Name- Shanjal Gupta
Section- D
Rollno- 35

Date.
Page No.

Tutorial-1

Ques 1.
Soln. 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 the different types of asymptotic notation.

Soln 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 $O(\log n)$

Soln 3. $T(n) = 3T(n-1)$ -(1) $T(0) = 1$
let $n = n-1$
$T(n-1) = 3T(n-1-1)$
$T(n-1) = 3T(n-2)$
$T(n) = 3[3T(n-2)]$ -(2)
$T(n-2) = 3T(n-2-1)$
$T(n) = 3*[3 \cdot 3T(n-3)]$ -(3)
So, from above 3 eq^ns we should obtain a Rel^n
$T(n) = 3^k T(n-k)$
Let $n-k = 0$
$n = k$

Date
Page No.

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

Here $T(0) = 1$

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

$= 3^n$

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

$T(n) = 2T(n-1) - 1$ & $T(0) = 1$

Sol^n

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

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

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

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

$T(n) = 4T(n-2) - 3 \quad -(2)$

$n = n-2$

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

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

$T(n) = 8T(n-3) - 7 \quad -(3)$

$\vdots$

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

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 given relation.

Date
Page No.

Ans 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

$$1 + 2 + 3 + \dots + k = \frac{k(k+1)}{2} > n$$

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

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

Sol^n 6 =

let, n=8 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	11
⋮		⋮
i=n	For k=n	n^2

$$1\ 4\ 9\ 16 \dots n^2$$

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

$$= O(\log^2 1) + O(\log^2 2) + \dots O(\log^2 n) \leq c \cdot O(\log^2 n)$$

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

Let n = 12

Ans:

i	j	k
6	12	12
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 & check the condition for k.

So,

time complexity is like

$(n \cdot \log n \cdot \log n)$

$= O(n \log_n^2)$

8.

i	j	no. of times part look.
1	1	1
2	2	2
1	1	1
1	1	1
n	n	n times.

i = n times

j = n times

i.j = n.n

(n) (n) time.

Here $n = n-3$

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

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

$= O(n^2)$ is time complexity

Date
Page No.

Let $n = 12$

9.
```
for (i = 1 to n) {
  for (j = 1; j <= n; j = j + i)
    printf("*");
}
```

i	j
1	1, 3, 5
2	5
3	
4	

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

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

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

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

For each of i value n it iterates through $(n-1)$ time for n $(n-1)$ time.

$= (n^2 - n)$

$= O(n^2)$

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