DATE

Assignment

(1) It is a mathematical notation that describe the behaviour of a function as its input size approaches infinity. It is used to analyze the time and space complexity of algorithm.

Types

(1) Big oh :- It is used to describe the upper bound of the running time or space complexity of algorithm

$$f(n) = O(g(n))$$
$$\text{Iff } f(n) \leq c\, g(n)$$
$$\forall\ n \geq n_0, \text{ some constant } (c > 0$$

(ii) Big omega (Ω):- It is used to describe the lower bound of the running time or space complexity of an algor

$$f(n) = \Omega(g(n)$$
$$\text{Iff } f(n) \geq c(g(n)$$
$$n \geq n_0 \text{ some constant } (c > 0)$$

iii Theta notation (θ): This is used to describe both upper and lower bound of the running time a space complexity

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

PAGE No
DATE

(2) $i = 1, 2, 4, 8, \ldots n$

~~a~~ It is GP

$a_n = ar^{n-1}$

$2n = 2^K.$

$\log_2(2n) = \log_2(2^K)$

$K = \log_2(2) + \log_2(n)$

$K = 1 + \log_2(n)$

$O(\log_2(n))$

(3) $T(n) = \{3T(n-1)$ if $n > 0$ otherwise $1\}$

$T(n) = 3T(n-1)$

$T(0) = 1$

$T(1) = 3T(1-1) = 3T(0) = 3$

$T(2) = 3T(2-1) = 3T(1) = 3.3$

$T(3) = 3T(3-1) = 3T(2) = 3.3.3$

$O(3n) = O(n)$

4 $T(n) = 2T(n-1)-1$ if $(n>0)$ other wis i

~~$T(n) = 3T(n-1)$~~

Here $a = 2$

$b = 1$

$c = \log_2(2) = 1$

$f(n) = 1$

$n^c = f(n)$

$T(n) = \Theta(n\log n)$

5) let us assume input size is n

on first iteratio

i	s
2	3
3	6
4	10
5	15
6	21
'	
'	
.	
k+1	

after k^{th} iteration.

s will be $\frac{k(k+1)}{2}$

loop will stop after.

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

$$k^2 + k > 2n$$

$$k^2 > 2n$$

$$k = \sqrt{2n}$$

Complexity $O(\sqrt{n})$

---

6 let us assume the input size is n.

when	i	after loop i
	1	1
	2	4
	3	9
	4	16
	5	25

DATE

after k^{th} time i becomes k^2.

so $k^2 > n$
$k > \sqrt{n}$

complexity $= O(\sqrt{n})$

7 Time complexity of inner must loop.
$k = 1$ to n $\quad k = k * 2$
$1, 2, 4, 8, 16 \ldots k^{th}$ t.

k^{th} term $= 2^{k-1}$

$n = \frac{2^k}{2} \Rightarrow 2n = 2^k = 1$

taking log both side

$\log_2 2n = \log_2 2^k$
$\log_2 2n = k$.
$k = \log_2 2 + \log_2 n$
$k = 1 + \log_2 n$.

complexity of middle loop

$j = 1$ to n $\quad j = j * 2$
$1, 2, 4, 8, 16 \ldots$
$= 1 + (\log_2 n)$

outer loop.

$i = \frac{n}{2}$ to n $i++$

$k = n - \frac{n}{2}$

Total complexity

$\Rightarrow \frac{n}{2} + (1 + \log_2 n)^{*}(1 + \log_2 \frac{n}{2}) + \frac{n}{2} \log_2 n +$

$\frac{n}{2}(\log_2 n)$

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

8 $\quad i \quad j$

1	$1 \rightarrow n$	
2	$1 \rightarrow n$	$n^{*}n$ times
3	$1 \rightarrow n.$	

for function $(n-3)$

$n, n-3, n-6, n-9 \ldots \ldots k^{th}$

$k^{th} = n(k-1) \cdot 3$

$1 = n - 3k - 3$

$n - 3k - 4 = 0$

$k = \frac{n-4}{3}$

Inner most loop.

$= n * n * n-4$

$= \frac{n^3 - 4n^2}{3}$

complexity $= O(n^3)$.

9 for $i=1$ j will run n times

$i=2$ $\frac{n}{2}$ times

$i=3$ $\frac{n}{3}$ times.

$$\left(n + \frac{n}{2} + \frac{n}{3} \cdots \frac{n}{n}\right)$$

$= n \log_2 n$

complexity $= O(n.\log n)$