Name - Priyanshu Bhatt
Section - D
Roll no - 30

B+

Tutorial - 1

1) 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 the ~~particular~~ different types of asymptotic notation.

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) for n 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) $T(n) = 3T(n-1)$ ——(i) $T(0) = 1$

let $n = n-1$

$T(n-1) = 3T(n-1-1)$
$T(n-1) = 3T(n-2)$
$T(n) = 3[3T(n-2)]$ ——(ii)
$T(n-2) = 3T(n-2-1)$
$T(n) = 3[3.3T(n-3)]$ ——(iii)
So, from above three equations we should obtain function

$$T(n) = 3^k\, T(n-k)$$

let $n-k=0$

$n=k$

$T(n) = 3^k\, T(0)$ Here $T(0) = 1$

$= 3^k \cdot 1$

$= 3^n$

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

~~$T(n) = 2T(n-1) - 1$, $n>0$ $T(0) = 1$~~

4) $T(n) = 2T(n-1) - 1$ — (i) $T(0) = 1$

Let $n = n-1$

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

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

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

$T(n) = 4T(n-2) - 3$ — (ii)

$n = n-2$

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

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

$T(n) = 8T(n-3) - 7$ — (iii)

$T(n) = 2^k\, T(n-k) - \{2^{k-1} + 2^{k-2} + \dots + 2^2 + 2 + 1\}$

let $n-k=0$

$n=k$

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

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

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

$2^n + 1$

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

5) Here $S_i = S_{i-1} + i$
the value of i increase by 1 for each iteration
the value contained in 's' at the i^{th} iteration is
the sum of the first i positive 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 n = 16

Let	Passes	
$j = 1$	for $K = 1$	1
$j = 2$	for $K = 2$	4
$j = 3$	for $K = 3$	9
$i = 4$	for $K = 4$	16
⋮	⋮	⋮
		n^2
$j = n$	for $K = n$	

$$1 \quad 4 \quad 9 \quad 16 \quad \cdots \quad n^2$$

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

$$= O(\log^2 1) + O(\log^2 2) + \cdots O(\log^2 n) \leq C \cdot O(\log^2 n)$$

Therefore time complexity is
$O(\log^2 n)$. //

7)

i	j	k	
6	1	1	
7	2	2	
8	4	4	
9	8	8	
10	16	16	(out of bound)

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

as constants can be ignored
Here for each value of i it iterates and
check the condition for k
So,
time complexity is like
$(n \cdot \log n \cdot \log n)$

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

8.

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

n times

i = n times
j = n times
i j = n.n

(n) (n) time
Here n = n-3
(n-3) (n-3)
$O(n^2 + 9 - 6n)$
$\Rightarrow O(n^2)$ is the time complexity

9) let $n \in R$

```
for (i = 1 to n){
  for (j = 1; j <= n; j = j+1)
  printf ("*");
}
```

i

$i = 1, j = 2, 3, 4, 5 \ldots (n-1)$

$i = 2, j = 3, 4, 5, 6 \ldots (n-1)$

$i = 3, j = 4, 5, 6 \ldots (n-1)$

$i = n, j = (n+1) \ldots (n-1)$

for each of i value n it iterates through $(n-1)$ times

for $n(n-1)$ time

$= (n^2 - n)$

$O(n^2)$

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