TUTORIAL - 1

M T W T F S S

Page No.:

Date:

YOUVA

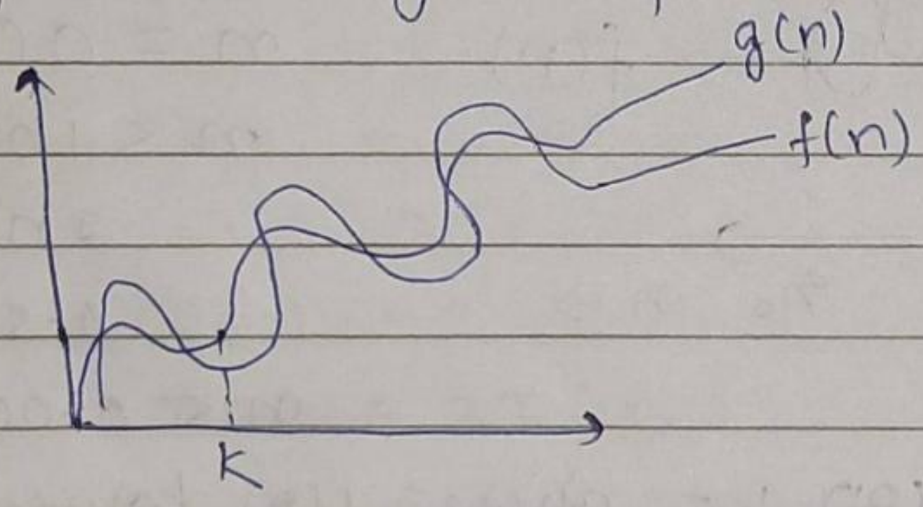
Q1: What do you understand by Asymptotic notations. Define different Asymptotic notation with examples.

Ans:- Asymptotic Notation :- are used to tell the complexity of an algorithm when the input is very large.

→ It describes the algorithm efficiency and the peryormance in a meaningful way. It ducribes the behaviour of time or space complexity for large instance characteristics.

The asymptotic notation of an algorithm is classified into 5 types:-

(i) Big - oh - notation (O):- (Asymptotic upper bound) The function $f(n) = O(g(n))$, iy and only iy there exist a the constant c and k such that $f(n) \leq c * g(n)$ for all n, $n \geq k$.



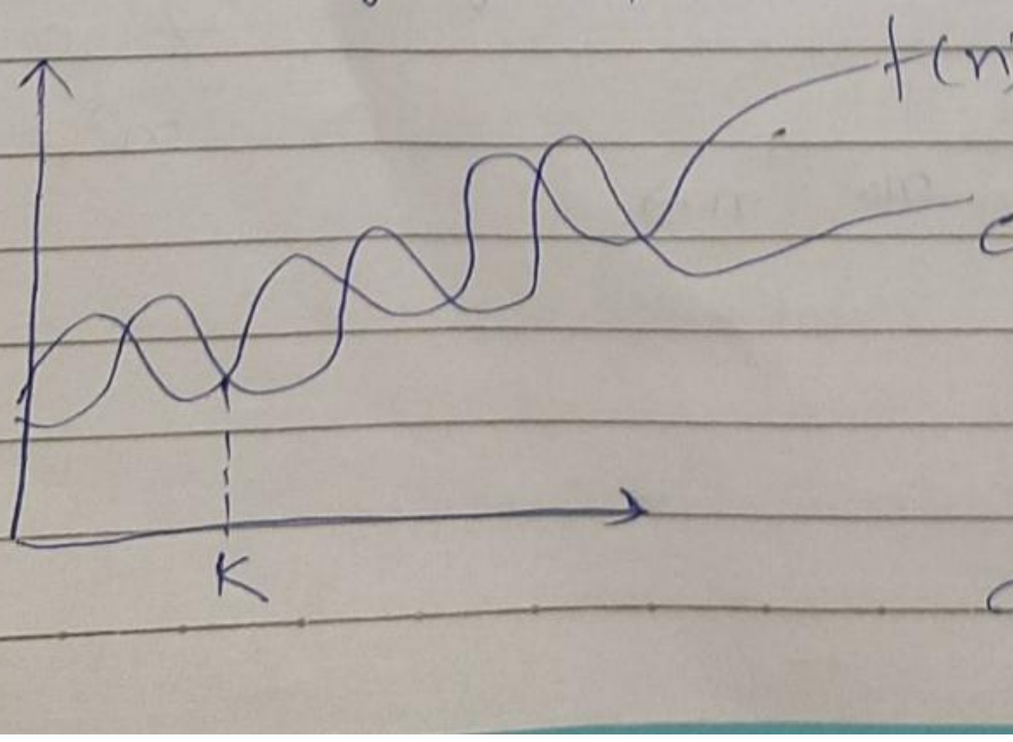
$f(n) = O(g(n))$

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

$\forall\ n \geq n_0$

some constant $c > 0$

(ii) Big - omega - notation (Ω):- (Asymptotic lower bound). The function $f(n) = \Omega(g(n))$, iff there exists the constant 'c and k such that $f(n) \geq c * g(n)$ for all, n, $n \geq k$

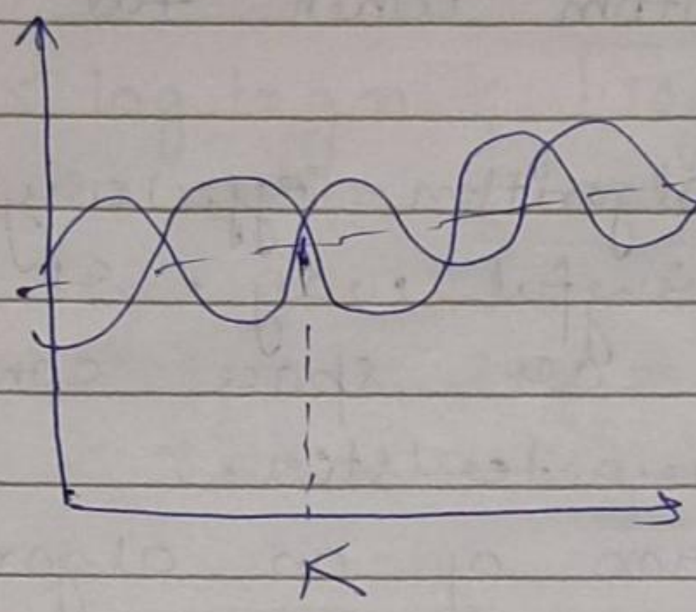


$f(n) = \Omega g(n)$

iff $f(n) \geq c \cdot g(n)$

$\forall\ n \geq n_0$ & some constants $c > 0$

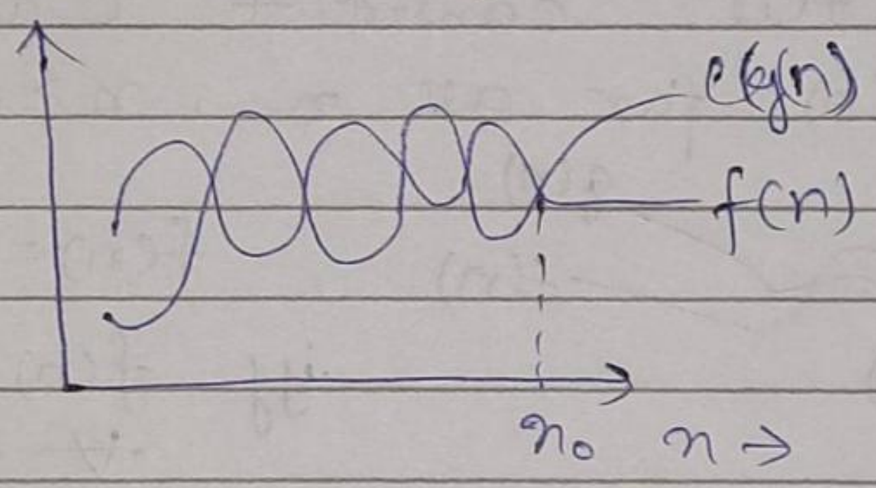
(iii) Big theta notation (Θ): (Asymptotic tight bound) The function $f(n) = \Theta(g(n))$, iff there exists a +ve constant c_1, c_2 & k such that $c_1 * g(n) \le f(n) \le 2 * g(n)$ for all $n, n \ge k$



$F(n) = \Theta(g(n))$
iff $c_1 g(n) \le f(n) \le c_2 g(n)$
$\forall\ n \ge \max(n_1, n_2)$

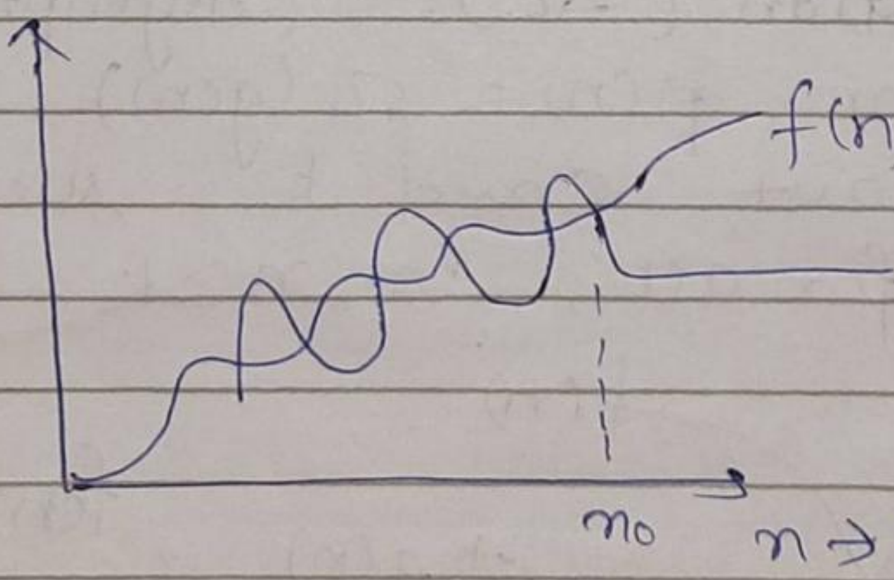
(iv) Small-oh (o) :- gives us upper bound

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



$f(n) < c\, g(n)$
$\forall\ n > n_0$ [$\forall\ c > 0$
$n = o(n^2)$
$n < 1n^2$
$2n^2$
$0.5n^2$
$n < 0.001 n^2 n_0$

(v) Small-omega (ω) :- gives us lower bound



$f(n) = \omega(g(n))$
$f(n) > c \cdot g(n)$
$\forall\ n > n_0$ &
$\forall\ c > 0$
$n^2 = \omega(n)$

Q 2:- What should be time complexity

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

Time complexity for a loop means no. of times loop has run.

→ loop will run for following values of i:-

i	1	2	4	8	16	....	2^k
value	2^0	2^1	2^4	2^3	2^4	....	n

i = 1, 2, 4, 8, 16, ... 2^k this means k times

i.e $2^k = n$

$$k \log_2 2 = \log_2 n$$

$$k = \log n$$

$$\boxed{TC = O(\log n)}$$

Q(3).. $$T(n) = \begin{cases} 3T(n-1) & , n > 0 \\ 1 & \end{cases}$$

By forward substitution

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

$T(0) = 1$

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

$T(2) = 3T(2-1) = 3 * 3 = 3^2$

$T(3) = 3T(3-1) = 3 * 3^2 = 3^3$

$\vdots$

$T(n) = 3^n$

$$\boxed{TC = O(3^n)}$$

Q 4 :- $$T(n) = \begin{cases} 2T(n-1) - 1 & , n > 0 \\ 1 & \end{cases}$$

By forward substitution

$T(0) = 1$

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

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

$= 2T(1) - 1$

$= 2(2-1) - 1$

$= 2^2 - 2^1 - 1$

$T(3) = 2T(3-1) - 1$

$= 2T(2) - 1$

$= 2(2^2 - 2^1 - 1) - 1 = 2^3 - 2^2 - 2^1 - 1$

$\vdots$

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} \cdots 2^2 - 2^1 - 2^0$

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

$TC = 1$

Q(5) What should be time complexity.

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

$s_i = s_{i-1} + i$

The value of 'i' increases by one for each loop value contained in 's' at the ith iteration is the sum of the first 'i' +ve integers. If k is the total no. of iterations taken by any program then while loop terminates if: $1+2+3+\cdots+k$.

$= [k(k+1)/2] > n$

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

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

Q(6)

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

$TC = O(n)$

Q7 :- void function (int n)
{
int i, j, k, count = 0;
for (i = n/2 ; i <= n ; i++) n
{ for (j = 1 ; j <= n ; j = j * 2) log n
{ for (k = 1 ; k <= n ; k = k * 2) log n
{ count ++ ;
}
}
}
}

n * log n * log n

TC = O(n log² n)

Q8 :- function (int n)
{ if (n == 1)
return ;
for (i = 1 to n) n
{ for (j = 1 to n) n
printf ("*");
}
function (n-3);
}

TC = O(n²)

Q9 :- void function (int n)
{
for (i = 1 to n) n
{
for (j = 1 ; j <= n ; j = j + 1) n
printf ("*");
}
}

TC = O(n²)

Q.10. For the functions, n^k and c^n, what is the asymptotic relationship between these functions?
Assume that $k >= 1$ and $c > 1$ are constants. Find out the value of c and n_0 for which relation holds.

Ans:- n^k is $O(c^n)$

$$\boxed{n^k = O(c^n)} \text{ — relationship}$$

As given n^k and c^n

$$n^k \leq a(c^n)$$

$\forall\ n \geq n_0$ & konstant $a > 0$

for $n_0 = 1$; $c = 2$

$\Rightarrow \quad 1^k < a^2$

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