`N`	each input element is processed
`N log N`	solve a problem by breaking it into a number of smaller problems, solve them independently, and combine the solutions
N^2	process all pairs of data items
2^N	brute-force

Polynomial vs. Exponential

f(n)	size n 10	20	30	40	50
n	.00001 sec	.00002 sec	.00003 sec	.00004 sec	.00005 sec
n^2	.0001 sec	.0004 sec	.0009 sec	.0016 sec	.0025 sec
n^3	.001 sec	.008 sec	.027 sec	.064 sec	.125 sec
n^5	.1 sec	3.2 sec	24.3 sec	1.7 min	5.2 min
n^n					10^8

Polynomial vs. Exponential

Time complexity	present computer	100 times faster	1000 times faster
n	T_1	$100\ T_1$	$1000\ T_1$
n^2	T_2	$10\ T_2$	$31.6\ T_2$
n^3	T_3	$4.64\ T_3$	$10\ T_3$
n^5	T_4	$2.5\ T_4$	$3.98\ T_4$
2^n	T_5	$T_5 + 6.64$	$T_5 + 9.97$
3^n	T_6	$T_6 + 4.19$	$T_6 + 6.29$

Asymptotics

- The study of functions of a parameter n, as n becomes larger and larger without bound.
- Frequency of basic actions is much more important than a total counts of all operations including housekeeping.
  - Houskeeping is too dependent on
    - programming language
    - programmer's particular style
- Change in fundamental method can make a vital difference (e.g. sequential vs. binary search).

Big Oh Notation

Algorithms	จำนวนการเปรียบเทียบเฉลี่ย		
	กรณีที่หาพบ	กรณีที่หาไม่พบ	
Sequential	0.5(n+1)	n	O(n)
Binary1	$\log_2 n + 1$	$\log_2 n + 1$	O(log n)
Binary2	$2\log_2 n - 3$	$2\log_2 n$	O(log n)

Big Oh Notation

Definition

If f(n) and g(n) are functions defined for positive integers, then

$$f(n) \text{ is } O(g(n))$$

means than there exists a constant c such that

$$|f(n)| \leq c\,|g(n)|$$

for all sufficiently large positive integers n

Example :

$$2n + 4n - 6 \rightarrow O(n)$$

$$7n^3 - 4n + 1 \rightarrow O(n^3)$$

$$2^n + 4n^2 - 7 \rightarrow O(2^n)$$

Growth Rates of Common Functions