TUTORIAL-1

Ans-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.

1) Big-O notation → It describes the worst case complexity of an algorithm. It gives the upper bound of a program/algo. eg. $O(\log n)$ describes the Big O of binary search algo.

2) Big-Θ notation → It specifies both upper and lower bounds for a function and provides the average time complexity of an algo.

3) Big-Ω notation → It specifies the lower bound for a function i.e it describes the best case running time of a program. eg. bubble sort Big-Ω complexity is $\Omega(N)$ when the array is sorted.

4) <u>Small-o notation</u> → It is used to describe an upper bound that cannot be tight i.e it provides loose upper bound of $f(n)$.

5) <u>Small omega notation</u> → It denotes the lower bound (that is not asymptotically tight) on the growth rate of runtime of an algo.

<u>Ans-2</u>

for ($i = 1$ to N) { $i = i * 2$; }

$i = 1, 2, 4 \ldots$

$i = 2^0, 2^1, 2^2 \ldots n$

This is a GP

$a = 1, r = 2/1 = 2$

$t_K = ar^{K-1}$

$2n = 2^K$

$\log_2 2 + \log_2 n = K \log_2 2$

$K = 1 + \log_2 n$

$K \propto \log_2 n \Rightarrow O(\log_2 n)$ Ans

$\sum_{i=1}^{N} (1 + 1 + \ldots \log_2 n) = O(\log_2 n)$

Ans-3

$T(n) = 3T(n-1)$ — ①

put $n = n-1$ in ①

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

put $n = n-2$ in ①

$T(n-2) = 3T(n-3)$ — ③

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

$T(n) = 9T(n-2)$ — ②

put ③ in ②

$T(n) = 3.9T(n-3)$

$T(n) = 27T(n-3)$

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

assume $n-k = 0 \Rightarrow n = k$

$T(n) = 3^n \cdot T(0)$

$T(0) = 1$

$\therefore O(3^n)$ Ans

Ans-4

$T(n) = 2T(n-1) - 1$ — ①

put $n = n-1$ in ①

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

$T(n) = 4T(n-2) - 2 - 1$ — ②

put $n = n-2$ in ①

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

substitute in ②

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

$T(n) = 8T(n-3) - 4 - 2 - 1$

$T(n) = 2^k T(n-k) - \underbrace{(2^0 + 2^1 + 2^2 \ldots 2^{k-1})}_{\leftarrow k \text{ terms} \rightarrow}$

$n - k = 0$

$n = k$

$= 2^n T(0) - \dfrac{1.(2^k - 1)}{2-1}$

$= 2^n - 2^k + 1$

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

$T(n) = 1$

$\therefore O(1)$ Ans

Ans-5

```
int i=1, s=1;
while (s<=n) {
    i++;              → O(1)
    s+=i;             → O(1)
    print("#");       → O(1)
}
```

$$S = 1, \underbrace{3}_{2}, \underbrace{6}_{3}, \underbrace{10}_{4}, \underbrace{15}_{5} \ldots$$

difference in AP

$$T_n = AK^2 + BK + C$$

putting $K=1$

$$A + B + C = 1 \quad \text{--- (1)}$$

putting $K=2$

$$4A + 2B + C = 3 \quad \text{--- (2)}$$

putting $K=3$

$$9A + 3B + C = 6 \quad \text{--- (3)}$$

solving (1), (2) & (3)

$$A = 1/2, \quad B = 1/2, \quad C = 0$$

$$T(n) = \frac{K^2}{2} + \frac{K}{2} = \frac{K(K+1)}{2}, \quad n < \frac{K(K+1)}{2}$$

Time Complexity $= O(\sqrt{n})$ Ans

Ans-6

```
for(i=1; i*i<=n; i++)
```

values of i: $\underbrace{1, 4, 9, 16 \ldots (\sqrt{n})^2}_{K}$

first difference forms AP

$AK^2 + BK + C = t_K$

put $K=1$: $A+B+C=1$ — (1)

put $K=2$: $4A+2B+C=4$ — (2)

put $K=3$: $9A+3B+C=9$ — (3)

Solving (1), (2) & (3)

$A=1, B=0, C=0$

$n = AK^2 + BK + C$

$n = AK^2 + 0K + 0$

$n = 1K^2$

$n = K^2$

$K = \sqrt{n}$

Time Complexity $= O(\sqrt{n})$ Ans

Ans-7

i	j	k
$n/2$	$\log n$	$\log n \times \log n$
$n/2+1$	$\log n$	$\log n \times \log n$
$\vdots$		
n	$\log n$	$\log n \times \log n$

$\Rightarrow O(n \times (\log_2 n)^2)$ Ans

Ans-8

$(n-3), (n-6), (n-9), \ldots\ldots (1) \rightarrow k$ terms

$a = n-3, \quad d = n-6-n+3 = -3$

$1 = (n-3) + (k-1)(-3)$

$1 = (n-3) - 3k + 3$

$1 = n - \cancel{3} - 3k + \cancel{3}$

$3k = n-1$

$k = \dfrac{n-1}{3}$

$O(n \times n^2)$

$\Rightarrow O(n^3)$ Ans

Ans-9

for $i=1, j=n$ times

$i=2, j=n/2$ times

$i=3, j=n/3$ times

⋮

$i=k, j=n/k$ times

$i=n, j=n/n$ times

Total time Complexity $= n + n/2 + n/3 + \dots n/n$

$$n * \underbrace{(1 + 1/2 + 1/3 + \dots + 1/n)}_{\log n}$$

$$1 + 1/2 + 1/3 + \dots 1/n = \sum_{k=1}^{n} 1/k$$

$$= \log n + O(1)$$

$$O(\log n \times n) \Rightarrow O(n \log n)$$ Ans

Ans-10

$f(n) = n^k, \quad g(n) = c^n$

where $k >= 1$ & $c > 1$

let $k = 1, \quad c = 2$

$f(1) = (1)^1 = f(1)$

$g(1) = 2^1$

$f(1) < g(1)$

$f(2) = (2)^1 \qquad g(2) = (2^2) = 4$

$f(2) < g(2)$

Satisfies O notation

$$f(n) \leq c \cdot g(n)$$

$$f(n_0) = c_0 \cdot g(n_0)$$

$$n_0^k = c_0 \cdot c^{n_0}$$

$K = 1, c = 2$

$$n_0^1 = c_0 \cdot 2^{n_0}$$

$$\left(\frac{n_0}{c_0}\right)^1 = (2)^{n_0}$$

Comparing, $n_0 = 1$

$$\frac{n_0}{c_0} = 2$$

$$\frac{1}{2} = c_0$$

$$f(n) \leq 0.5\, g(n)$$

$$f(n) = O\, g(n)$$