Tutorial - 1

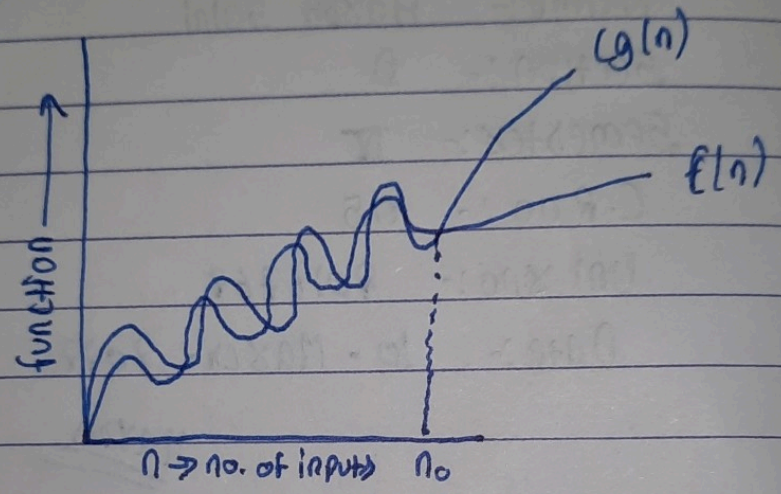
Name :- Harsh Saini
Section :- D
Semester :- IV
C.R no :- 05
Uni rno :- 2016762
Date :- 10 - March - 2022

Harsh

Q1 Asymptotic notations:-

They help you to find the complexity of algorithm when input is very large.

1.) Big O(o):-



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

iff $f(n) \leq c.g(n)$

$\forall\ n \geq n_0$

for some constant $c > 0$

$\Rightarrow g(n)$ is tight upper bound of $f(n)$

2. Big omega (Ω):-

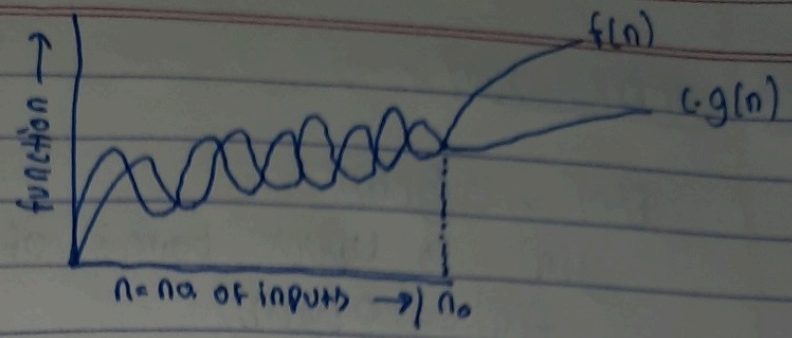
$f(n) = \Omega(g(n))$

$g(n)$ is tight lower bound of $f(n)$

$f(n) = \Omega(g(n))$

iff $f(n) \geq c.g(n)$

$\forall\ n \geq n_0$ for some constant $c > 0$



(iii) Theta (θ) :-

$f(n) = \theta(g(n))$

$g(n)$ is both tight upper & lower bound of $f(n)$

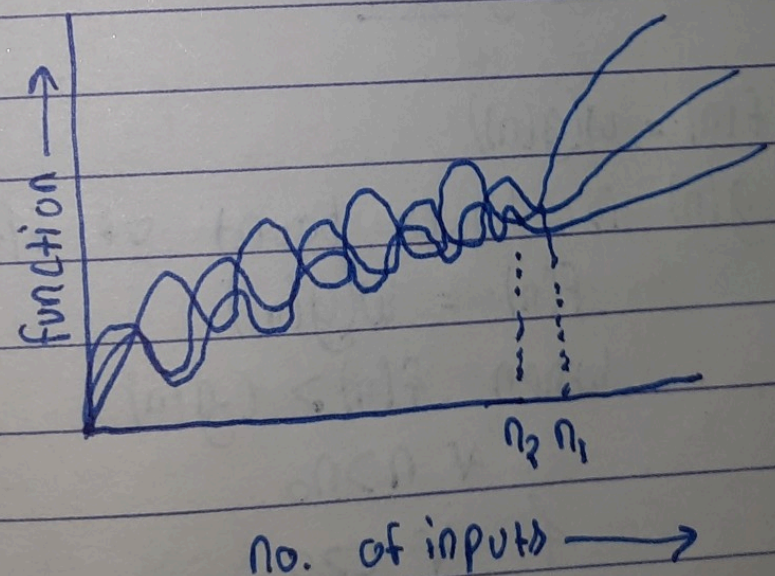
$$f(n) = \theta(g(n))$$

iff

$$c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$$

$$\forall \quad n \ge \max(n_1, n_2)$$

for some constant $c_1 > 0$ & $c_2 > 0$



4. <u>Small o(o)</u> :-

$f(n) = o(g(n))$

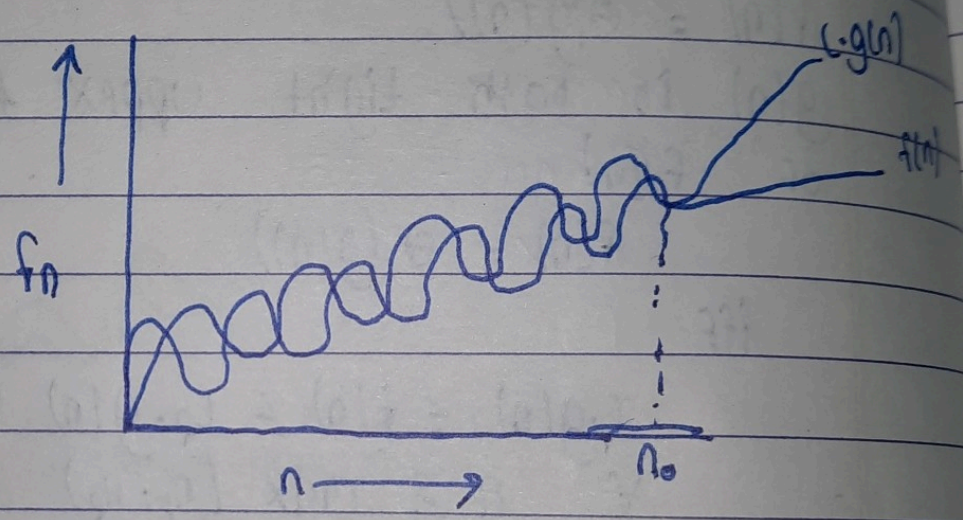
$g(n)$ is upper bound of fn $f(n)$

$f(n) = o(g(n))$

when $f(n) < c \cdot g(n)$

$\forall\ n > n_0$

& $\forall\ c > 0$



5. <u>Small omega (w)</u> :-

$f(n) = w(g(n))$

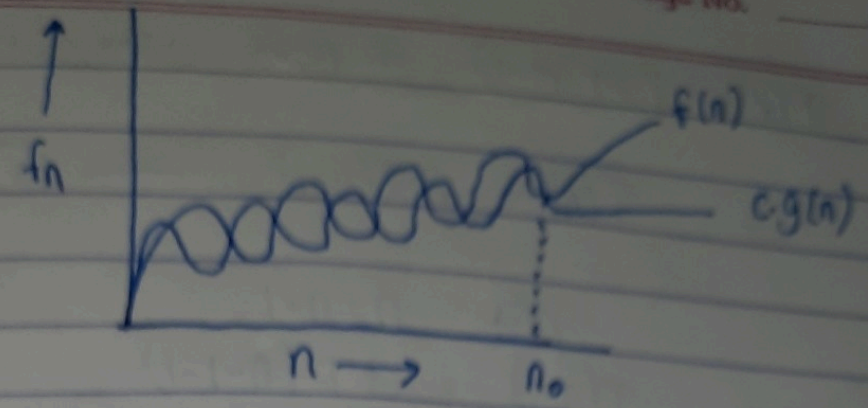
$g(n)$ is lower bound of $f(n)$

$f(n) = w(g(n))$

when $f(n) > c \cdot g(n)$

$\forall\ n > n_0$

& $\forall\ c > 0$



Q2 What should be time complexity of
for (i=1 to n) {i = i*2}

Sol^n:-

```
for (i=1 ton)          // i = 1, 2, 4, 8 ---- n
  { i = i*2 }          // O(1)
```

$\Rightarrow \sum_{i=1}^{n} 1+2+4+8+ ---- n$

G.P:- k^{th} value $\Rightarrow T_k = ar^{k-1}$

$\Rightarrow 1 \times 2^{k-1}$

$\Rightarrow n = 2^k$

$\Rightarrow 2n = 2^k$

$\Rightarrow \log 2n = k \log 2$

$\Rightarrow \log_2 2 + \log_2 n = k \log 2$

$\Rightarrow \log n + 1 = k$

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

$\Rightarrow = O(\log n)$

Q2 $T(n) = \{3T(n-1)$ if $n>0$ otherwise 1

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

put $n = n-1$

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

$$T(n-1) = 3T(n-2) \quad \text{--- (2)}$$

from 1 & 2 :-

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

$$= 9T(n-2) \quad \text{--- (3)}$$

putting $n = n-2$ in (1)

$$T(n-2) \cancel{T(n)} = 3(T(n-3) \quad \text{--- (4)}$$

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

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

putting, $n-k=0$

$$\Rightarrow n = k$$

$$\Rightarrow T(n) = 3^n[T(n-n)]$$

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

$$\Rightarrow T(n) = 3^n \times 1 \qquad [T(0)=1]$$

$$\Rightarrow \underline{\underline{T(n) = O(3^n)}}$$

4) $T(n) = \{2T(n-1) - 1$ if $n>0$, otherwise $1\}$

$$T(n) = 2T(n-1) - 1 \quad \text{—①}$$

let ~~A-1=A~~ $n = n-1$

$$\Rightarrow T(n-1) = 2T(n-2) - 1 \quad \text{—②}$$

from ① & ②

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

$$\Rightarrow T(n) = 4T(n-2) - 2 - 1 \quad \text{—③}$$

let $n = n-2$

$$\Rightarrow T(n-2) = 2T(n-3) - 1 \quad \text{—④}$$

from ③ & ④

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

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

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

$$\Rightarrow G.P = 2^{k-1} + 2^{k-2} + 2^{k-3} + \dots$$

$$a = 2^{k-1}$$

$$r = 1/2$$

$$\Rightarrow GP = \frac{a(1-r^n)}{1-r}$$

$$= \frac{2^{k-1}\left(1-(1/2)^n\right)}{\frac{1}{2}}$$

$$\Rightarrow 2^k\left(1-(1/2)^k\right)$$

$$= 2^k - 1$$

let $n-k=0$

$n=k$

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

$$\Rightarrow T(n) = 2^n - 1 - (2^n-1)$$

$$\Rightarrow T(n) = 2^n - (2^n-1)$$

$$\Rightarrow T(n) = O(1)$$

Q5 What should be time complexity of

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

$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \ldots$

$s = 1 + 3 + 6 + 10 + 15 + 21 \ldots\ldots n$

Sum of $s = 1 + 3 + 6 + 10 + \ldots + T_n$ ①

Date.
Page No.

also $b = 1+3+6+10+ \cdots T_{n-1} + T_n$ — ②

from ① — ②

$0 = 1+2+3+4+ \cdots n - T_n$

$\Rightarrow T_k = 1+2+3+4+ \cdots k$

$\Rightarrow T_k = \frac{1}{2} k(k+1)$

$\Rightarrow$ for k iterations

$1+2+3+ \cdots + k \quad \leq n$

$\Rightarrow \frac{k(k+1)}{2} \quad \leq n$

$\Rightarrow \frac{k^2+k}{2} \quad \leq n$

$\Rightarrow O(k^2) \quad \leq n$

$\Rightarrow k = O(\sqrt{n})$

$\Rightarrow T(n) = O(\sqrt{n})$

Q6 Time Complexity of -

```
void fn(int n)
{
    int i, count = 0;
    for (i=1; i*i<=n; ++i)
        count++          //O(1)
}
```

as $i^2 <= n$

$\Rightarrow i <= \sqrt{n}$

$i = 1, 2, 3, 4, ----, \sqrt{n}$

$\sum_{i=1}^{n} 1+2+3+4+ ---- + \sqrt{n}$

$\Rightarrow T(n) = \frac{\sqrt{n} \times (\sqrt{n}+1)}{2}$

$\Rightarrow T(n) = \frac{n \times \sqrt{n}}{2}$

$\Rightarrow \underline{\underline{T(n) = O(n)}}$

Q7 Time Complexity of

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

for $k = k*2$

$k = 1, 2, 4, 8, ------ n$

Date.
Page No.

$\Rightarrow$ G.P $\Rightarrow$ $a=1$, $x=2$

$$= \frac{a(x^n-1)}{x-1}$$

$$\Rightarrow \frac{1(2^k-1)}{1}$$

$$n \Rightarrow 2^k$$

$$\Rightarrow \underline{\underline{\log n = k}}$$

$\Rightarrow$

i	J	K
1	$\log n$	$\log n * \log n$
2	$\log n$	$\log n * \log n$
⋮	⋮	⋮
n	$\log n$	$\log n * \log n$

$$\Rightarrow O(n * \log n * \log n)$$

$$\Rightarrow \underline{\underline{O(n \log^2 n)}}$$

(*2)

8 Time complexity of function (int n)

```
{
    int (n==1)
    return n;        // O(1)
```

```
for (i=1 to n)        // ---- O(n)
{
    for (j=1 to n)    // ---- O(n²)
    }
        function(n-3);        T(n/3)
}
```

$\Rightarrow T(n) = T(n/3) + n^2$

$\Rightarrow a=1, \; b=3, \; f(n)=n^2$

$c = \log_3 1 = 0$

$\Rightarrow n^0 = 1 > (f(n) = n^2)$

$\Rightarrow \underline{\underline{T(n) = \Theta(n^2)}}$

Q9 Time Complexity of -

```
void function (int n)
{
    for (i=1 to n)                    // O(n)
    {
        for (j=1 ; j<=n ; j=j+1)
            printf("*")               // O(1)
    }
}
```

Date.
Page No.

for $i=1 \Rightarrow j = 1,2,3,4,\ldots n = n$
for $i=2 \Rightarrow j = 1, 3, 5, \ldots n = n/2$
for $i=3 \Rightarrow j = 1, 4, 7, \ldots n = n/3$
$\vdots$
for $i=n \Rightarrow j = 1 \ldots 1$

$$\Rightarrow \sum_{j=n}^{1} n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \ldots + 1$$

$$\Rightarrow \sum_{j=n}^{1} n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots \frac{1}{n}\right]$$

$$\Rightarrow \sum_{j=n} n[\log n]$$

$$\Rightarrow T(n) = [n\log n]$$
$$\Rightarrow T(n) = O(n\log n)$$

Q10

as given n^k & c^n
relation b/w n^k & c^n is
$n^k = O(c^n)$
as $n^k \le c^n$
$\forall\, n \ge n_0$ & some constant $a > 0$
for $n_0 = 1$
$c = 2$
$\Rightarrow 1^k \le a\, a^1$
$\Rightarrow n_0 = 1$ & $c = 2$