FREEMIND

Date

Page

Algorithm

Algorithm
Design
Domain knowledge
any language
Hardware & software O/S independent
Analysis

Program
Implementation
Programmer
Programming language
dependent upon hardware & software O/S
Testing

Priori analysis
1) done on algorithm
2) independent of language
3) hardware independent
4) Time & space function

Posteriori testing
1) done on program
2) dependent on the langu
3) hardware dependent
4) watch time & Bytes

Characteristics of Algorithm (Algorithm is like procedure)

1) Input → Algorithm can take 0 or more input
2) Output → must generate one output
3) Definiteness → every statement should be single & exact meaning (it can be solved)
   Example we can't use $\sqrt{-1}$ as real no.
4) finiteness → must be finite set of statement
5) effectiveness

FREEMIND

Date

Page

How to write an algorithm

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

how to analyze an Algorithm

1) time → it must be fast
2) space
3) N/W → data transfer or network connection
4) Power consumption
5) CPU registers

time analysis

Suppose every statement takes 1 unit of time

```
Algorithm swap(a, b)
{
    temp = a;   — 1
    a = b;      — 1
    b = temp;   — 1
}
```

$f(n) = 3$ $O(1)$

order of 1 ⇒ Constant

Space analysis

a = 1
b = 1
c = 1
→ 3 variables used

$f(n) = 3$

$O(1)$ → order of 1 constant

FREEMIND

Date

Page

Frequency count method

```
Algorithm sum(A, n)
{
    s = 0                          → 1
    for(i=0; i<n; i++)             → n+1
    { s = s + A[i];                → n times
    }
    return s;                      → 1
}
```

A[8 | 3 | 9 | 7 | 2]
0 1 2 3 4

n = 5 ⇓ array of 5 element

condition checked for n+1 times

$f(n) = 1 + n+1 + n + 1$

$\Rightarrow (2n+3)$

$O(n)$ ↓ order of n

Space → A, n, s, i

n words, 1, 1, 1

$\Rightarrow (n+3)$ & $O(n^2)$

$f(n) = n+3$

```
Algorithm Add(A, B, n)
{
    for(i=0; i<n; i++)             → n+1
    {for j=0; j<n; j++)            → n X (n+1)
        { C(i,j) = A(i,j) + B[i,j]; → n x n
        }
    }
}
```

{ Space ⇒ A, B, C, n, i, j
Complexity: n^2, n^2, n^2, 1, 1, 1 } $(3n^2+3)$

second order $\Leftarrow O(n^2)$

$f(n) \Rightarrow n+1 + n^2 + n + n^2$

$f(n) \; (2n^2 + 2n + 1)$

Date
Page

Algorithm multiply

```
Algorithm multiply (A, B, n)
{
n+1 —— for (i=0; i<n; i++)
n(n+1) —— {
n(n) —— for (j=0; j<n; j++)
(n+1)n(n) —— {
(n)n(n) —— C(i,j) = C(i,j) + A(i,k) * B(k,j)
}
}
}
```

Time →

$f(n)$

$\Rightarrow n+1+n^2+n+n^2+n^2(n+1)+n^3$

$n+1+n^2+n+n^2+n^3+n^2+n^3$

$2n^3+3n^2+2n+1$

$O(n^3) \Rightarrow$ order of n^3

Space →

A	B	C	n	i	j	k
⇓	⇓	⇓	⇓	⇓	⇓	⇓
n^2	n^2	n^2	1	1	1	1

$S(n)$ $3n^2+4$

$O(n^2) \Rightarrow$ order of n^2

space complexity

Time $\Rightarrow O(n^3)$

Space $\Rightarrow O(n^2)$

Date

Page

for (i=0; i<n; i++) → n+1

① {

something; → n

}

$f(n) = 2n+1 \Rightarrow O(n)$ ⇒ order of n

for (i=n; i>0; i--) → n+1

{ something: → n

}

O(n)

for (i=1; i<n; i = i+2) → n/2 + 1

{ Something; → n/2

}

O(n) ⇒ order of n

② for (i=0; i<n; i++) → n+1

{ for (j=0; j<n; j++) → n(n+1)

{ something } → n(n)

}

$n^2 + n^2 + n + n + 1$

$\Rightarrow 2n^2 + 2n + 1$

$O(n^2)$ ⇒ order of n^2

FREEMIND

Date

Page

(8)
```
for (i=0; i<n; i++)
{
    for (j=0; j<i; j++)
    {
        something;
    }
}
```

i	j	no. of times
0	0	0
1	0	1
2	0,1	2
3	0 1 2 3	3
⋮	⋮	⋮
n		n

$1+2+3+\cdots+n \Rightarrow \frac{n(n+1)}{2}$

$f(n) \Rightarrow O(n^2)$ order of n^2

(9)
```
p=0
for (i=0; p<n; i++)
{
    p = p+i;
}
```

n	i	p
1	0	0
2	1	1
	2	3
	3	6
	⋮	⋮
	k	$\frac{k(k+1)}{2}$

$p + n < n$

FREEMIND

Date

Page

④ `p = 0`
`for (i=1; p<=n; i++)`
`{`
`p = p + i;`
`}`

i	p
1	1
2	1+2
3	1+2+3
.	.
.	.
k	1+2+3 ... k

to tak koega ye
tk p<n nahoga ye

$\frac{k(k+1)}{2} \not< n$

$k^2 > n$

$k > \sqrt{n}$

$O(\sqrt{n})$

⑤ `for (i=1; i<n; i=i*2)`
`{`
`Smt;`
`}`

$i = 1$
$i = 2$
$i = 4$
$i = 2^3$
.
.

$2^k < n$

$k < \log_2(n)$

let k works

terminates when $k \geq \log_2 n$

$O(\log_2(n)) \Rightarrow$ order

FREEMIND

Date

Page

$\lceil \log n \rceil$ → ceil value leta hai

for eg

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

2^0 1 2 2^2 ---

$2^k = n$

$\log_2 n$

take $n = 8$

take $n = 10$

i
1
2
4
8

→ 3 times

i
1
2
4
8
10

→ 4 times

$\log_2 8 \Rightarrow 3$

$\log_2 10 = 3.2$

so take ceil value

$\lceil \log_2 n \rceil$

FREEMIND

Date

Page

⑥ for(i = n; i >= 1; i = i/2)
{ something }

i
n
n/2
n/4
.
.
1

n/2^k

loop will continue till

n/2^k >= 1

will stop when

n/2^k < 1

n >= 2^k

k = log_2 n

O(log_2(n))

⑦ for(i=0; (i*i)<n; i++)
{ something }

i + i < n
1 < n
4 < n
.
.
1

k^2 < n

√n

O(√n)

FREEMIND

Date

Page

(3)
```
for (i=0; i<n; i++)
{
  something;        → n
}
for (j=0; j<n; j++)
{
  something         → n
}
```

N + n → 2n ⇒ O(n)

(4)
```
p=0
for (i=1; i<n; i=i*2)
{
  p++;              → log n
}
for (j=1; j<p; j=j*2)
{
  something         ⇒ log2(p)
}
```

1 2 —— 2^k

$2^k \not< p$

$k = \log_2(p)$

$p = \log n$

$k = \log(\log(n))$

$O(\log(\log(n)))$

FREEMIND

Date

Page

(10)

```
for(i=0; i<n; i++)          → n
{
  for j=1; j<n; j=j*2)      → n(log n)
  {
    something               → n (log n)
  }
)
```

⊛ $f(n) \Rightarrow 2n\log(n) + n$

$O(n\log(n))$

for $(i=0; i<n; i++)$ → $O(n)$

for $(i=0; i<\frac{n}{2}; i++)$ → $O(n)$ ⇒ $\frac{n}{2}$

for $(i=n; i>1; i--)$ ⇒ $O(n)$

for $(i=1; i<n; i=i*2)$ ⇒ $O(\log_2 n)$

for $(i=1; i<n; i=i*3)$ ⇒ $O(\log_3 n)$

for $(i=n; i>1; i=\frac{i}{2})$ ⇒ $O(\log_2(n))$

FREEMIND

Date

Page

Types of time functions

$O(1)$ ⇒ constant

If $f(n) = 2$
$f(n) = 5$
$f(n) = 5000$
→ $O(1)$ ⇒ constant

$O(\log n)$ ⇒ logarithmic

$O(n)$ ⇒ linear

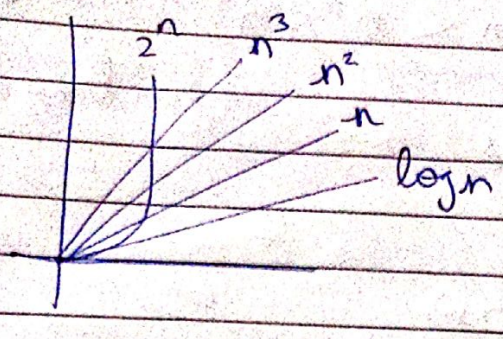
$f(n) = 2n+3 \rightarrow O(n)$
$500n + 700 \nearrow$
$\frac{n}{5000} + 6 \nearrow$

$O(n^2)$ ⇒ quadratic
$O(n^3)$ ⇒ cubic
$O(2^n)$ ⇒ exponential

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

$\log n$	n	n^2	2^n
0	1	1	2
$\log_2 2 = 1$	2	4	4
$\log_2 8 = 3$	8	64	2^8

$n^k < 2^n$ ⇒ at large value of n



Asymptotic notation

O big-oh upper bound of the function

Ω big-omega lower bound of the function

Θ theta average bound of the function

Big-Oh notation

The function $f(n) = O(g(n))$ iff $\exists$ the constants c & n_0 such that $f(n) \leq c \cdot g(n)$ $\forall$ $n \geq n_0$

eg $f(n) = 2n + 3$

$2n+3 \leq 7n$

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3$ $f(n) = O(n)$

$< 2^n < 3^n \ldots$ Big O-notation

lower bound — average bound — upper bound

$2n + 3 \leq 2n + 3n$

$2n+3 \leq 5n$ $(n \geq 1)$

also

$2n+3 \leq 2n^2 + 3n^2$

$5n^2 \Rightarrow O(n^2)$

$f(n) \Rightarrow O(n)$

$g(n) = O(n^4)$

$f(n) = O(2^n)$

$f(n) = O(\log n)$

$f(n) = O(n)$

$f(n) = O(n^2)$

FREEMIND

Date

Page

$< \log n < \sqrt{n} < n$ $< n \log n < n^2 < n^3 \dots > 2^n < 3^n \dots < n^n$

lower bound = average bound = upper bound

in above qus

$f(n) = O(n)$ ✓

$f(n) = O(n^2)$ ✓

$f(n) = O(2^n)$ ✓

but

~~$f(n) = O(\log(n))$~~ ⇒ wrong

in writing Big-O notation write the closest function

Omega notation

The function $f(n) = \Omega(g(n))$ iff ∃ +ve constants c & n_0 such that $f(n) \geq c * g(n)$ $\forall n \geq n_0$

$f(n) = 2n + 3$

$2n + 3 \geq 1 \times n$ $\forall n \geq 1$

$f(n)$ $c(g(n))$

$f(n) = \Omega(n)$

$2n + 3 \geq 1 \times \log n$

$f(n) = \Omega(\log n)$

but $f(n) \neq \Omega(n^2)$

FREEMIND

Date

Page

Theta notation

the function $f(n) = \Theta(g(n))$ iff there exists c_1, c_2 and n_0 such that

$$c_1(g(n)) \leq f(n) \leq c_2(g(n))$$

eg

$$f(n) = 2n+3$$

$$1n \leq 2n+3 \leq 5n$$

$c_1(g(n)) \quad f(n) \quad c_2(g(n)) \qquad f(n) = \Theta(n)$

eg $f(n) = 2n^2 + 3n + 4$

$$2n^2 + 3n + 4 \leq 2n^2 + 3n^2 + 4n^2$$

$$2n^2 + 3n + 4 \leq 9n^2$$

$$2n^2 + 3n + 4 \leq c_0(g(n))$$

so, $f(n) = O(n^2)$

also

$$2n^2 + 3n + 4 \geq 1n^2$$

$$f(n) = \Omega(n^2)$$

$$1n^2 \leq 2n^2 + 3n + 4 \leq 9n^2$$

$$f(n) = \Theta(n^2)$$

eg $f(n) = n^2 \log n + n$

$$n^2 \log n \leq n^2 \log n + n \leq 10 n^2 \log n$$

$O(n^2 \log n)$, $\Omega(n^2 \log n)$, $\Theta(n^2 \log n)$

Date

Page

eg $f(n) = n!$

$n! = n(n-1)(n-2) \cdots 3 \times 2 \times 1$

$1 \times 1 \times 1 \times 1 \times 1 \leq 1 \times 2 \times 3 \cdots n \leq n \times n \times n \cdots$

lower bound $1 \leq n! \leq n^n$ ⇒ upper bound

$\Omega(1)$, $\Theta(n!)$, $O(n^n)$

eg $f(n) = \log(n!)$

$\log(1 \times 1 \times 1) \leq \log(1 \times 2 \times 3 \cdots n) \leq \log(n \times n \times n \times n \times n)$

$1 \leq \log n! \leq \log(n^n)$

$1 \leq \log n! \leq n \log n$

$\Omega(1)$ $O(n \log n)$

Θ theta is more accurate

Ω & O are used when we're not sure it can be lesser or greater than respectively

Date

Page

Properties of Asymptotic notation

General property

- if $f(n)$ is $O(g(n))$ then $a \cdot f(n)$ is $O(g(n))$ (a: constant)

eg if $f(n) = 2n^2 + 5$ is $O(n^2)$

then

$7f(n) \Rightarrow 14n^2 + 35$ also $O(n^2)$

it'll be true for Ω & Θ also

Reflexive property

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

eg $f(n) = n^2$ $O(n^2)$

Transitive property

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

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

eg $f(n) = n$, $g(n) = n^2$, $h(n) = n^3$

n is $O(n^2)$ and n^2 is $O(n^3)$

then n is $O(n^3)$

true for Ω & Θ also

FREEMIND

Date

Page

Symmetric property

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

eg $f(n) = n^2$ and $g(n) = n^2$

$f(n) = \Theta(n^2)$

and

$g(n) = \Theta(n^2)$

true only in case of Θ notation

Transpose symmetric (true for Ω & O notation)

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

eg $f(n) = n$, $g(n) = n^2$

then n is $O(n^2)$ and

n^2 is $\Omega(n)$

if $f(n) = O(g(n))$

and

$f(n) = \Omega(g(n))$

$g(n) \leq f(n) \leq g(n)$

$f(n) = \Theta(g(n))$

if $f(n) = O(g(n))$ and $d(n) = O(e(n))$

then

$f(n) + d(n) = O(\max(g(n), e(n)))$

FREEMIND

Date

Page

Eg $f(n) = n \Rightarrow O(n)$

$d(n) = n^2 \Rightarrow O(n^2)$

$f(n) + d(n) = n + n^2 \Rightarrow O(n^2)$

if $f(n) = O(g(n))$

and

$d(n) = O(e(n))$

then $f(n) * d(n) \Rightarrow O(g(n) * e(n))$

$n \cdot n^2 \qquad O(n^3)$

Comparison of functions

n	n^2	n^3
2	$2^2 = 4$	$2^3 = 8$
3	$3^2 = 9$	$3^3 = 27$

$4 < 8$

$9 < 27$

$n^2 < n^3$

first method ⇒ values leke dekho

second method ⇒ apply log on both side

(useful in complex function)

$\log n^2 \qquad \log n^3$

$2 \log n < 3 \log n$

FREEMIND

Date

Page

$\log ab = \log a + \log b$

$\log \frac{a}{b} = \log a - \log b$

$\log a^b = b \log a$

$a^{\log_c b} \Rightarrow b^{\log_c a}$

$a^b = n$ then $b = \log_a n$

Eg $f(n) = n^2 \log n$ and $g(n) = n(\log n)^{10}$

take log on both side

$\log(n^2 \log n)$ $\qquad$ $\log(n(\log n)^{10})$

$2\log(n) + \log(\log n)$ $\qquad$ $\log n + 10 \log(\log n)$

$f(n) > g(n)$

Eg $f(n) = 3n^{\sqrt{n}}$ $\qquad$ $g(n) = 2^{\sqrt{n}\log n}$

$3n^{\sqrt{n}}$ $\qquad$ $2^{\log_2 n^{\sqrt{n}}}$

$3 n^{\sqrt{n}}$ $\qquad$ $n^{\sqrt{n}}$

$f(n) > g(n)$

FREEMIND

Date

Page

eg $f(n) = n^{\log n}$ $g(n) = 2^{\sqrt{n}}$

Apply log

$\log n^{\log n}$ $\log 2^{\sqrt{n}}$

$(\log n)(\log n)$ $\sqrt{n} \log_2 2$

$\log^2(n)$ $\sqrt{n}$

Apply log again if you aren't able to judge it

$\log(\log^2(n))$ $\log(\sqrt{n})$

$2\log(\log(n))$ $\frac{1}{2}\log n$

$\log(n) \neq \log(\log(n))$

$g(n) > f(n)$

eg $f(n) = 2^{\log n}$ and $g(n) = n^{\sqrt{n}}$

$\log n \times \log_2 2$ $\sqrt{n}\log n$

$\log n$ $\sqrt{n}\log n$

$g(n) > f(n)$

FREEMIND

Date

Page

eg $f(n) = 2^n$, $g(n) = 2^{2n}$

$\log(2^n)$ $\log(2^{2n})$

$n \log_2 2$ $2n \log_2 2$

1 1

n $2n$

$f(n) < g(n)$

$g_1(n)$ → n^3 $n < 100$

→ n^2 $n \geq 100$

$g_2(n)$ → n^2 $n < 10,000$

→ n^3 $n \geq 10,000$

0 —— 100 —— 10000 —— ∞

$g_1(n)$ $g_1 = g_2$ $g(2) > g_1$

$g_1(n) > g(2)$

$g_2 > g_1$ $n \geq 10,000$

we just dont check ans smaller value & decide

FREEMIND

Date

Page

True or false

1) $(n+k)^m = \Theta(n^m)$

True

$(n+3)^2 = \Theta(n^2)$

2) $2^{n+1} = O(2^n)$

True

$2 \cdot 2^n \Rightarrow \Theta(2^n), \Omega(2^n), O(2^n)$

3) $2^{2n} = O(2^n)$

$4^n > 2^n$ it cannot be upper bound

False

4) $\sqrt{\log n} = O(\log \log(n))$

$\frac{1}{2} \log(\log n)$ False

$\log \log \log n$ can't be upper bound

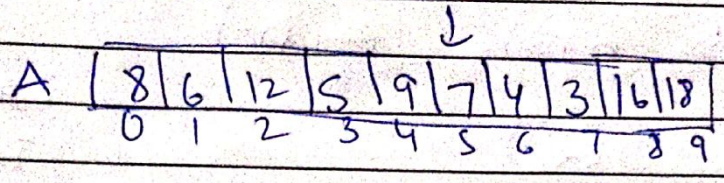
5) $n^{\log n} = O(2^n)$

$\log n \log n \quad n$ true as n is greater

Date
Page

Best, worst and Average case Analysis

I linear search



Key = 7

After 6 comparison we get our element

When we search for key element 20
20 is not present search unsuccessful

Best case → searching key element present at first index

Best case time = 1 (O(1))

constant (order of 1)

$B(n) = O(1)$

Worst case → searching a key at last index

worst case time = n → n element

$W(n) = O(n)$

Average case → $\frac{\text{all possible case time}}{\text{no. of cases}}$

$$\text{Average time} = \frac{1+2+3 \cdots \cdots n}{n}$$

$$\frac{\frac{n(n+1)}{2}}{n} \Rightarrow \frac{n+1}{2}$$

$B(n) = O(1)$

$W(n) = O(n)$

$A(n) = \frac{n+1}{2}$

FREEMIND

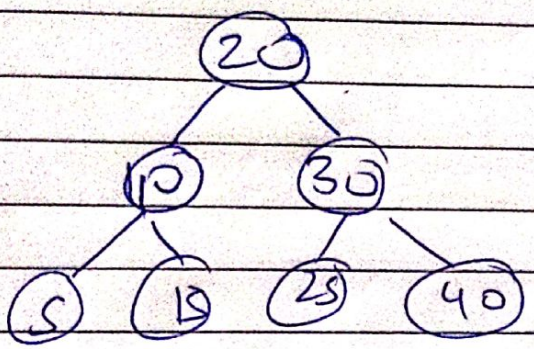
Date

Page

$B(n) = 1$
$B(n) = O(1)$
$B(n) = \Omega(1)$
$B(n) = \Theta(1)$

$W(n) = n$
$W(n) = O(n)$
$W(n) = \Omega(n)$
$W(n) = \Theta(n)$

II Binary search tree



$\log n$ → time taken

Best case ⇒ searching root element
Best case time = 1
$B(n) = 1$

Worst case ⇒ search for leaf element
(reach the root)
worst case time = $W(n) = h$

height of binary search tree

$\min(W(n)) = \log n$
$\max(W(n)) = n$