JARIWALA MANAV
Section : D
R. No : 20

2016844

Date : / /
Page No. :

1] ASYMPTOTIC NOTATIONS

→ These are the mathematical notations used to describe the running time of an algorithm when input tends towards a particular value or a limiting value.

→ Those notations which are used to tell complexity of an algorithm when input is very large.

i) Big O Notation

→ It describes the worst-case running time of a program.

→ We compute big-O of an algorithm by counting how many iterations are there in an algorithm.

→ Here, g(n) is "tight" upper bound.

→ For example, O(log n) describes Big-O of a binary search algorithm.

ii) Big Ω Notation

→ It describes the best case running time of a program

→ We compute the Big Ω by counting how many iterations an algorithm will take in best case scenario based on input N.

→ E.g. Bubble ~~Time~~ Sort algorithm has a running time of O(N) because in the best case scenario the list is already sorted & bubble sort will terminate after first iteration.

→ g(n) is "tight" lower bound.

A Quality Product by Ginger Your Lucky Brand

Teacher's Signature

iii) Big Θ Notation:

→ We compute big Θ of an algorithm by counting number of iterations the algorithm takes as input of n.

→ Here, g(n) is both "tight" UB & LB

iv) Small o Notation:

→ It is used to describe an upper bound that cannot be tight.

→ In other words, g(n) is loose upper bound of f(n).

v) Small ω Notation:

→ Commonly written as ω is an asymptotic notation to denote the lower bound (that is not asymptotically tight) on growth rate of runtime of an algorithm.

→ Here, g(n) is lower bound of function f(n)

Date : / /
Page No. :

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

$i = 1, 2, 4 \ldots$
$i = \underbrace{2^0, 2^1, 2^2, \ldots}_{k \text{ terms}} 2^n$

Here, $a=1$, $r=2$

$t_k = ar^{k-1}$
$= 1\ 2^{k-1}$
$n = \frac{2^k}{2}$

~~2n~~ $2n = 2^k$

Taking log on both sides:
$\log 2 + \log_2 n = k \log_2 2$
$k = 1 + \log_2 n$

Time Complexity : $\boxed{O(\log_2 n)}$

A Quality Product by Ginger Your Lucky Brand

Teacher's Signature

Date : / /
Page No. :

3] $T(n) = 3\,T(n-1)$ — ①

$$T(n) = \begin{cases} 1 & , n = 0 \\ 3\,T(n-1) & , n > 0 \end{cases}$$

→ Put $n = (n-1)$ in eq^n ①:
$T(n-1) = 3\,T(n-2)$ — ②

Subs ② in ①:
$T(n) = 3 \cdot 3\,T(n-2)$ — ③

→ Put $n = (n-2)$ in eq^n ①:
$T(n-2) = 3\,T(n-3)$ — ④

Subs ④ in ③:
$T(n) = 3 \cdot 3 \cdot 3\,T(n-3)$

→ Generalizing it: $T(n) = 3^k\,T(n-k)$ — ⑤
Let $\cancel{n-k=0}$ $n-k=1$
$\cancel{n=k}$ $\cancel{n=k-1}$ $k = n-1$

Subs $k=n-1$ in eq^n ⑤:
$T(n) = 3^n\,T(1)$
$= 3^n$ $(\because T(1) = 1)$

→ $\therefore$ Time Complexity : $\boxed{O(3^n)}$

A Quality Product by Ginger Your Lucky Brand

Teacher's Signature

Date : / /
Page No. :

4) $T_n = \begin{cases} 1 & n = 0 \\ 2T(n-1) - 1 & n > 0 \end{cases}$

$\therefore T(n) = 2T(n-1) - 1$ — eqⁿ ①

→ Put $n = n-1$ in eqⁿ ①:

$T(n-1) = 2T(n-2) - 1$ — ②

Subs ② in ①:

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

$= 2.2(T n-2) - 3$ — ③

→ Put $n = n-2$ in eqⁿ ①:

$T(n-2) = 2T(n-3) - 1$ — ④

Subs ④ in ③:

$T(n) = 2.2[2T(n-3) - 1] - 3$

$= 2.2.2\,T(n-3) - 7$

$\therefore T(n) = 2.2.2\,T(n-3) - 4 - 2 - 1$ — ⑤

→ Generalizing it:

$T(n) = 2^k\,T(n-k) - (2^0 + 2^1 + \ldots + 2^{k-1})$ — ⑥

Let $n - k = 0$

$n = k$

→ $\therefore T(n) = 2^n\,T(n-n) - 1 \times \left(\frac{2^k - 1}{2-1}\right)$

$= 2^n(1) - 2^n + 1$

$= \boxed{O(1)}$

A Quality Product by Ginger Your Lucky Brand

Teacher's Signature

Date : / /
Page No. :

5]
```
int i=1 ,s=1;
while (s<=n){
    i++ ;s=s+i;
    printf("#");
}
```

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

→ First differenre is in AP:
$T_n = AK^2 + BK + C$ —①

→ Put $k=1$ in ①:
$A+B+C = 1$ —②

→ Put $k=2$ in ①:
$4A + 2B + C = 3$ —③

→ Put $k=3$ in ①:
$9A + 3B + C = 6$ —④

Solving ②, ③ & ④:
$A = \frac{1}{2}$, $B = \frac{1}{2}$, $C = 0$

→ ∴ $T(n) = \frac{K^2}{2} + \frac{K}{2} = \frac{K(K+1)}{2}$

$n < \frac{K(K+1)}{2}$

∴ Time complexity $\boxed{O(\sqrt{n})}$

A Quality Product by Ginger Your Lucky Brand

Teacher's Signature

Date : / /
Page No. :

6]
```
void Function(int n)
{
    int i, count = 0;
    for(i=1, i*i<=n ; i++)
    count++;
}
```

→ $\underbrace{1, 4, 9, 16, \ldots, (\sqrt{n})}_{k}{}^2$

→ First Difference of series 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:

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

→ $\therefore$ $t(k) = K^2 + 0 + 0$

$n = K^2$

$K = \sqrt{n}$

Time Complexity : $\boxed{O(\sqrt{n})}$

A Quality Product by Ginger Your Lucky Brand

Teacher's Signature

Date : / /
Page No. :

7] void function (int n)

```
{
        int i, j, k, count = 0;
        For( i = n/2; i <= n; i++)
             for (j = 1; j <= n; j = j*2);
                    for (k = 1; k <= n; k = k*2);
                           count++;
}
```

i	j	k
$n/2$	$\log n$	$\log n \times \log n$
$\frac{n}{2}+1$	$\log n$	$\log n \times \log n$
⋮	⋮	⋮
n	$\log n$	$\log n \log n$

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

Teacher's Signature ____________

8]

```
function (int n)
{
    if (n==1) return;
    for (i=1 to n)
    {
        for (j=1 to n)
        {
            printf "*";
        }
    }
    function(n-3);
}
```

$$\underbrace{(n-3), (n-6) \; (n-9) \; \ldots\ldots}_{k} \; (1)$$

→ $a = n-3, \quad d = -3$

$1 = (n-3) + (k-1)(-3) \qquad [\because a_n = a + (n-1)d]$

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

$3k = n-1$

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

$= O(n * n^2)$

Time Complexity : $\boxed{O(n^3)}$

Date : / /
Page No. :

9] void function (int n)

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

For $i=1 \rightarrow j = n$ times

$i=2 \rightarrow j = n/2$ times

$i=3 \rightarrow j = n/3$ times

⋮ ⋮

$i=k \rightarrow j = n/k$ times

$\therefore$ $i=n$ & $j = \frac{n}{n}$ times

Time Complexity : $\left(n + \frac{n}{2} + \frac{n}{3} + \dots \frac{n}{n}\right)$

$$n\left[\frac{1}{1} + \underbrace{\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n}}_{\log n}\right]$$

$\therefore$ Time complexity : $\boxed{O(n \log n)}$

A Quality Product by Ginger Your Lucky Brand

Teacher's Signature

Date : / /
Page No. :

10] $f(n) = n^k$, $g(n) = c^n$
where $k=1$ of $c=2$

→ $F(1) = (1)^1$ $g(1) = (2)^1$
$f(1) < g(1)$

→ $f(2) = (2)^1$ $g(2) = (2)^2 = 4$
$f(2) < g(2)$

Satisfies O notation $f(n) \leq c\,g(n)$

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

$n_0^k = c_0 \cdot c^{n_0}$
$k=1$, $c=2$

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

Comparing : $\boxed{n_0 = 1}$, $\frac{n_0}{c_0} = 2$
$\boxed{c_0 = \frac{1}{2}}$

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

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

Teacher's Signature