Name- Upendra Pandit
Section - D
Roll no - 48

Tutorial 1

Q1 Asymptotic ~~Notation~~ 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 the different types of asymptotic notations.

Q2

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 $2^k = n$
$\log 2^k = \log n$
$k \log 2 = \log n$
$k = \frac{\log n}{\log 2}$

$k = \log_2 n$

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

PAGE No.
DATE

Soln $T(n) = 3T(n-1)$ — ① $T(0)=1$

let $n = n-1$

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

$T(n-1) = 3T(n-2)$ — ②

put ~~it~~ ② in ①

$T(n) = 3 \cdot 3T(n-2)$

put $n = n-2$

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

$T(n-2) = 3T(n-3)$ — ③

put ③ in ①

$T(n) = 3[3 \cdot 3T(n-3)]$ — ③

~~So from~~ above 3 equations we should obtain a rel^n.

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

let $n-k=0$

$\boxed{n=k}$

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

Here $T(0)=1$

So, $T(n) = 3^k \cdot 1$

$= 3^n$

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

Sol 4. $T(n) = 2T(n-1) - 1$ — ①

$T(0) = 1$

let $n = n-1$

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

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

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

$T(n) = 4T(n-2) - 3$ — (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)~~

$\vdots$

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

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 give relation.

Soln. Here $S_i = S_i - 1 + i$

the value of i increased 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 the loop like

$1+2+3+ \ldots k$

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

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

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

~~let~~

Sol n6

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

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

$$= \frac{n(n+1)(2n+1)}{6}$$

$$= O(\log^2 1) + O(\log_2^2) + \ldots O(\log_n^2)$$

$$\leq C \cdot O(\log^2 n)$$

therefore time complexity $= O(\log^2 n)$

PAGE No.
DATE

Soln 7 let $n = 12$

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

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

as consts can be ignored.
then ~~for~~ for each value of i it iterates & check the condition for k.

So,
time complexity is
$= \cancel{n f} \cdot (n \log n . \log n)$

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

8

i	j	no. of times pass root
1	1	-1
2	2	-2
1	1	1
1	1	1
n	n	n time

$i = n$ times
$j = n$ times
$i \cdot j = n \cdot n$.

(n).(n) times
Here n = n-3
(n-3) (n-3)
$O(n^2+9-6n)$
$= O(n^2)$ is time complexity

(9) Let n = 12

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

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

for each value of i n, it iterates through (n-1) times for n (n-1) time.

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

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