Name → Karan Singh Bisht
Sec → D
Roll No → [redacted] 14

TUTORIALS – 1.

1).

ASymptotic Notations :-

ASymptotic Notations are the mathematical notations used to describe the running time of an algorithm, when the input tends towards a particular Value or a limiting Value.

big O, big Θ, big Ω are different types of asymptotic notations.

2).
2^0 $i = 1$
2^1 $i = 2$
2^2 $i = 4$
2^3 $i = 8$
2^4 $i = 16$ .... 2^k (k times). form Values.

So $2^k = n$

$\log 2^k = \log n$

$k \log_2 2 = \log_2 n$

$k = \log_2 n$

Hence, the time complexity is $O(\log n)$.

3) $T(n) = 3T(n-1)$ —— (1) $\quad T(0) = 1$

Let $n = n-1$

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

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

$T(n) = 3[3T(n-2)]$ —— (2)

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

$T(n) = 3[3.3T(n-3)]$ —— (3)

So, from above 3 Eq^n's we should obtain a Relat^n.

$T(n) = 3^K T(n-K)$

Let $n-K = 0$

$n = K$

$T(n) = 3^K T(0)$

Here $T(0) = 1$

So, $T(n) = 3^K . 1$

$= 3^K$

So time complexity is $3^n = O(3^n)$

4). $T(n) = 2T(n-1) - 1$, $T(0) = 1$

Let $n = n-1$

$T(n-1) = 2T(n-1-1) - 1$

$T(n-1) = 2T(n-2) - 1$

$T(n) = 2[2T(n-2) - 1] - 1$

$T(n)\ 4T(n-2) - 3$ ——→ (2)

$n = n-2$

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

$T(n) = 4[2T(n-3) - 1] - 3$

$T(n) = 8T(n-3) - 7$ ——→ (3)

$T(n) = 2^k [(n-k) - \{ 1 + \cancel{k} \; 2^{k-1} + 2^{k-2} + \cdots 2^2 + 2 + 1$

Let $n-k = 0$

$n = k$

$= 2^n T(0) - \{ 1 + 2 + 2^2 + \cdots + 2^{k-1} \}$

$= 2^n \times 1 + 2^k + 1$

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

$= 2^n + 1$

$O(2^n)$ is the given time complexity for given relation.

5). Here $S_i = S_{i-1} + i$

the value of 'i' increases by 1 for each iteration
the value contained in 's' at the i^{th} iteration is the sum of the first i positive ~~it~~ integers.
If 'k' is the total no. of iterations taken by program then loop like

$1 + 2 + 3 + \cdots k$

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

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

Hence the time complexity is $O(\sqrt{n})$.

6). Let | Passes | Let $k = 16$.

$i = 1$	for $k = 1$	1
$i = 2$	for $k = 2$	4
$i = 3$	for $k = 3$	9
$i = 4$	for $k = 4$	16
⋮	⋮	⋮
		n^2
$i = n$	for $k = n$	

$1 \quad 4 \quad 9 \quad 16 \;——\; n^2$

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

$$= O(\log^n 1) + O(\log^n 2) + \cdots O(\log^n n) < c.O(\log^2 n)$$

Therefore time complexity is $O(\log^n n)$

7). Let $n = 12$

i	j	k
6	1	1
7	2	2
8	4	4
9	8	8
10	16	16

(Out of boud)

So, $\left(\frac{n}{2}\right) \times (\log n) \times (\log n)$

as Constants can be ignored.

Here for each value of it iterate & check the condition for k.

So,

time Complexity is O
$$(n \cdot \log n \cdot \log n)$$
$$= O(n \log^2 n)$$

8)

i	j	no. of times
1	1	-1
2	2	-2
⋮	⋮	⋮
n	n	n times

$(n)(n)$ times

Here $n = \underline{n-3}$.

$(n-3)(n-3)$

$O(n^2 + 9 - 6n)$

$= O(n^2)$ is the time Complexity.

9). for (i = 1 to n) Let n = 12
{
for (j = 1 ; j <= n ; j = j + i)
print ("*");

$i = 1$, $j = 2, 3, 4, 5$ ——— $(n-1)$
$i = 2$, $j = 3, 4, 5, 6,$ ——— $(n-1)$
$i = 3$, $j = 4, 5, 6$ ——— $(n-1)$
$i = 4$, $j = (n+1)$ ——— $(n-1)$

for each Value of i, n iterates through $(n-1)$ times
for $n(n-1)$ time.

$= (n^2 - n)$

$= O(n^2)$.

Hence the time Complexity is $O(n \log n)$.