

Ques.

Solu-

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.

$$2^0 \quad i = 1$$

$$2^1 \quad i = 2,$$

$$2^2 \quad i = 4$$

$$2^3 \quad i = 8$$

$$2^4 \quad i = 16 \quad \dots \quad 2^k \text{ (K times). for } n \text{ values.}$$

$$\text{so } 2^k = n$$

$$\log 2^k = \log n$$

$$k \log_2 2 = \log_2 n$$

$$k = \log_2 n$$

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

$$\text{Solu-} \quad T(n) = 3T(n-1) \quad -(1) \quad T(0) = 1$$

$$\text{let } n = n-1$$

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

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

$$T(n) = 3[3T(n-2)] \quad -(2)$$

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

$$T(n) = 3[3 \cdot 3T(n-3)] \quad -(3)$$

So, from above 3 eqns we should obtain a reln.

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

$$\text{Let } n-k = 0$$

$$n = k$$

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

Here  $T(0) = 1$

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

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

$$T(n) = 2T(n-1) - 1 \quad T(0) = 1$$

Sol<sup>n</sup>  
 U.

$$\text{let } n = \frac{n}{2} - 1$$

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

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

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

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

$$n = n-2$$

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

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

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

⋮

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

$$\text{let } n-k=0.$$

$$n=k.$$

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

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

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

$$= 2^n + 1$$

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

Ans 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  $i^{\text{th}}$  iteration is the sum  
 of the first  $i$  positive integers.

If  $K$  is the total no. of iterations taken by program  
 then loop like

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

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

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

at  $n=8$  passes

at  $n=16$ .

Sol <sup>n</sup> 6	$i=1$	for $K=1$	1
=	$i=2$	for $K=2$	4
	$i=3$	for $K=3$	9
	$i=4$	for $K=4$	16
	$\vdots$		$\vdots$
	$i=n$	for $K=n$	$n^2$

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

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

$$= O(\log_1) + O(\log_2) + \dots + O(\log_n) \leq C \cdot O(\log_n)$$

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

Let  $n = 12$

<u>Answr</u>	$i$	$j$	$k$
6		12	13
7		2	2
8		4	4
9		8	8
10		16	16. (out of bound)

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

as constants can be ignored.  
Here for each value of  $i$  it iterates & checks the condition for  $k$ .

So,  
time complexity is like

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

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

<u>8-</u>	$i$	$j$	$k$	no. of times
	1	1	-1	1 time
	2	2	-2	1 time
	1	1	1	1 time
	1	1	1	1 time
	$n$	$n$	$n$	<u><math>n</math> times.</u>

$$P = n \text{ times}$$

$$J = n \text{ times}$$

$$(i \cdot j) = n \cdot n$$

$(n) (n)$  time.

Here  $n = n-3$

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

$$O(n^2 + 89 - 6n)$$

$$= O(n^2)$$

is time complexity

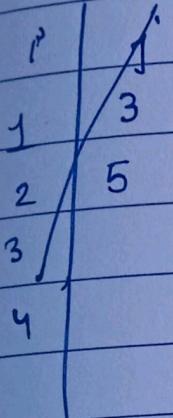
Let  $n = 12$

for ( $i = 1$  to  $n$ ) {

    for ( $j = 1$ ;  $j \leq n$ ;  $j = j + i$ )

        printf(" \* ");

}



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

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

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

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

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

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

$$= O(n^2)$$

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