

Chapter 4

Mathematical Foundations

* 7 most important functions :-

- ① Constant: 1 ② Logarithm: $\log_2 n$ ③ Linear: n ④ $N \cdot \log N$: $n \log n$
 ⑤ quadratic: n^2 ⑥ Cubic: n^3 ⑦ exponent: a^n

1 Constant function, $f(n) = c$

$c \rightarrow$ fixed constant $c=5, c=27, c=2^{10}$

$\rightarrow$ most fundamental constant fn. $\rightarrow \boxed{g(n)=1}$ typical constant fn.

$f(n) = c g(n) \Leftarrow c, g(n) \rightarrow$ constant fn. si *

adding 2 numbers $\leftarrow$ $\sim n^1$ *

Comparing " " $\leftarrow$

assigning a value to var. $\leftarrow$

2 The Logarithm function, $f(n) = \log_b n$

$\rightarrow x = \log_b n \Leftrightarrow b^x = n$; b : base of algor. $b > 1$, Common base $\boxed{b=2}$

$\rightarrow \log_2 n = \log n$

$\rightarrow$ Compute the smallest integer greater than or equal $\log_a n$

* $\log_3 27 = 27/3/3/3 = 1 \rightarrow \boxed{\therefore \log_3 27 = 3}$

* $\log_4 46 = 46/4/4/4/4 = 1 \rightarrow \boxed{\therefore \log_4 46 = 4}$

* $\log_2 12 = 12/2/2/2/2 = 0.75 \leq 1 \rightarrow \boxed{\therefore \log_2 12 = 4}$

* Logarithm Rules

① $\log a c = \log a + \log c$

② $\log a/c = \log a - \log c$

③ $\log a^c = c \log a$

④ $\log_b a = \log_d a / \log_d b$

⑤ $\log^{\log a} a = a^{\log_d b}$

Examples

$$\log(2n) = \log 2 + \log n = 1 + \log n$$

$$\log(n/2) = \log n - \log 2 = \log n - 1$$

$$\log n^3 = 3 \log n$$

$$\log_2^n = n \log 2 = n \cdot 1 = n$$

$$\log_4 n = \log n / \log 4 = \log n / 2$$

$$2^{\log n} = n^{\log 2} = n^1 = n$$

③ Linear function, $f(n) = n$

→ doing a single basic operation for each of n elements.

④ The N -Log- N function, $f(n) = n \log n$

→ grows little faster than the linear fn. and a lot slower than the quadratic fn.

⑤ The Quadratic function, $f(n) = n^2$

→ for nested loops

- outer loop: linear fn. $\odot n$
- inner loop: linear fn. $\odot n$

→ $n \cdot n = n^2$

→ $\times$ of operation performed by nested loop if the $\times$ of operation performed inside the loop increase by one with each iteration of the outer loop →

$$1 + 2 + 3 + \dots + (n-2) + (n-1) + n = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}$$

$\frac{1}{2} \rightarrow \text{constant factor}$

$\downarrow$
 n^2

⑥ The Cubic function, $f(n) = n^3$

linear fn. → Constant time

⑦ Exponential function, $f(n) = b^n$

common base $\rightarrow 2 \rightarrow f(n) = 2^n$

④ Exponent Rules

$$(1) (b^a)^c = b^{a \cdot c}$$

$$(2) b^a \cdot b^c = b^{a+c}$$

$$(3) b^a / b^c = b^{a-c}$$

* Comparing Growth Rates

$$1 < \log n < n < n \log n < n^2 < n^3 < a^n$$

* Analysis of Algorithm

→ Experimental Study

ناتج التنفيذ ووقت التنفيذ method ووقت التنفيذ inputs ووقت التنفيذ
هذا الوقت لا يمكن أن يكون implementation كل ال code ووقت
ما يقدر لكل inputs
عنه 2 algorithms ووقت تنفيذ في hardware
software "

→ Theoretical Analysis

الطريقة التي التي algorithm في pseudocode ووقت
operation time ال primitive operation

* pseudocode → high level description of an algorithm

* primitive operations

- ① Arithmetic operation : $+, -, *, /$
- ② Comparing : $a < b$
- ③ Assigning a value to a var. : $i = 0$
- ④ Indexing into an array : $A[i]$
- ⑤ Calling a method : $Reverse()$
- ⑥ Returning from a method : $return X$

R 4.18

primitive operation	# of operations	Executed time	$O()$
$S \leftarrow A[0]$	2 : Assigning + indexing	1	$O(1)$
$t \leftarrow S$	1 : Assigning	1	$O(1)$
$i \leftarrow 1$	1 : Assigning	1	$O(1)$
$i < n-1$	1 : Comparing	$\frac{n}{1 \rightarrow n-1}$	$O(n)$
$S \leftarrow S + A[i]$	3 : Assigning + indexing + sum	$n-1$	$O(n)$
$t \leftarrow t + S$	2 : Assigning + sum	$n-1$	$O(n)$
$i++$	2 : Assigning + sum	$n-1$	$O(n)$
return t	1 : return	1	$O(1)$

Run time = $2 + 1 + 1 + n + 3(n-1) + 2(n-1) + 2(n-1) + 1 = 8n - 2 \rightarrow \underline{\underline{O(n)}}$

4.17

primitive operation	# of operations	Executed time	$O()$
$S \leftarrow 0$	1 : Assigning	1	$O(1)$
$i \leftarrow 0$	1 : Assigning	1	$O(1)$
$i < n-1$	1 : Comparing	$\frac{0 \rightarrow n-1}{n+1}$	$O(n)$
$S \leftarrow S + A[0]$	3 : Assigning + Indexing + sum	n	$O(n)$
$j \leftarrow 1$	1 : Assigning	n	$O(n)$
$j < i$	1 : Comparing	$1+2+3+\dots+n = \frac{n(n+1)}{2-n}$	$O(n^2)$
$S \leftarrow S + A[i]$	3 : Assigning + Indexing + sum	$\frac{n(n+1)}{2-n}$	$O(n^2)$
$j++$	2 : Assigning + sum	$\frac{n(n+1)}{2-n}$	$O(n^2)$
$i++$	2 : Assigning + sum	n	$O(n)$
return t	1 : return	1	$O(1)$

$\Rightarrow \therefore O(n^2)$

4

* Big-Oh Notation

Given functions $f(n)$, $g(n) \rightarrow$ we say that ' $f(n)$ is $O(g(n))$ '
" $f(n)$ is big-oh of $g(n)$ "
" $f(n)$ is order of $g(n)$ "

if there is a real constant $c > 0$ and an integer constant $n_0 \geq 1$
such that $f(n) \leq cg(n)$ for $n \geq n_0$

* Big-Oh Examples

① $8n-2$ is $O(n)$

$$f(n) \leq cg(n)$$

$$8n-2 \leq cn$$

$$c=8, n_0=1$$

② $5n^4+3n^3+2n^2+4n+1$ is $O(n^4)$

$$5n^4+3n^3+2n^2+4n+1 \leq 15n^4 = cn^4$$

$$c=15, n_0=1$$

③ $5n^2+3n\log n+2n+5$ is $O(n^2)$

$$5n^2+3n\log n+2n+5 \leq 15n^2 = cn^2$$

$$c=15, \underline{n_0=2} \text{ as } \log 1 = \text{zero}$$

④ $20n^3+10\log n+5$ is $O(n^3)$

$$20n^3+10\log n+5 \leq 35n^3 = cn^3$$

$$c=35, n_0=2$$

⑤ $3\log n+2$ is $O(\log n)$

$$3\log n+2 \leq 5\log n$$

$$c=5, n_0=2$$

⑥ 2^{n+2} is $O(2^n)$

$$2^{n+2} = 2^n \cdot 2^2 = 4 \cdot 2^n$$

$$c=4, n_0=1$$

⑦ $2n+100\log n$ is $O(n)$

$$2n+100\log n \leq 102\log n$$

$$c=102, n_0=2$$

⑧ $f(n) = 1000$

$$1000 \leq c \cdot 1$$

$$c=1000, n_0=1$$

* Big-Omega

$f(n)$ is $\Omega(g(n))$ if $g(n)$ is $O(f(n))$

there are $c > 0$, $n_0 \geq 1$ such that $f(n) \geq cg(n)$ for $n \geq n_0$

* Big-Theta

$\rightarrow f(n)$ is $\Theta(g(n))$ if $f(n)$ is asymptotically equal to $g(n)$

$\rightarrow f(n)$ is $\Theta(g(n))$ if there are constants $c' > 0$, $c'' > 0$, $n_0 \geq 1$
such that $c'g(n) \leq f(n) \leq c''g(n)$

More Examples

For each of the following pairs of functions, either $f(n)$ is $O(g(n))$, $f(n)$ is $\Omega(g(n))$, or $f(n) = \Theta(g(n))$. Determine which relation relationship is correct.

- ① $f(n) = \log n^2$ $g(n) = \log n + 5$ $f(n) = g(n)$ $f(n) = \Theta(g(n))$
- ② $f(n) = n$ $g(n) = \log n^2$ $f(n) > g(n)$ $f(n) = \Omega(g(n))$
- ③ $f(n) = \log \log n$ $g(n) = \log n$ $f(n) < g(n)$ $f(n) = O(g(n))$
- ④ $f(n) = n$ $g(n) = \log^2 n$ منفردة لا تربيع $f(n) = \Omega(g(n))$
- ⑤ $f(n) = n \log n$ $g(n) = \log n$ $f(n) = \Omega(g(n))$
- ⑥ $f(n) = 10$ $g(n) = \log 10$ both are constants $f(n) = \Theta(g(n))$
- ⑦ $f(n) = 2^n$ $g(n) = 10n^2$ $f(n) = \Omega(g(n))$
- ⑧ $f(n) = 2^n$ $g(n) = 3^n$ $f(n) = O(g(n))$

Each of the following lines gives the runtime of an algorithm for its performance on two different numbers of data sets. Please characterize the runtime in terms of Big-O

Algorithm	#elements = 1000	#elements = 4000	Runtime
1	10 seconds $\frac{1000}{10} = 100$	160 seconds $\frac{4000}{160} = 25$	$O(n^2)$
2	10 seconds 100	640 seconds 6	$O(n^3)$
3	1 seconds 1000	4 seconds 1000	$O(n)$
4	4 seconds 250	4 seconds 1000	$O(1)$