Tutorial - 1

Name: Saatwik Semwal

Section: CST SPL - 1

Semester: 4

Rollno: 15

Univ. Rollno: 2017538

Date: 10 March 2023

Signature:

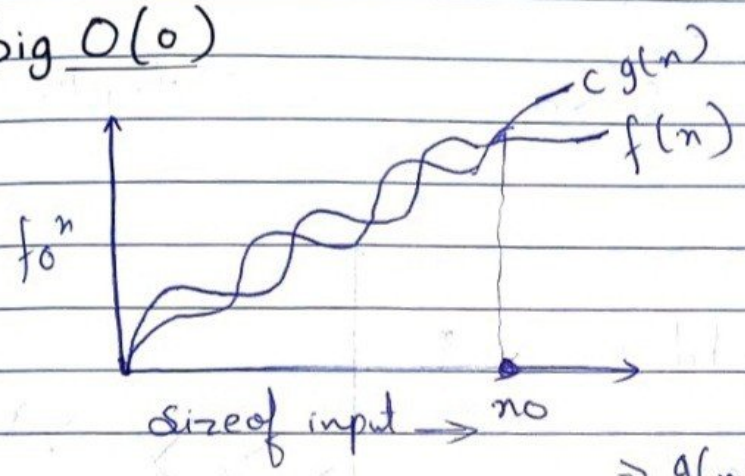
Design & Analysis of Algorithm

Date. _____
Page No. _____

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)

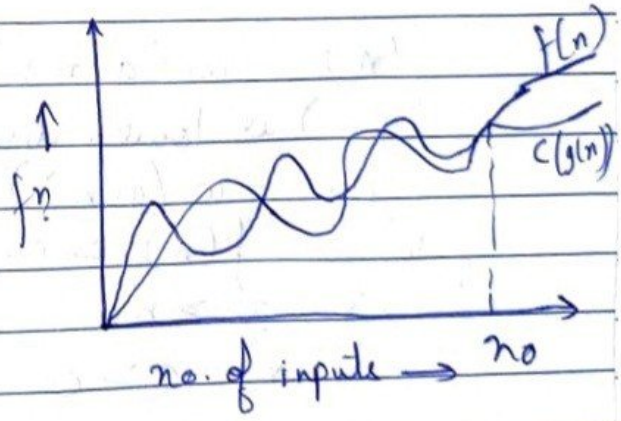


$f(n) = O(g(n))$
If $f(n) \leq c \cdot g(n)$
$\forall\ n \geq n_0$
for some constant $c > 0$

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

2) Big Omega (Ω)

$f(n) = \Omega(g(n))$
$g(n)$ is tight lower bound of $f(n)$
$f(n) = \Omega(g(n))$
if $f(n) \geq c(g(n))$
$\forall\ n \geq n_0$ for some constant $c > 0$

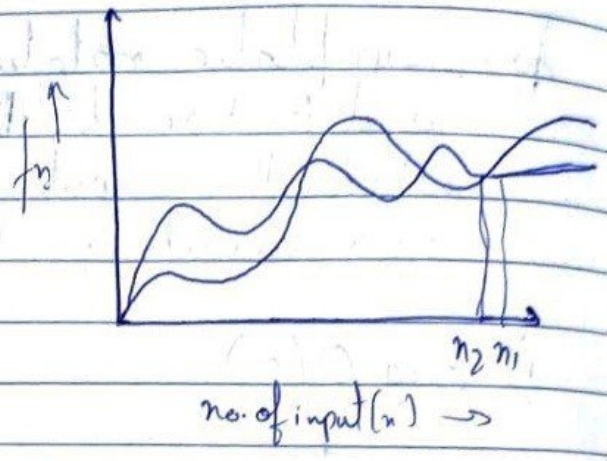


3) Theta (θ)

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

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

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



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

$\forall\ n \geq c_{max}(n_1, n_2)$

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

4) Small o (o)

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

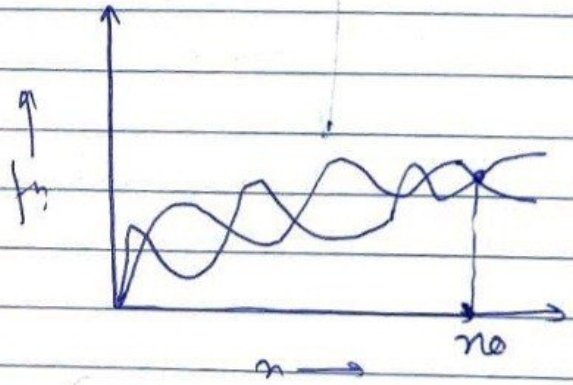
$g(n)$ is upper bound of $f(n)$

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

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

$\forall\ n > n_0$

& $\forall\ c > 0$



5) Small omega (ω)

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

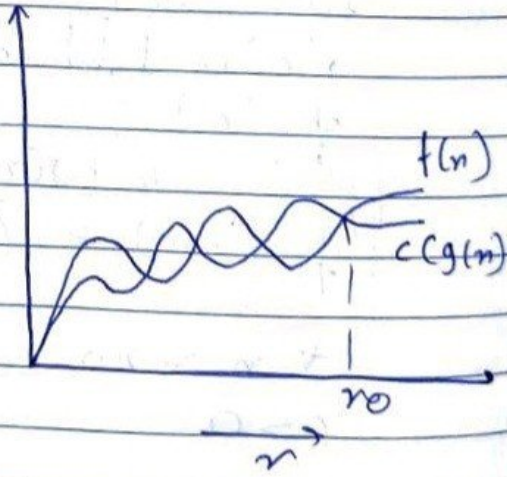
$g(n)$ is lower bound of $f(n)$

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

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 x 2;}

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

$\Rightarrow \sum_{i=1}^{n}$ 1 + 2 + 4 + 8 + ... 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 $n - k = 0$

$\Rightarrow n = k$

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

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

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

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

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

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

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

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

$T(n) = O(1)$

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 + . . . n − [n

$T_K = 1 + 2 + 3 + 4 \cdots K$

$T_K = \frac{1}{2} K(K+1)$

For K iterations.

$1 + 2 + 3 + \cdots + K <= n$

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

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

$O(K^2) <= n$

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

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

Q6 Time Complexity:

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

Sol ⇒ $i^2 <= n$

⇒ $i <= \sqrt{n}$

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

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

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

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

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

Q7 Time Complexity:

```
void fn(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

⇒ GP = a = 1 r = 2

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

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

$\Rightarrow n \Rightarrow 2^k$

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

→

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

⇒ $O(n * \log n * \log n)$

⇒ $O(n \log^2 n)$

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);
}
```

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

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

$\Rightarrow c \log_3 1 = 0$

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

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

Q1 Time Complexity

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

Sol

for $i=1 \Rightarrow j = 1, 2, 3, 4, \cdots n = n$

for $i=2 \Rightarrow j = 1, 3, 5, \cdots n = n/2$

for $i=3 \Rightarrow j = 1, 4, 7, \cdots n = n/3$

⋮

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

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

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

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

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

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

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 & n_0 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 $n^k \leq ac^n$

$\forall\ n \geq n_0$ & some constant $a > 0$

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a_2 \cdot 1$

$\Rightarrow$ $n_0 = 1$ & $c = 2$