Name - Ujjwal Gupta
Section → CST
Sem → 4
Class Roll no - 6
University Roll no → 2017573

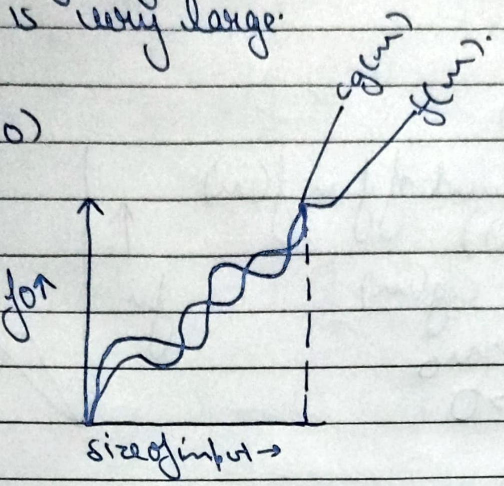
classmate
Date ______
Page ______

Q1. Asymptotic Notations

↳ Tending to infinity

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

1.) Big O (O)

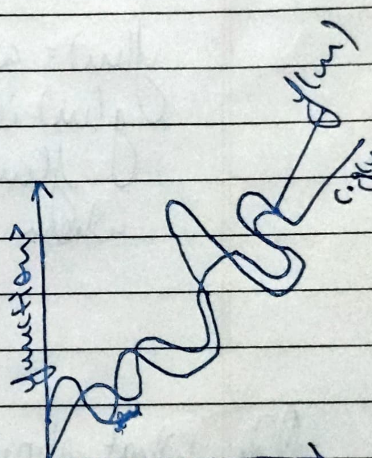


$f(n) = O(g(n))$
iff $f(n) \leq c \cdot g(n)$.
$\forall\ n \geq n_0$
for some constant $c > 0$

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

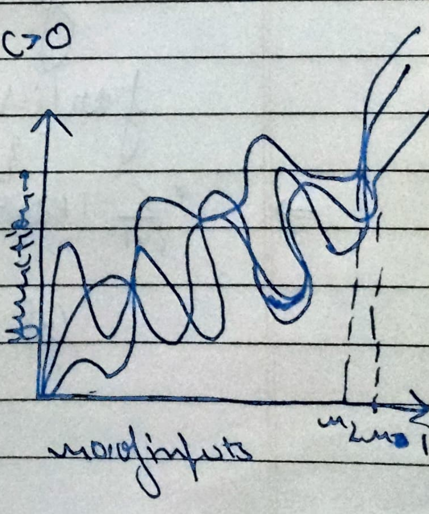
ii.) Big Omega (Ω)

$f(n) = \Omega(g(n))$
$g(n)$ is tight lower bound of $f(n)$



$f(n) = \Omega(g(n))$
iff $f(n) \geq c \cdot g(n)$
$\forall\ n \geq n_0$ for some constant $c > 0$

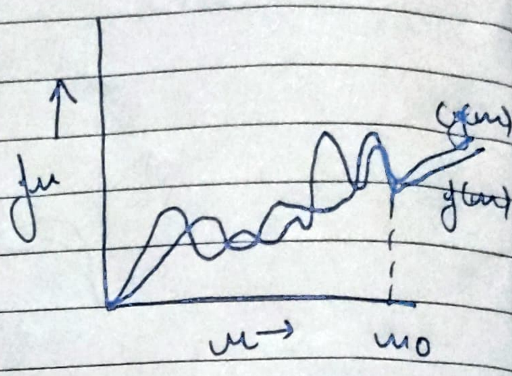
iii.) Theta (Θ)
$f(n) = \Theta(g(n))$.
$g(n)$ is both 'tight' upper & lower bound of fn ($f(n)$)
$f(n) = \Theta(g(n))$



If $c_1 g(n) \le f(n) \le c_2 \cdot g(n)$
$\forall n \ge \max(n_1, n_2)$.
for some constant $c_1 > 0$ & $c_2 > 0$

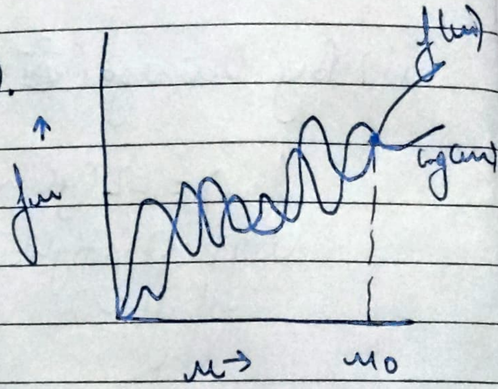
4) Small o(O)

$f(n) = O(g(n))$
$g(n)$ is upper bound of fn $f(n)$
$f(n) = O(g(n))$
when $f(n) < c.g(n)$
$\forall n > n_0$
and $\forall c > 0$



5) Small omega (ω)

$f(n) = \omega(g(n))$
$g(n)$ is lower bound of fn $f(n)$.
$f(n) = \omega(g(n))$
when $f(n) > c.g(n)$
$\forall n > n_0$
& $\forall c > 0$



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

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

⇒ $\sum_{i=1} 1+2+4+8+ \text{----} \; n$

GP k^{th} value ⇒ $T_k = a r^{k-1}$
⇒ $1 \times 2^{k-1}$
⇒ $n = 2^k$

$\Rightarrow 2n = 2^k$

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

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

$\Rightarrow \log n + 1 = k$

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

$= O \log n$

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

$T(n) = 3T(n-1)$ — (1)

put $n = n-1$

$T(n-1) = 3T(n-2)$ — (2)

from (1) & 2

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

$= 9T(n-2)$ — (3)

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

$T(n) = 3T(n-3)$ — (4)

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

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

putting $n-k=0$

$\Rightarrow n = k$

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

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

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

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

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

$T(n) = 2T(n-1) - 1$ — ①

Let $n = n-1$

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

from ① and ②

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

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

Let $n = n-2$

$\Rightarrow T(n-2) = 2T(n-3 - 1$ — ④

from ③ and ④

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

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

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

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

$a = 2^{k-1}$

$r = 1/2$

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

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

$= 2^k (1 - (1/2)^k)$

$= 2^k - 1$

Let $n - k = 0$

$\Rightarrow n = k$

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

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

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

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

classmate

Date
Page

Q5. What should be time complexity of

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

i=1 2 3 4 5 6 - - - -

s = 1 + 3 + 6 + 10 + 15 + 21 - - - n

sum of s = 1+3+6 +10 + - - - + T_n — (1)

also s = 1+3+ 6+10+ - - T_{n-1} + T_n — (2)

from (1) & (2)

0 = 1+2+3+ 4+ - - - n − T_n

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

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

⇒ for k iterations

$1+2+3+ - - - - + k <= n$

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

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

$\Rightarrow O(k^2) <= n$

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

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

classmate
Date
Page

Q6. Time complexity of

```
void fun2(int n)
{ int i, count=0;
for( i=1; i* i<=n; ++i)
    count++          // O(1)
}
```

as $i^2 <= n$

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

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

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

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

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

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

Q7 Time complexity of

```
void fun(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 $k = k*2$

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

classmate

Date ____

Page ____

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

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

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

$n \Rightarrow 2^k - 1$

$\Rightarrow \log n \Rightarrow k$

$\rightarrow$

i	j	k
1	$\log n$	$\log n * \log n$
2	$\log n$	$\log n * \log n$
|	|	|
|	|	|
n	$\log n$	$\log n * \log n$

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

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

Q.8 Time complexity of

```
function (int n)
    int (n==1)
    ~~return;~~
~~for(i=1 to n)~~ return;        // O(1)
for(i=1 to n)                   // i=1,2,3,4 --- n => O(n)
{
    for (j=1 to n)   // j=1,2,3,4 --- n => O(n^2)
    {
        print ('*');
    }
}
```

function(n-3);   T(n/3)
}

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

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

$c = \log_3 1 = 0$

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

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

Q9 Time complexity of

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

for $i=1 \Rightarrow j=1,2,3,4 \text{ ---- } n = n$

for $i=2 \Rightarrow j=1,3,5, \text{ ---- } n = n/2$

for $i=3 \Rightarrow j=1,4,7, \text{ ---- } n = n/3$

for $i=n \Rightarrow j=1 \text{ ---- }$ 1

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

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

$$\sum_{j=n} n(\log n)$$

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

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

Q10. for function, n^k and c^n, what is the asymptotic relation b/w these function?
assume that $k>=1$ and $c>1$ are constant
Find out the value of c and n_0 for which relation holds.

as given n^k and c^n

relation b/w n^k and c^n is

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

as $n^k \leq ac^n$

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

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a^1$

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