Tutorial-1

Name: Ambuj Tripathi

Section: CST- SPL-1

Semester: IV

C. R no. : 48

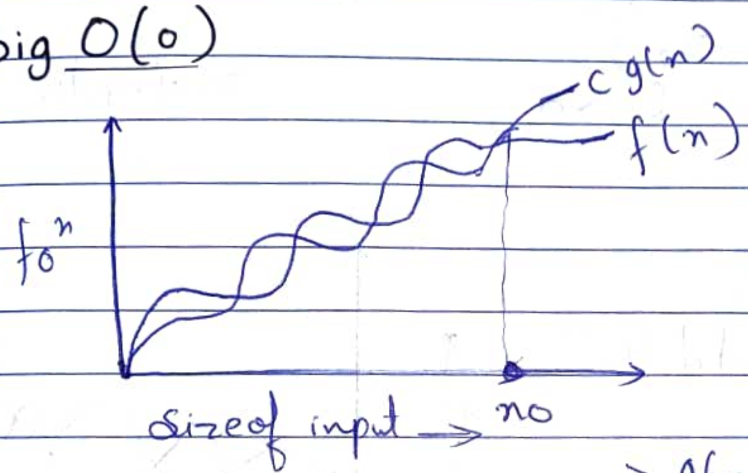
Uni Roll no. 2017459

Date: 10-March-2022

Q1

Ans 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 (o)



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

If $f(n) < c.g(n)$ $\forall$ $n \geq n_0$

for some constant $c > 0$

⇒ g(n) is tight upper bound of f(n)

2) Big Omega (Ω)

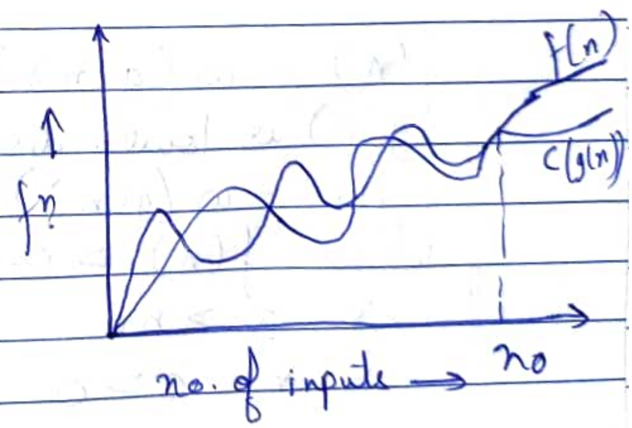
$f(n) = \Omega(g(n))$

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

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

if $f(n) \geq c(g(n))$

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



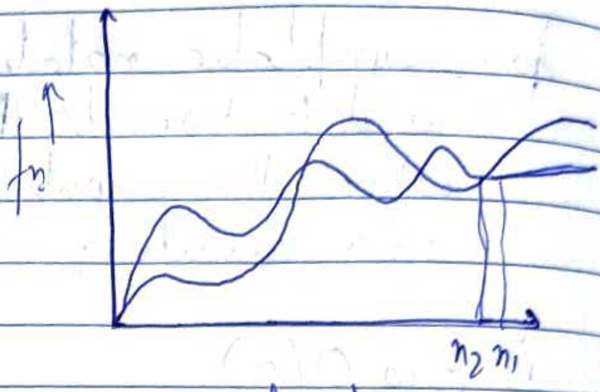
Date.
Page No.

3) Theta (Θ)

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

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

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



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

$\forall \; n \ge C\max(n_1, n_2)$

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

4) Small o (O)

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

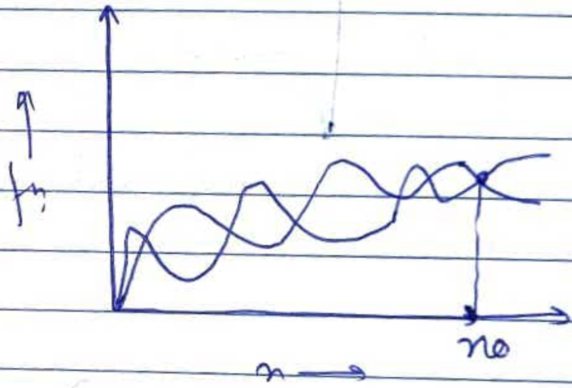
g(n) is upper bound of f(n)

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

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

$\forall \; n > n_0$.

& $\forall \; c > 0$



5) Small omega (ω)

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

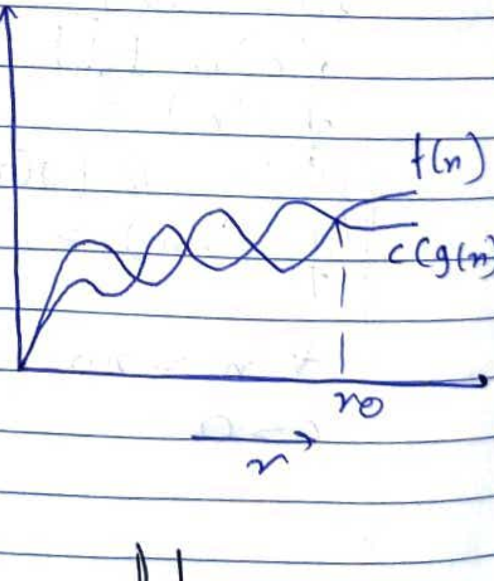
g(n) is lower bound of f(n)

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

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

$\forall \; n > n_0$

& $\forall \; c > 0$



Date.
Page No.

Q2.

Sol for (i = 1 to n) { i = i × 2; }

// i = 1, 2, 4, 8, ... n
// O(1)

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

GP k^{th} value $\Rightarrow T_K = a r^{K-1}$

$= 1 \times 2^{K-1}$

$\Rightarrow n = 2^K$

$\Rightarrow 2n = 2^K$

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

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

$\Rightarrow \log n + 1 = K$

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

$= O(\log n)$.

Q3 $T(n) = \{ 3T(n-1)$ if $n > 0$ otherwise $1 \}$

Sol $T(n) = 3T(n-1) \longrightarrow$ 1

put $n = n-1$

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

From 1 & 2

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

$= 9T(n-2) \longrightarrow$ 3

Putting $n = n-2$ in 1

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

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

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

Date. ______
Page No. ______

putting $n-K=0$

$\Rightarrow n=K$

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

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

$T(n) = 3^n \times 1$

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

Date.
Page No.

Q4 $T(n) = \{2T(n-1)-1$ if $n>0$, otherwise $1\}$

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

Let $n = n-1$

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

From (1) & (2)

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

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

Let $n = n-2$

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

From (3) & (4)

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

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

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

$\Rightarrow GP = 2^{k-1} + 2^{k-2} + 2^{k-3} + \dots 1$

$a = 2^{k-1}$

$r = 1/2$

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

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

$= 2^k - 1$

Let $n - k = 0 \quad n = k$

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

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

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

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

Date.
Page No.

Q3 Time Complexity:

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

Sol: i = 1 2 3 4 5 ...

s = 1 + 3 + 6 + 10 + 15 ... n

Sum of s = 1 + 3 + 6 + 10 + ... + n → (1)

also s = 1 + 3 + 6 + 10 + ... n−1 + n → (2)

From (1) − (2)

$0 = 1 + 2 + 3 + 4 + \dots n - [n$

$T_k = 1 + 2 + 3 + 4 \dots k$

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

For k iterations.

$1 + 2 + 3 + \dots + k <= n$

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

$\frac{k^2 + k}{2} <= n$

$O(k^2) <= n$

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

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

Date.
Page No.

Q6 Time Complexity:

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

Sol → $i^2 <= n$

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

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

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

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

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

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

Date. ________
Page No. ________

Q7 Time Complexity:

```
void fn(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++;
}
```

⇒ for k = k * 2

k = 1, 2, 4, 8, ... n

⇒ GP = a = 1 r = 2

$R_0 = \frac{a(r^n - 1)}{r - 1}$

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

$\Rightarrow n \Rightarrow 2^k$

$\Rightarrow n = \log n > k$

⇒

i	j	k
1	log n	log n * log n
2	log n	log n * log n
⋮	⋮	⋮
n	log n	log n * log n

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

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

Date.
Page No.

Q8 Time Complexity of

```
function (int n)
{
    int (n == 1)
        return;
    for (i = 1 to n)
    {
        for (j = 1 to n)
        {
            print ('*');
        }
    }
    function (n - 3);
}
```

Sol → $T(n) = T(n/3) + n^2$

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

$\Rightarrow c \log_3 1 = 0$

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

$\Rightarrow T(n) = \theta(n^2)$

Date.
Page No.

Q1 Time Complexity

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

Sol 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 $i=3 \Rightarrow j = 1, 4, 7, \ldots n = n/3$

⋮

for $i=n \Rightarrow j = 1 \ldots 1$

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

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

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

$$\Rightarrow T(n) = n\log n$$

$$T(n) = O(n\log n)$$

Date.
Page No.

Q10 for func. n^k & c^n. what is asymptotic relation? Assume $k >= 1$ & $c > 1$ are constant. Find out value of c & no. for which relation holds.

Sol) As given n^k & c^n

Relation b/w n^k & c^n is

$$n^k = O(c^n)$$

As $n^k \leq a c^n$
$\forall\ n \geq n_0$ & some constant $a > 0$

for $n_0 = 1$
$c = 2$
$\Rightarrow 1^k \leq a_2'$

$\Rightarrow \underline{n_0 = 1}$ & $\underline{c = 2}$