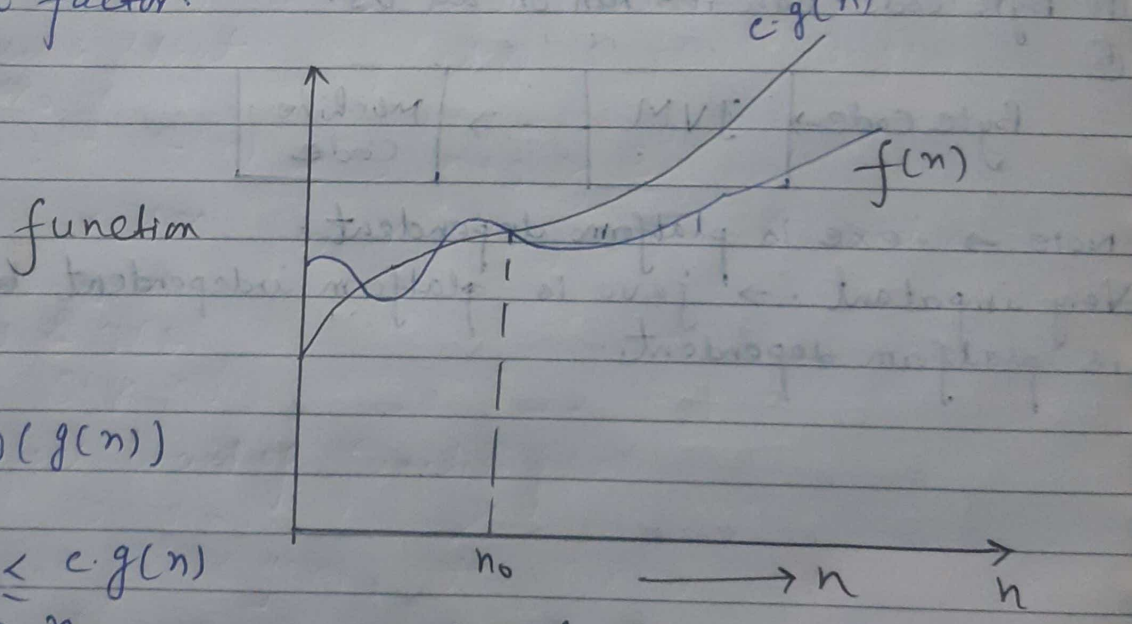
Name:→ Sandeep Singh U Roll.NO: 2017547
Section:→ CST-SPL-2 Class RollNO : 02

Date Sandeep

Design and Analysis of Algorithms
Tutorial-1

Q.(1) Asymptotic notations:→ Asymptotic notations are used to represent the complexities of algorithms for asymptotic analysis. These notations are mathematical tools to represent the complexities. There are three notations that are commonly used.

① Big O Notation:→ Big O notation gives an upper bound for a function $f(n)$ to within a constant factor.



$f(n) = O(g(n))$
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 $f(n)$
or
$f(n) = O(g(n))$
these exist positive constant c and n_0 such that

$$0 \leq f(n) \leq g(n) \quad , \forall \; n \geq n_0$$

Spiral

Date.....

Eg. ① $f(n) = n^2 + n$

$g(n) = n^3$

$n^2 + n \leq c \cdot n^3$

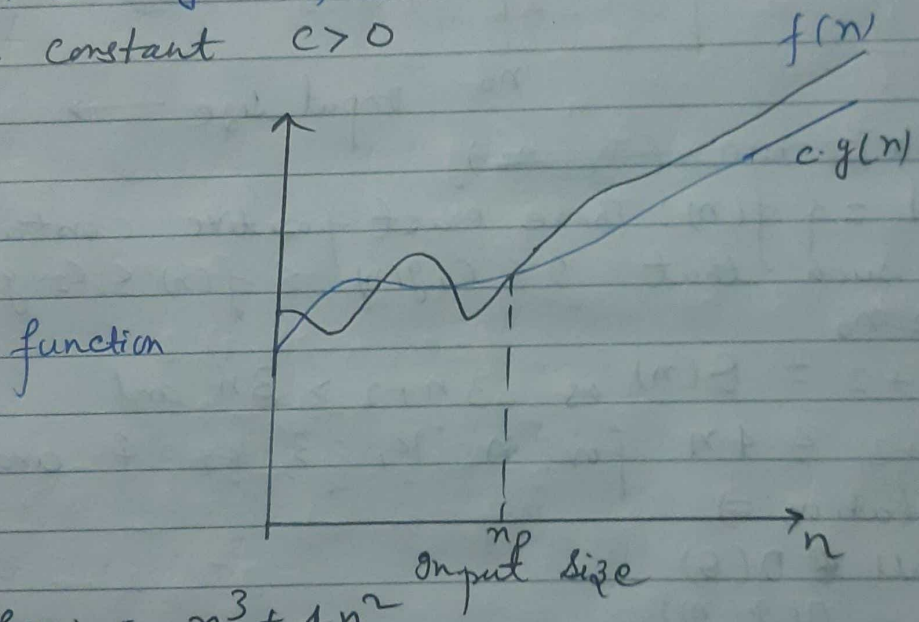
$n^2 + n = O(n^3)$

(ii) Big Omega (Ω) :-

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

iff $f(n) \geq c \cdot g(n), \forall \; n \geq n_0$

for some constant $c > 0$



Eg(1) $f(n) = n^3 + 4n^2$

$g(n) = n^2$

$n^3 + 4n^2 = \Omega(n^2)$

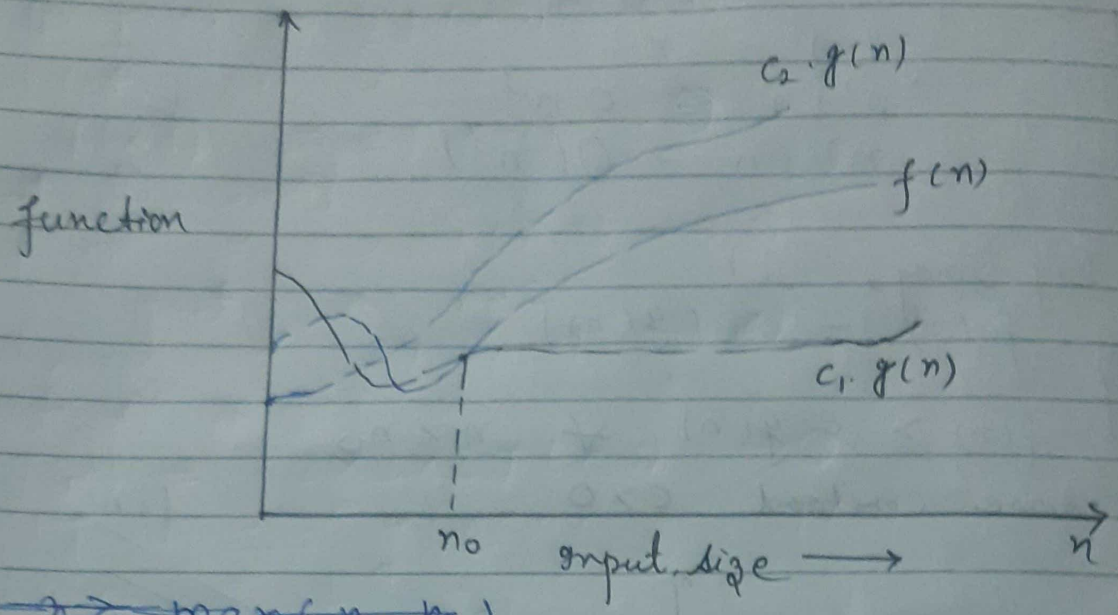
(iii) Big Theta (Θ) :-

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

$g(n)$ is both "tight" upper and lower bound of function $f(n)$

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

iff $c \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$



~~$\forall \quad n \geq \max(n_1, k_2)$~~

$\Omega(g(n)) = \{ f(n)$: There exist positive constant c_1, c_2 and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 \cdot g(n)$ $\forall \quad n \geq n_0$

Eg $3n+2 = \Theta(n)$ as $3n+2 \geq 3n$ and $3n+2 \leq 4n$ for n, $k_1 = 3$, $k_2 = 4$ and $n_0 = 2$

other Notations: ⇒

(iv) Small O(θ)

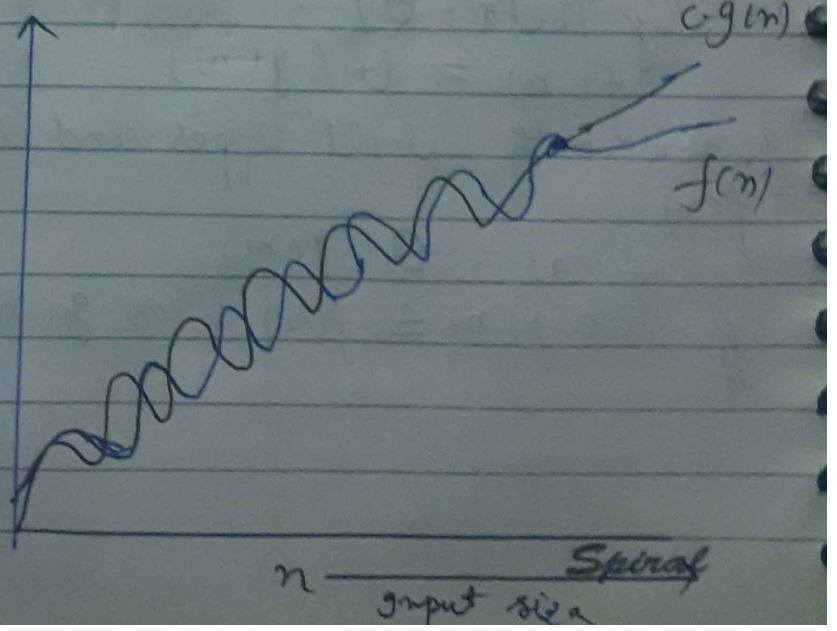
$f(n) = O(g(n))$

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

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

when $f(n) < c \cdot g(n)$, $\forall \quad n > n_0$ and

$\forall$ Constants, $c > 0$



Eg $f(n) = \text{function } n^2$

$g(n) = n^3$

$n^2 = O(n^3)$

Spiral

Date.....

① Small omega (ω) :→

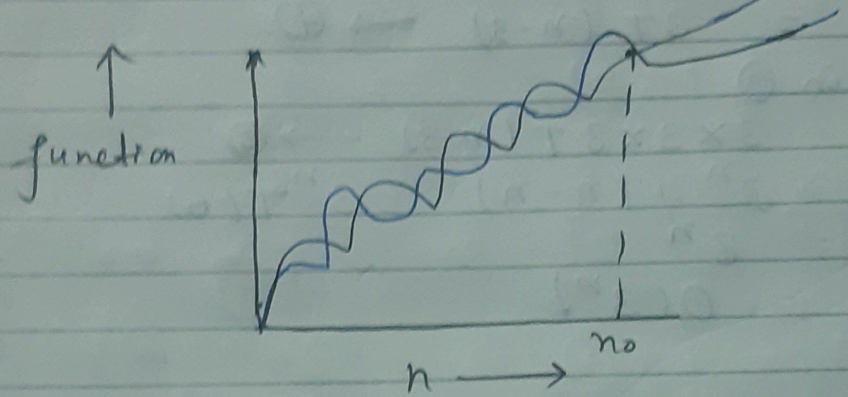
$f(n) \in \omega(g(n))$

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

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

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

$\forall$ $n > n_0$ and $\forall$ constants $c > 0$



Eg $f(n) = 4n + 6$

$g(n) = (1)$

Q: ② for(i = 1 to n)

{ i = i * 2 }

- $i = 1, 2, 4, 8, 16 \dots n$ (G.P)

$a = 1, \quad r = 2/1 = 2$

k^{th} term of G.P $= ar^{k-1}$

$n = 1 * 2^{k-1}$

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

$2n = 2^k$

$\log(2n) = k \log_2(2)$

$k = \log(2n)$

$k = \log_2(2) + \log_2(n)$

$k = 1 + \log_2(n)$

Time Complexity $= O(1 + \log(n))$

$= O(\log n)$

Spiral

Date.....

Ans No. (3) $T(n) = 3T(n-1) \longrightarrow (1)$

Let $n = n-1$

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

put (2) in (1)

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

put $n = n-2$

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

put (4) in (3)

$T(n) = 3 \times 3 \times 3T(n-3) \longrightarrow (5)$

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

$T(n) = 3^n$

$= O(3^n)$

Ans No. (4). $T(n) = 2T(n-1) - 1$

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

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

$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3}$

$= \cdots 2^2 - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} \cdots - 2^0$

$= 2^n [2^n - 1]$

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

$= 1 \Rightarrow$

$T(n) = O(1)$ Ans

Date.....

ANS NO: (5). i=1, s=1;

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

↓ stop

s 1 3 6 10 15 21 ..... (> n)

i 1 2 3 4 5 6 ..... (k)

Let k iteration while loop runs.

Observation: s is of sum of first n natural number :→

For i = k

s = sum of first n natural numbers

for k^{th} s the condition is $\frac{(k)(k+1)}{2} > n$ for while loop

$\therefore \frac{k^2+k}{2} > n$

$\therefore k^2 > n \Rightarrow k = O(\sqrt{n})$

ANS NO: (6) ⇒

```
Void function (int n)
    int i, count = 0;  → O(1)
  for(i=1; i*i <= n; i++)  ← √n
    count++ ;  ← Time Complexity ?
}
```

Time complexity = $O(1) + O(\sqrt{n})$

$\Rightarrow O(\sqrt{n})$ Ans

Date.........

Ans (7) Void function (int n)
{
int i, j, k, count = 0; → O(1)
for (i = n/2; i <= n; i++) → O(n)
for (j = 1; j <= n; j = j * 2) → O(log n)
for (k = 1; k <= n; k = k * 2) ⟷ O(log n)
count ++

}

for (j = 1; j <= n; j = j * 2

$j = 1 = 2^0$
$j = 2 = 2^1$
$j = 4 = 2^2$
$j = 8 = 2^3$

G.P

find pth j =

$a = 1, r = 2$

$\therefore j = ar^{p-1} = n$

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

$\Rightarrow n = 2^{p-1} \longrightarrow (A)$

taking log on both side of eq^n (A)

$\log_2(n) = (p-1)\log_2(2)$

$\log_2(n) = (p-1)$

$p = \log_2(n) + 1$

$= O(\log n)$

Spiral

O(1)

Date.....

for(k=1; k<=n; k=k*2)

k=1 = 2^0

k=2 = 2^1

k=4 = 2^2

k=8 = 2^3

⋮

for p^{th} (G.P)

a = 1, r = 2/1 = 2

for p^{th} term $k = (1)2^{p-1} = n$ (by $ar^{terms-1}$)

$n = k = 2^{p-1}$ → (B)

taking log on both sides

$\log(n) = (p-1)\log_2(2)$

$\Rightarrow p = \log_2(n) + 1$

$\Rightarrow O(\log_2(n)+1) \Rightarrow O(\log n)$

$\therefore n \times \log n \times \log(n) \Rightarrow n(\log n)^2$

Time Complexity = $O(n(\log n)^2)$

Date

Ans No (9): Void function(int n)
{ for (i=1 to n)
{ for (j=1 ; j<=n ; j=j+i)
{ printf("*")
}
}

i = 1 j : 1 to n n-times	i = 2 j : 1 to n . j will increment n/2 times

i = 3
j : 1 to n . .
increment n/3 times

Now
for i = k
j : 1 to n
increment = n/k

for i = n
j = 1 to n
n/n times increment

$\therefore T(n) = n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}$

$T(n) = n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$

$= n \log(n)$

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

Ans NO: 10.

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

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

as $n^k \leq a \cdot c^n \; \forall \; n \geq n_0$ and some constant $a > 0$

for $n_0 = 1$

$c = 2$

$1^k \leq a \cdot 2^1$

$n_0 = 1$ and $c = 2$ Ans

ANS NO: (8)

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

$\Rightarrow T(n) = (1)T(n/3) + n^2$

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

$c = \log_b(a) = \log_3(1) \Rightarrow 0$

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

$\therefore T(n) = O(n^2)$ Ans