

①

22-08-2020Types of Time Complexity

① for ($i=0$, $i < n$, $i++$) $n+1$
 {
 state
 }
 ↓
 we need to
 how many time
 statements are
 running.

So, $O(n) \rightarrow$ time complexity

next,

②

for ($i=0$; $i < n$; $i = i+20$)
 {
 state
 }
 ↓
 len $n/20$ times

So, $O(n)$ again.

because, $f(n) = n/20$

* degree of polynomial is n
 $\Rightarrow O(n)$

②

③ for () --- $n+1$
 {
 for () --- $n(n+1)$
 {
 state --- $n \times n$
 }
 }

$f(n) = n^2 \rightarrow$ polynomial 2

So, $O(n^2) \rightarrow$ time complexity.

Complex Exmp: ④ for ($i=0; i < n; i++$)
 { for ($j=0; j < i; j++$)
 { state ---
 }
 }

i	j	no. of time
0	0	1
1	0	1
	1	2
2	0	1
	1	2
	2	3

So, total no
 of time -
 $1+2+3+\dots+n$
 $= \frac{n(n+1)}{2}$
 $\Rightarrow f(n) = \frac{n^2}{2} + \frac{1}{2}$
 and so on. $\Rightarrow O(n^2)$.

③

⑤ Complex:

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

When, loop will be stopped?

$p > n$

We know, $p = \frac{k(k+1)}{2}$

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

$\Rightarrow \frac{k^2 + k}{2} > n$

$\Rightarrow k^2 > n$

$\Rightarrow k > \sqrt{n}$

so, $O(\sqrt{n})$.

i	p
1	0+1 = 1
2	1+2 = 3
3	1+2+3 = 6
4	1+2+3+4 = 10
5	1+2+3+4+5 = 15
...	...
k	1+2+3+4+5+...+k
	$\Downarrow$
	$\frac{k(k+1)}{2}$

How many times?
time complexity

$\frac{0}{x}$

(4)

⑥ for ($i=1; i \leq n, i=i*2$)
 $\{$ ~~start~~ - ;
 $\}$

how many times?
 if the following condition
 satisfies —

$$\begin{aligned} i &\geq n \\ \Rightarrow 2^k &\geq n & | i = 2^k \\ \Rightarrow k &\geq \log_2 n \end{aligned}$$

$$\begin{aligned} &i \\ &\underline{1} \\ &1 \times 2 \\ &2 \times 2 \\ &2^2 \times 2 \\ &2^3 \times 2 \\ &\vdots \\ &\underline{2^k} \end{aligned}$$

so, time complexity $\rightarrow$
 $O(\log_2 n)$

⑦ for ($i=n; i \geq 1; i=i/2$)

when it will be stopped $\rightarrow$

$$\begin{aligned} i &< 1 \\ \Rightarrow n/2^k &< 1 & | i = n/2^k \\ \Rightarrow n/2^k &\approx 1 \\ \Rightarrow n &= 2^k \Rightarrow k = \log_2 n \\ \Rightarrow O(\log_2 n) \end{aligned}$$

$$\begin{aligned} &i \\ &\underline{n} \\ &n/2 \\ &n/2 \cdot 1/2 \\ &n/2^2 \cdot 1/2 \\ &n/2^3 \cdot 1/2 \\ &\vdots \\ &n/2^k \end{aligned}$$

⑤

⑧ for ()
 {
 }
 time
 ←
 O(n) complex
 } → n times
 normal for loop
 for ()
 {
 }
 time
 ←
 O(n) complex
 } → n times
 normal for loop
 ↓
 2n times
 ↓
O(n)
 combine → O(n).

⑨ Complex: -

for (i=1; i<n; i=i*2) { let, p=0
 { p++ → log_n times

}
 for (j=1; j<p; j=j*2)
 { statem; → log p
 }

that means
 $O(\log p)$
 $= O(\log \log n)$ | $p = \log n$

⑥

⑩ for($i=0; i < n; i++$) $\rightarrow n$
 { for($j=1; j < n; j=j \times 2$) $\rightarrow n \log_2 n$
 { state $\rightarrow n \log_2 n$
 }

So, $n + 2n \times \log_2 n$

$\Rightarrow \underline{\underline{O(n \log_2 n)}}$

(8)

Types of time functions

// $O(1) \rightarrow$ constant

$$f(n) = 2 = O(1)$$

$$f(500) = 2 = O(1)$$

// $O(\log n) \rightarrow$ logarithmic

// $O(n) \rightarrow$ linear

$$\left. \begin{array}{l} f(n) = 2n + 3 \\ f(n) = 700n \\ f(n) = 5n - 200 \end{array} \right\} O(n)$$

// $O(n^2) \rightarrow$
Quadratic

// $O(n^3) \rightarrow$ cube

// $O(2^n) \rightarrow$ exponential

// $O(3^n)$

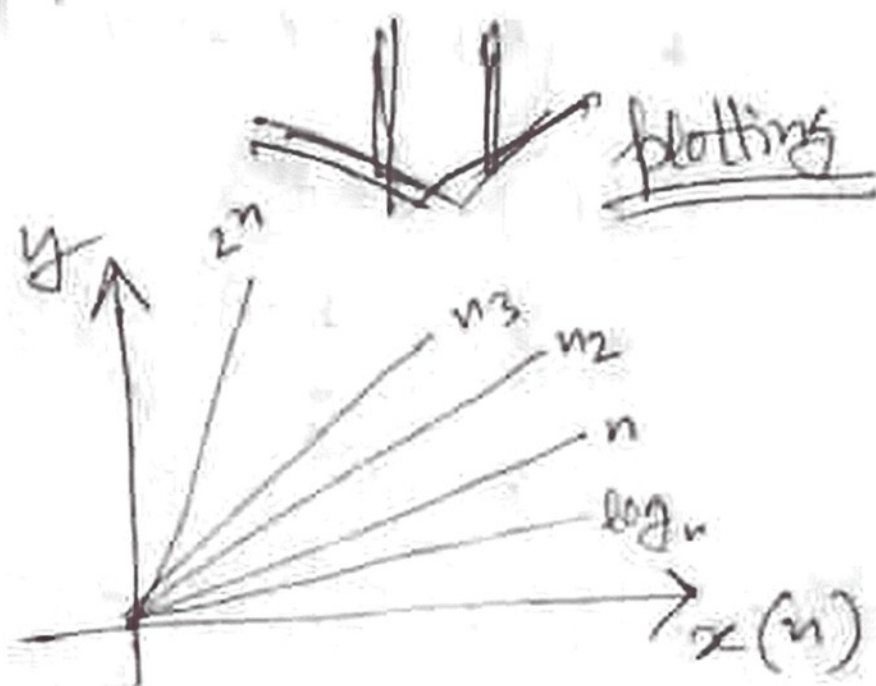
// $O(n^n)$

(9)

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

for example:-

$\log n$	n	n^2	2^n
0	$n=1$	1	2
1	$n=2$	4	4
2	$n=4$	16	16
3	$n=8$	64	256
⋮	⋮	⋮	⋮



(1)

"Asymptotic Notations"

We already know that,

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

$O \rightarrow$ big-oh $\rightarrow$ upper bound
 $\Omega \rightarrow$ big omega $\rightarrow$ lower bound
 $\Theta \rightarrow$ theta $\rightarrow$ average bound

" $O \rightarrow$ big-oh $\rightarrow$ upper bound"

The function $f(n) = O(g(n))$ iff
 $\exists$ positive constant c and c'
such that $f(n) \leq c * g(n) \quad [\forall n > c']$

Example.

$$\begin{array}{ccc} f(n) = 2n + 3 & & \\ 2n + 3 \leq 10n & & [n \geq 1] \\ \downarrow & \downarrow & \downarrow \\ f(n) & c & g(n) \end{array}$$

Therefore, $f(n) = O(g(n)) = O(n)$

(P.T.O)

(11)

Say, $2n+3 \leq 2n+3n$
 $\leq 5n \quad [n \geq 1]$

again, $f(n) = O(n)$.

Extend:-

$$2n+3 \leq 2n^2+3n^2$$
$$\leq 5n^2 \quad [n \geq 1]$$

polynomial?

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

Again,

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

$$\therefore f(n) = O(n^3)$$

all are true,
but target is
to reach the next
function s.t. $O(n)$.

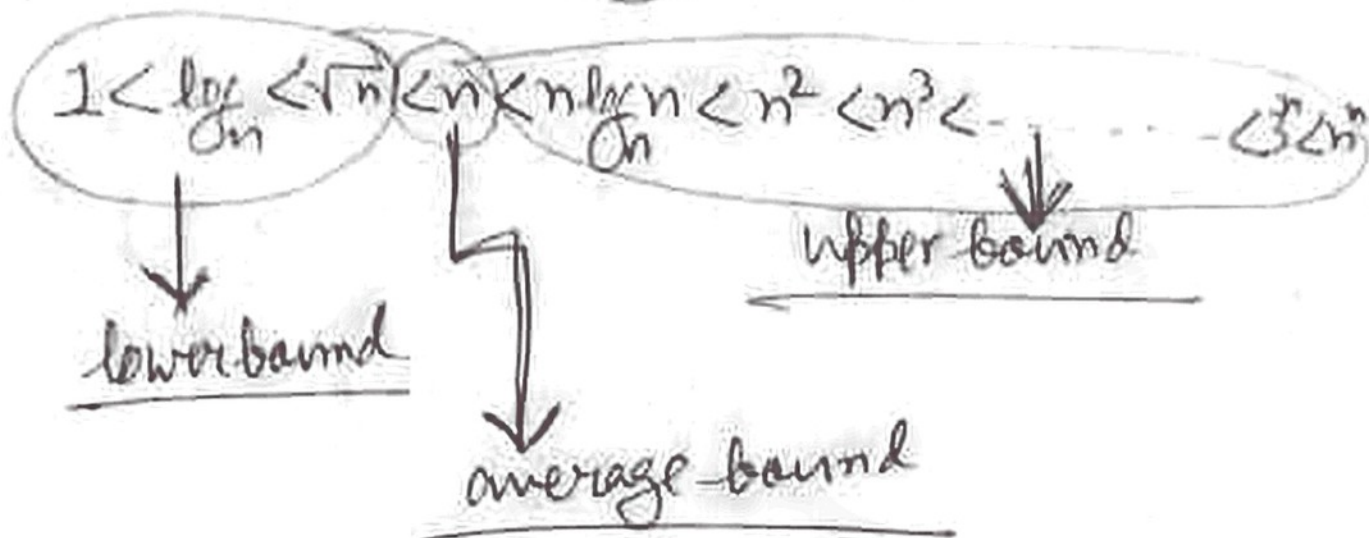
and so on...

but not, $f(n) = O(\log n)$ X

So,

(P.T.O)

(12)



Ω - big-omega $\rightarrow$ lower bound

The function $f(n) = \Omega(g(n))$ iff $\exists$ positive constant c and c' s.t. $f(n) \geq c * g(n) \forall n \geq c'$

Example :- $f(n) = 2n + 3$

$$\frac{2n+3}{f(n)} \geq \frac{1 * n}{g(n)} \quad [n \geq 1]$$

Arrows indicate the mapping from the expression to $f(n)$ and $g(n)$.

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

$$2n+3 \geq \frac{1}{2} * \log n \quad [n \geq 1]$$

Arrows indicate the mapping from the expression to $f(n)$ and $g(n)$.

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

(13)

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

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

and so on ...

but, $f(n) = \Omega(n^2)$ ~~X~~
not upper-bound.

So, for lower bound its wanted
to take closest one. s.t. $\Omega(n)$.

" Θ - Theta notation $\rightarrow$ avg. bound"

The function, $f(n) = \Theta(g(n))$ iff $\exists$ positive
constant c_1, c_2 , and c' . Such that $\rightarrow$

~~for~~
$$c_1 * g(n) \leq f(n) \leq c_2 * g(n)$$

for example, $f(n) = 2n + 3$

$$\begin{array}{ccccc} 1 \times n & \leq & \frac{2n+3}{1} & \leq & 5 \times n \\ \downarrow & & \downarrow & & \downarrow \\ c_1 & & f(n) & & c_2 \cdot g(n) \end{array}$$

hence, $f(n) = \Theta(n)$. (only this)

(P.T.)

(14)

* Never mistake with best case and worst case of an algorithm.

There are notations to representing of bounds of a certain functions.

Examples :-

$$\begin{array}{ccc} 2n^2 + 3n + 4 & \leq & 2n^2 + 3n^2 + 4n^2 \\ \downarrow & & \downarrow \quad \downarrow \\ f(n) & \leq & c \quad g(n) \end{array}$$

$$\text{So, } f(n) = O(g(n)) = \underline{O(n^2)}$$

$$\begin{array}{ccc} 1Xn^2 & \leq & 2n^2 + 3n + 4 \leq 2n^2 + 3n^2 + 4n^2 \\ \downarrow \quad \downarrow & & \downarrow \quad \downarrow \\ c_1 \quad g(n) & & c_2 \quad g(n) \end{array}$$

$$\text{So, } f(n) = \Omega(g(n)) = \Omega(n^2).$$

(15)

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

$$1 \cdot \underline{n^2 \log n} \leq \underline{n^2 \log n + n} \leq \underline{10 n^2 \log n}$$

$$f(n) = O(g(n)) = O(n^2 \log n)$$

$$f(n) = n!$$

$$f(n) = \frac{n(n-1)(n-2)(n-3) \dots 3 \times 2 \times 1}{\downarrow \text{reverse}}$$
$$1 \times 2 \times 3 \times \dots \times n$$

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

$$\frac{1}{\text{lower bound}} \leq n! \leq \frac{n^n}{\text{upper bound}}$$
$$\Omega(1) \quad ? \quad O(n^n)$$

no average bound here

(P.T.O)

(16)

Properties of Asymptotic Notations

General Properties:-

if $f(n)$ is $O(g(n))$
 $\xrightarrow{\text{then}}$ $a * f(n)$ is $O(g(n))$.

Example:- $f(n) = 2n^2 + 5$ is $O(n^2)$
 $\forall f(n) = 14n^2 + 5 \xrightarrow{\text{also}} O(n^2)$

This is true for $\mathbb{R}$ and $\mathbb{Q}$ also.

Reflexive:-

if $f(n)$ is given $\xrightarrow{\text{then}}$ $f(n)$ is $O(f(n))$.

Example:- $f(n) = n^2 \Rightarrow O(n^2)$

Transitive:- if $f(n)$ is $O(g(n))$
and $g(n)$ is $O(h(n)) \xrightarrow{\text{then}}$
 $f(n) = O(h(n))$.

(12)

for example,

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

upper bound $\rightarrow$ n is $O(n^2)$ and n^2 is $O(n^3)$

then n is $O(n^3)$

$f(n)$ $g(n)$ $g(n)$ $h(n)$

Actually, $g(n)$ is upper bound for $f(n)$. $h(n)$ is upper bound for $g(n)$.
Hence, it is obvious that $h(n)$ is will be upper bound for $f(n)$.

* It is also true for Θ and Ω notation.

(18)

Symmetric:

If $f(n)$ is $O(g(n))$ then
 $g(n)$ is $\Omega(f(n))$. [only true for
 Θ -notation]

for example, $f(n) = n^2$ $g(n) = n^2$
 $f(n) = O(n^2)$; $g(n) = \Theta(n^2)$

Trans pose properties:-

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

for example, $f(n) = n$ and $g(n) = n^2$

upper
bound $\rightarrow$ n is $O(n^2)$

lower
bound $\rightarrow$ n^2 is $\Omega(n^2)$

(19)

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

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

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

$\Rightarrow \underline{\underline{f(n) = \Theta(g(n))}}$

Asymptotic Notations:-

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

and $d(n) = O(g(n))$

then: "

for example,

$f(n) = n = O(n)$

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

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

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

(20)

$$\text{if } f(n) = O(g(n))$$

$$\text{and } d(n) = O(e(n))$$

$$\Rightarrow f(n) * d(n) = O(\underline{g(n) * e(n)})$$

$$\downarrow$$
$$n^1$$

$$\downarrow$$
$$n^2$$

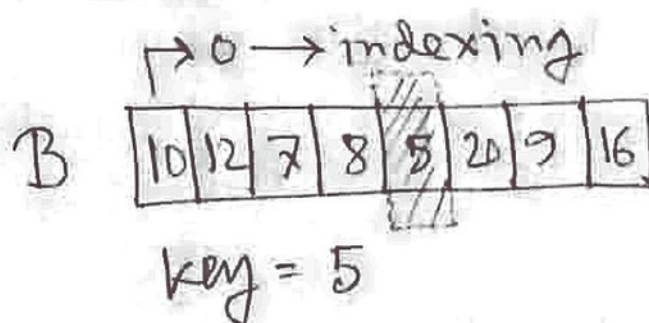
$$\Downarrow$$
$$O(n^3)$$

— 0 —

(21)

Best, Worst, and Average Case Analysis

Linear Search: -



- 1. Check through index.
- 2. If find return the index value
- 3. If not, go forward.

"Best case"

1. If you find the value that you are searching in the first index → it will be best case.

Hence, Best time (constant) $\frac{1}{1} \rightarrow O(1)$
 $\Rightarrow B(n) \rightarrow O(1)$

P.T.O

(22)

WORST CASE

If the desired key/data is in
last index $\longrightarrow$ it will be
worst case.

So, worst case time = n

$$O(n) = n \longrightarrow O(n)$$

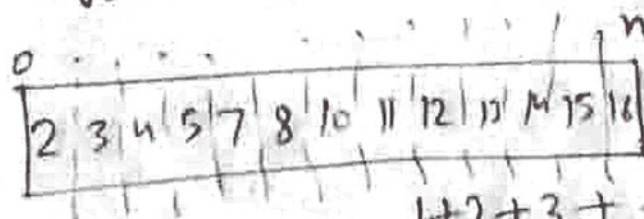
Average Case

It is defined as :-

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

* difficult to find

* Its prefer to find worst case
rather than average case.



$$\text{So, avg. time} = \frac{1+2+3+\dots+n}{n} = \frac{\frac{n(n+1)}{2}}{n}$$

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

$$= \frac{n+1}{2}$$