Mohammad Aves

13 CST

Tutorial :- 1

Ans (1) Asymptotic Notation is used to describe the running time of an algorithm and how much time of an algorithm takes with a given input. & when input is very large.

Types of Notations :-

① **Big Oh Notation, O** :- The notation O(n) is the formal way to express the upper bound of an algorithm's running time. It measure the worst case time complexity on the longest amount of time an algorithm can passibly take to complete.

Example :- for function f(n)

$O(f(n)) = O\{g(n)\}$ $\quad \forall c > 0, n > n_0$

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

g(n) is tight upper bond of f(n)

② **Big Omega Notation, Ω** :- The notation Ω(n) is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

$\forall c > 0, n \geq n_0$

ex :- $\theta(f(n)) = \Omega(g(n))$

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

(3) **Theta Notation, θ** :- This notation is formal way to express both lower bound and the upper bound of an algorithm's running time.

$$f(n) = \theta\, g(n)$$

$$\text{if } c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$$

$$\forall \; n \ge \max(n_1, n_2)$$

$$c_1 > 0 \;\&\; c_2 > 0$$

Ans :- (2)

```
for (i=1 to n) // O(log(n))
{
  i=i*2;   // O(1)
}
```

for $i = 1, 2, 4, 8, 16 \ldots 2^k$ this means (k) times

as per this code it will run till $2^k = n$

which means $k = \log n$

then

complexity is $O(\log n)$

$$\sum_{i=1}^{n} 1+2+4+8+\ldots+n.$$

$$T_k = ar^{k-1} \Rightarrow 1 \times 2^{k-1}, \quad n = 2^{k-1}$$

$$2n = 2^k$$

$$= \log(2n) = k \log 2$$

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

$$O(k) = O(1+\log n)$$

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

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

$$T(n) = 3T(n-1) \quad \text{— (i)}$$

Put $n = n-1$

$$T(n-1) = 3T(n-2) \quad \text{— (2)}$$

from 1 to 2

$$T(n) = 3(3T(n-2))$$
$$= 9T(n-2) \quad \text{— (3)}$$

Put $n = n-2$ in — (i)

$$T(n) = 3(T(n-3)) \quad \text{— (4)}$$
$$T(n) = 27(T(n-3))$$
$$T(n) = 3^k(T(n-k)) \qquad \{k = 3\}$$

Put, $n-k=0$
$n = k$

$$T(n) = 3^n[T(n-n)]$$
$$T(n) = 3^n[T(0)] \qquad \{T(0) = 1\}$$
$$T(n) = 3^n \times 1$$
$$T(n) = O(3^n)$$

Ans (4) :- $T(n) = \begin{cases} 2T(n-1) & , n>0 \\ 1 & , n \leq 0 \end{cases}$

using backward substitution,

$$\left.\begin{aligned} T(n) &= 2T(n-1) \\ T(n-1) &= 2T(n-2) \\ T(n-2) &= 2T(n-3) \\ &\vdots \\ T(1) &= 2T(0) \end{aligned}\right\} \text{ n levels}$$

$T(0) = 1$

Substituting value of $T(n-1)$ then $T(n-2)$ -- till $T(1)$ in eq^n $T(n)$

we get,

$T(n) = 2^n \times T(0)$

$\therefore T(n) = 2^n \times 1$

$= O(2^n)$

Ans (5) What should be time complexity of

```
int i=1 , s=1;
while (s<=n)
{
    i++; s=s+i;
    Printf ("#");
}
```

$i = 1, 2, 3, 4, 5, 6 - - -$

$S = 1 + 3 + 6 + 10 + 15 + 21 + - - - + n$

$0 = 1 + 2 + 3 + 4 + - - - - n - T(n)$

$1+2+3+4 - - - + k > n$

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

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

$k^2 > n$

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

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

Ans:-(6)

```
void function (int n)
{
  int i, count=0;
  for( i=1; i*i<=n; i++)
  count ++ ;
}
```

$i = 1, 2, 3, - - n$

$i^2 = 1, 4, 8, - - - n$

$i^2 <= n$ & $i <= \sqrt{n}$

kth term, $t_k = a + (k-1)d$

$a = 1, d = 1$

$t_k <= \sqrt{n}$

$\sqrt{n} = 1 + (k-1)1$

$\sqrt{n} = k$

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

Ans(7)

i	j	k
$\frac{n}{2}$	$\log(n)$	$\log(n)$
	$\vdots$	$\vdots$
$\frac{n}{1}$		
n	$\log(n)$	$\log n$

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

$$O(i * j * k) = O\left(\left(\frac{n+1}{2}\right) * \log n * \log n\right)$$

$$= O\left(\left(\frac{n+1}{2}\right) * (\log_2 n)^2\right)$$

$$T(n) = O(n(\log_2 n)^2)$$

Ans-6) $T(n) = T(n-3) + n^2$ — (i)

$T(1) = 1$ — (2)

Put $n = n-3$ in (i)

$T(n-3) = T(n-6) + (n-3)^2$ — (3)

Put (3) in (1)

$T(n) = T(n-6) + (n-3)^2 + n^2$ — (4)

Put $n = n-6$ in (1)

$T(n-6) = T(n-9) + (n-6)^2$ — (5)

Put (5) in (4)

$$T(n) = T(n-9) + (n-6)^2 + (n-3)^2 + n$$

$$T(n) = T(n-3k) + (n-3(k-1))^2 + (n-3(k-2))^2 + \ldots n^2$$

$$n - 3k = 1$$

$$\frac{n-1}{3} = k$$

$$T(n) = T(1) + \left(n-3\left(\frac{n-1}{3}-1\right)\right)^2 + \left(n-3\left(\frac{n-1}{3}-2\right)\right)^2$$

$$T(n) = 1 + [3+1]^2 + [6+1]^2 + \ldots n^2$$

$$T(n) = 1 + 4^2 + 6^2 + \ldots n^2$$

$$= n^2 + \ldots 1$$

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

Sol 10

$$n^k = O(c^n)$$

as $n^k \leq a \cdot c^n$

$\forall\ n \geq n_0$ for some constant $a > 0$

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq O(2)$

$n_0 = 1$ & $c = 2$

Sol: (9)

i	j
1	n times
2	$1+3+5+\dots+n$ times

$$a_n = a + (K-1)d$$

$$a=1, d=2$$

$$n = 1 + (K-1)2$$

$$\frac{n-1}{2} = K-1$$

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

$$K = \frac{n+1}{2} \quad \text{// No. of terms}$$

For $i=2$, $j = \frac{n+1}{2}$ times

for $i=3$, $j = 1+4+7+\dots+n$ times

$$n = 1 + (K-1)d$$

$$n = 1 + (K-1)3$$

$$\frac{n-1}{3} + 1 = K$$

$$K = \frac{n+2}{3} \quad \text{// No. of terms}$$

generalising,

for $i=n$ $\quad j = \frac{n+K-1}{K}$ times

$$T(n) = n + \frac{n+1}{2} + \frac{n+2}{3} + \dots \frac{n+K-1}{K} \quad \text{— n term}$$

$$\therefore \sum_{i=1}^{n} \frac{n+K-1}{K} \Rightarrow \frac{\sum_{i=1}^{n} n + \sum_{i=1}^{n} K - \sum 1}{K}$$

$$\frac{\frac{n(n+1)}{2} + nk - n}{k}$$

$$= \frac{\frac{n^2+n}{2} + nk - n}{k}$$

$$= \frac{n^2 + n + 2nk - 2n}{2k}$$

After remaing constant term and lower

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