Name - Anshul Kumar
Sec - SPL 2

Tutorial - I

Design And Analysis of Algorithms

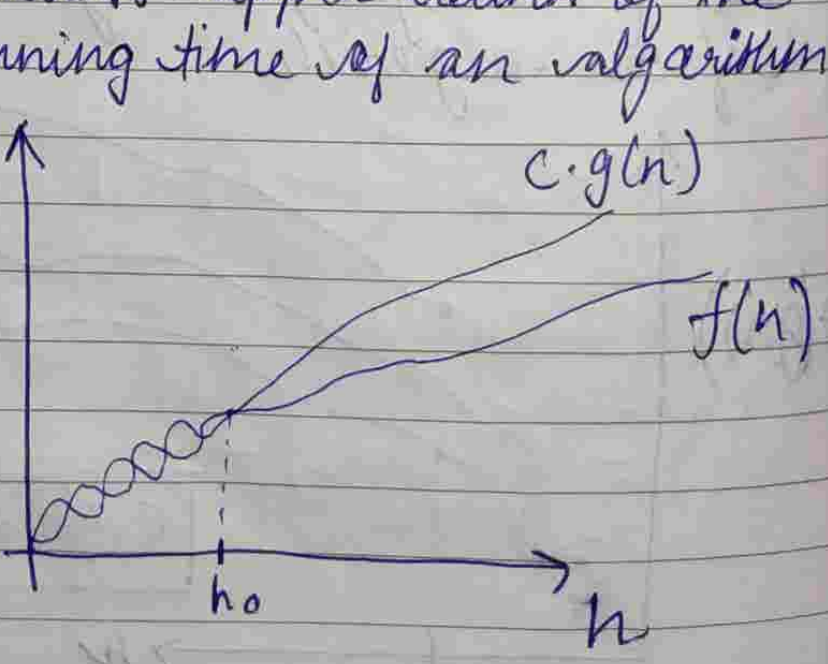
Ans 1. Asymptotic Notations are the Mathematical Notations used to describe the running time of an Algorithm when the inputs tends to a limiting value.

- The efficiency is measured with the help of asymptotic notations.

(1) Big-O notation

- represents upper bound of the running time of an algorithm

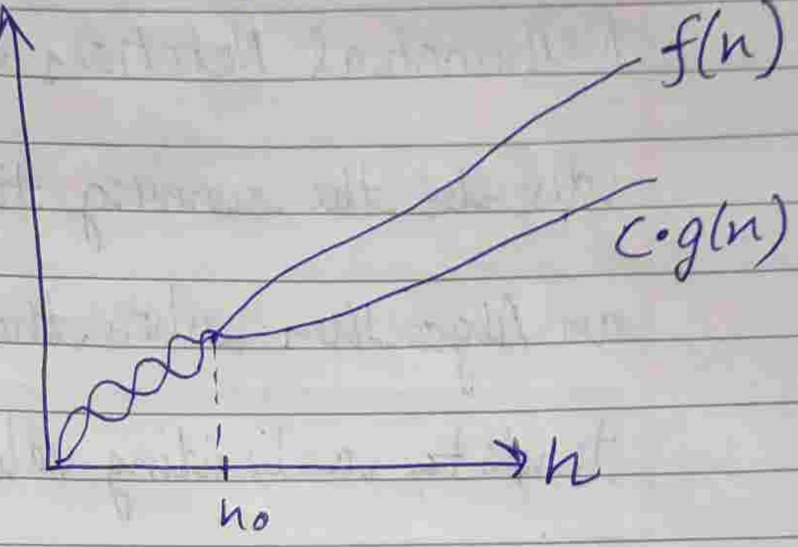
$f(n) = O(g(n))$



(2) Omega Notation (Ω)

- represents the lower bound of the runningtime of an algorithm.

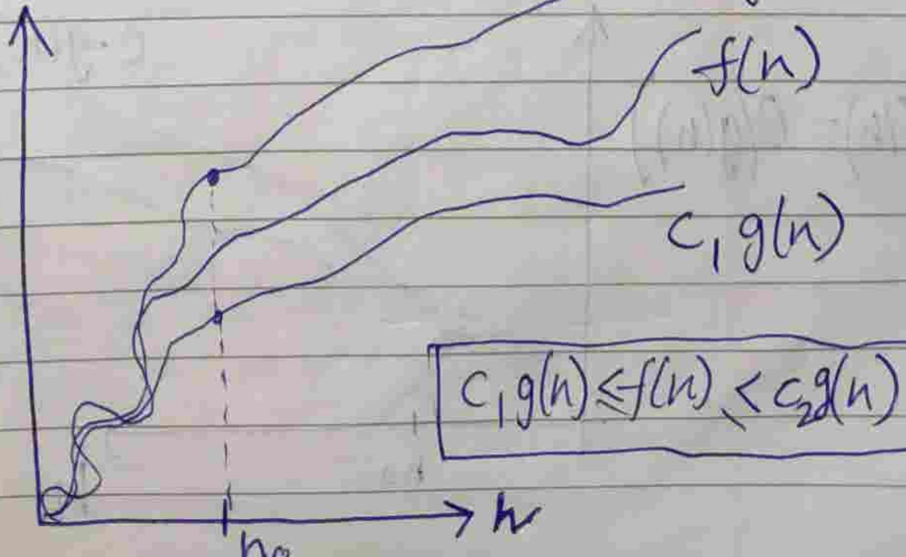
$f(n) = \Omega(g(n))$



(3) Theta Notation (θ)

- encloses the function f(n) from above and below
- used for analyzing average-case complexity of an algorithm

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



$\boxed{c_1 g(n) \leq f(n) \leq c_2 g(n)}$

classmate

Date

Page

Ans2.

```
for(i=1 to n){
    i=i*2;
}
```

$i = 1, 2, 4, 8, 16 \ldots n$

$a = 1, \ r = 2$

$t_k = ar^{k-1}$

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

$n = 2^{k-1}$ (taking log both sides)

$\log_2 n = \log_2 2^{k-1}$

$\because (\log_2 2^a = a)$

$\log_2 n = k-1$

$k = 1 + \log_2 n$

T.C. = $\underline{\underline{O(\log_2 n)}}$ Ans.

Ans 3. $T(n) = \begin{cases} 3T(n-1) & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$

putting $n=0$,

$T(0) = 1 \qquad (n \ngtr 0)$

$T(n-1) = 3T(n-2)$ —— (I)

putting (I) in $T(n)$

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

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

$T(n) = 9T(n-2)$ —— (II)

$T(n-2) = 3T(n-3)$ —— (III)

putting (III) in (II)

$T(n) = 27T(n-3)$ —— (IV)

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

put $n-k=0$
$k=n$

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

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

$T(0) = 1$ (as $n \not> 0$)

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

$\boxed{\text{Time Complexity} = O(3^n)}$ Ans

Ans 4. $T(n) = \begin{cases} 2T(n-1) - 1 & \text{if } n>0 \\ 1 & \text{else} \end{cases}$

$T(n) = 2T(n-1) - 1$ —— (1)

put $n = n-1$

$T(n-1) = 2T(n-2) - 1$ —— (11)

put $T(n-1)$ from (11) to (1)

$T(n) = 4T(n-2) - 3$ —— (111)

put $n = n-2$

$T(n-2) = 2T(n-3) - 1$ —— $\textcircled{IV}$

put $T(n-2)$ from $\textcircled{IV}$ to $\textcircled{III}$

$T(n) = 8T(n-3) - 7$ —— $\textcircled{V}$

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

put $n-k = 0$

$k = n$, $\quad (\because T(0) = 1)$

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

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

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

$\boxed{T(n) = O(1)}$ Ans

Ans 5.
```
int i=1,s=1;
while(s<=n){
    i++; s=s+i;
    printf("*");
}
```

Let the loop run for k times,

After 1st iteration: $S = S + 1$

After 2nd iteration: $S = S + 1 + 2$

It goes on for k times, as long as 's' is less than equal to 'n'

$$1+2+ \cdots + k <= n$$

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

$$\Rightarrow \left(\frac{k^2+k}{2}\right) <= n$$

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

$$\Rightarrow \boxed{k = O(\sqrt{n})}$$ Time Complexity.

Date
Page

Ans 6.
```
void function (int n){
    int i, count = 0;
    for(i=1; i*i<=n; i++){
        count++;
    }
}
```

$= i*i <= n$

$= i^2 <= n$ (taking root both sides)

$= i <= \sqrt{n}$

$$\sum_{i=1}^{\sqrt{n}} 1 = 1+1+1 - - - - + 1 \text{ (}\sqrt{n}\text{ times)}$$

$\boxed{T.C. = O(\sqrt{n})}$ - Ans

Ans7.
```
void function(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++;
            }
        }
    }
```

i	j	k
$n/2$	$\log_2 n$	$\log_2 n * \log_2 n$
$n/2+1$	$\log_2 n$	$\log_2 n * \log_2 n$
$n/2+2$	$\log_2 n$	⋮
$n/2+3$	⋮	⋮
⋮	⋮	⋮
n	$\log_2 n$	$\log_2 n * \log_2 n$

$$= O\left(\frac{n}{2} * \cancel{\log_2 n} * (\log_2 n * \log_2 n)\right)$$

$$= O(n(\log_2 n)^2) \text{ Ans}$$

Ans 8.
```
function (int n){
    if(n==1) return;          → O(1)
    for(i=1 to n){            → O(n)
        for(j=1 to n){        → O(n)
            printf("*");
        }
    } function(n-3);          → T(n-3)
}
```

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

put $n = n-3$

$T(n-3) = T(n-6) + (n-3)^2$

put $T(n-3)$ in $T(n)$

$T(n) = T(n-6) + (n-3)^2 + n^2$

$T(n) = T(n-3k) + (n-$

classmate
Date
Page

Ans 9.
```
void function(int n){
    for(i=1 to n){
        for(j=1; j<=n; j+=i){
            printf("*");
        }
    }
}
```

i	j
1	{1, 2, 3, 4 - - - . n}
2	{1, 3, 5, 7, 9, 11 - - -}
3	{1, 4, 7, 10, 13, - - - .}
4	{1, 5, 9, 13 — — — .}
⋮	⋮
n-2	{1, n-1}
n-1	{1, n}
n	{1}

$$\Rightarrow O\left(\sum_{i=1}^{n} 1 + \sum_{i=1\,(i=i+2)}^{n} 1 + \sum_{i=1\,(i=i+3)}^{n} 1 + \sum_{i=1\,(i=i+4)}^{n} 1 + \sum_{i=1\,(i=i+5)}^{n} 1 - - - + - - + - - + - - - - - - + \sum_{i=1\,(i=i+n)}^{n} 1\right)$$

~~$\Rightarrow O\left((n) + \left(\frac{n+1}{2}\right) + (n^2) +\right.$~~

$$\Rightarrow O\left(n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} - - - + 1\right)$$

$$\Rightarrow O\left(n\left(1 + \frac{1}{2} + \frac{1}{3} + - - + \frac{1}{n}\right)\right)$$

$$\Rightarrow \boxed{O(n \log n)}$$ Ans.