Chapter 1

Introduction to Algorithm.

1. Introduction to Algorithms

Algorithm → design   require Domain Knowledge

Program → implementation   by Programmer

1.1 Priori Analysis vs Posteriori Testing

Priori Analysis	Posteriori Testing
Algorithm	Program
Independent of lang	Language dependent
Independent of HW	HW dependent
Time & Space func	watch time & bytes

1.2 Characteristics of Algorithms

1) Input → 0 or more
2) Output → at least 1 output
3) Definiteness → Should be clear to computer.
4) Finiteness
5) Effectiveness

1.3 How to Analyze an algorithm

1) Time
2) Space
3) Network → How many requests
4) Power → How much power consuming
5) CPU registers → How many registers consuming

ex) SWAP FUNC

```
swap (a, b)
{
    temp = a;
     a = b;
     b = temp;
}
```

	time	space
temp = a;	1	a, b, temp
a = b;	1	
b = temp;	1	
	$f(n) = 3$, $O(1)$	$g(n) = 3$, $O(1)$

1.4 Frequency Count Method

```
Algorithm Sum ( A, n )
{
    s = 0;                              — 1
    for ( i = 0 ; i < N ; i++ )         — n+1
        s = s + A[i];                   — n
    return s;                           — 1
}
```

	time	space
`s = 0;`	1	A - n
`for ( i = 0 ; i < N ; i++ )`	n+1	n - 1
`s = s + A[i];`	n	s - 1
`return s;`	1	i - 1

time: $f(n) = 2n+3$, $O(n)$

space: $S(n) = n+3$, $O(n)$

1.5 Time complexity

ex)
```
for (i=1; i<n; i=i*2)
{
    something;
}
```

i
1
1×2
2×2
$2^2\times 2$
$\vdots$
2^k

Assume $i \geq n$

$\because i = 2^k$

$2^k \geq n$

$k \geq \log_2 n \quad \therefore O(\log_2 n)$

cf. other impl

$i = 1\times 2\times 2\times \cdots \geq n$

$2^k \geq n$

$k \geq \log_2 n$

ex2) for (i = n ; i >= 1 ; i = i/2) { smt }

$i \rightarrow n, \; n/2, \; n/2^2, \; \cdots, \; n/2^k$

Assume $i < 1 \rightarrow \frac{n}{2^k} < 1$

$n < 2^k$ $\qquad O(\log_2 n)$

$\log_2 n < k$

ex3) for (i = 0 ; i * i < n ; i++) { smt }

Assume, $i^2 > n$, $i = k$

$k^2 > n$ $\qquad O(\sqrt{n})$

ex4) p = 0

for (i = 1 ; i < n ; i = i * 2) { p+1 }

for (j = 1 ; j < p ; j = j * 2) { smt }

$p = \log_2 n$

$\longrightarrow \log_2 p \rightarrow O(\log_2 \log_2 n)$

$\text{for } (i = 0;\ i < n;\ i{+}{+})$ $\quad O(n)$

$\text{for } (i = 0;\ i < n;\ i = i + 2)$ $\quad O(n)$

$\text{for } (i = n;\ i > 1;\ i{-}{-})$ $\quad O(n)$

$\text{for } (i = 1;\ i < n;\ i = i \times 2)$ $\quad O(\log_2 n)$

$\text{for } (i = 1;\ i < n;\ i = i \times 3)$ $\quad O(\log_3 n)$

$\text{for } (i = n;\ i > n;\ i = i / 2)$ $\quad O(\log_2 n)$

1.6 Types of time complexity

$O(1)$ — constant

$O(\log n)$ - Logrithemic

$O(n)$ - Linear

$O(n^2)$ - Quadratic

$O(n^3)$ - cubic

$O(2^n)$ - exponential

1.7 compare class of Functions

$$1 < \log_2 n < \sqrt{n} < n < n\log_2 n < n^2 < n^3 < \cdots < 2^n < 3^n < \cdots$$

1.8 Asymptotic Notations Big Oh

O big-oh upper-bound

Ω big-omega lower bound

Θ theta average bound

① Big-Oh $f(n) = O(g(n)) \rightarrow f(n) \leq c * g(n) \; \forall \; n \geq n_0$

② Omega $f(n) = \Omega(g(n)) \rightarrow f(n) \geq c * g(n) \; \forall \; n \geq n_0$

③ Theta $f(n) = \Theta(g(n)) \rightarrow c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

1.9 Properties of Asymptotic Notation

General Properties

if $f(n)$ is $O(g(n))$ then $a * f(n)$ is $O(g(n))$ cf. same for all three notations

Reflexive Properties

if $f(n)$ is given then $f(n)$ is $O(f(n))$

Transitive Properties

if $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$

then $f(n) = O(h(n))$

Symmetric Properties Θ only

if $f(n)$ is $\Theta(g(n))$ then $g(n)$ is $\Theta(f(n))$

Transpose Symmetric O. Ω only

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

1.10 Comparison of functions

Logarithm

1. $\lg a \cdot b = \lg a + \lg b$

2. $\lg \frac{a}{b} = \lg a - \lg b$

3. $\lg a^{b} = b \lg a$

4. $a^{\lg_c b} = b^{\lg_c a}$

5. $a^{b} = n \rightarrow b = \lg_a n$