Assignment

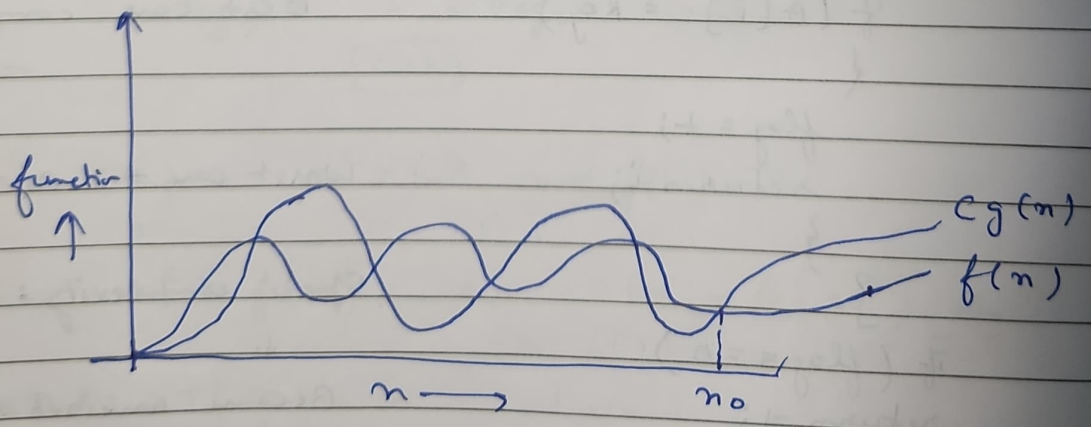
Q1: What do you understand by asymptotic notation. Define different asymptotic notation with examples.

A Asymptotic notation is used to describe the running time of an algorithm - how much time an algorithm takes with a give input n. There are ~~three~~ five different notation : big $O_n(g)$ big Theta (Θ) and big Omega (Ω), Small $O_n(o)$, Small Omega (ω)

There are five different notations:-

① Big $O_n(O)$:- The Big $O_n(O)$ notation describes the worst case running time of a program, we compute the big-O of an algorithm by counting how many iterations an algorithm will take in the worst-case scenario with an input of N.

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



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

Ex:- $n^2 + 5n + 6$

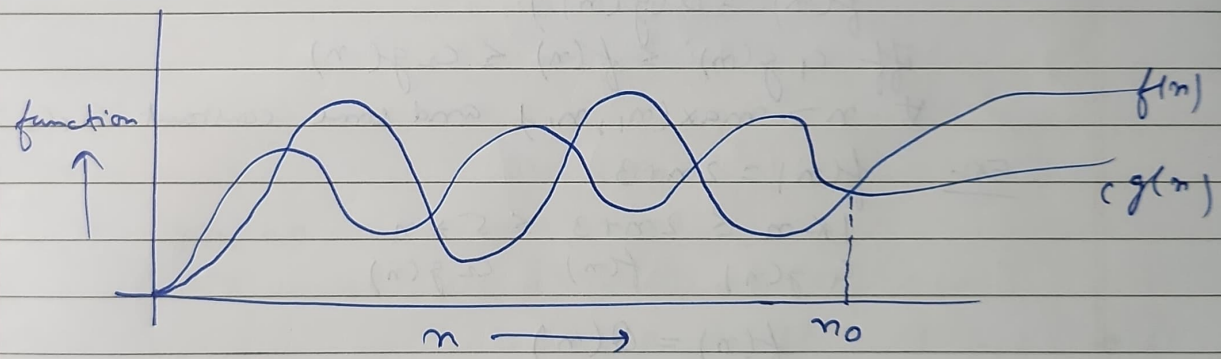
$n^2 + 5n$

$n^2 + n$

$O(n^2)$ ∴

② Big Omega (Ω) :- Big omega (Ω) describes the best running time of a program. we compute the big Ω by counting how many iterations an algorithm will take in the best-case scenario based on an input of N.

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



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

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

$\forall$ $n \geq n_0$, and some constant $c > 0$

eg :- $f(n) = 2n + 3$

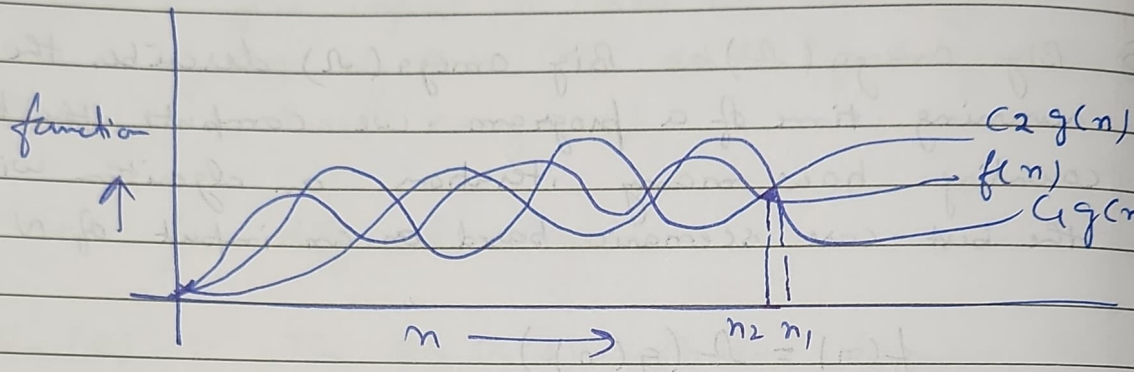
$2n + 3 \geq 1 \times n \quad \forall \; n \geq 1$

$f(n) \quad c \quad g(n)$

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

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

③ Big Theta (θ) :- It represents both the upper bound and lower bound of a function. It is used to describe the tight bound of an algo. It provides an exact bound on the growth rate of the function. It tells exact time.

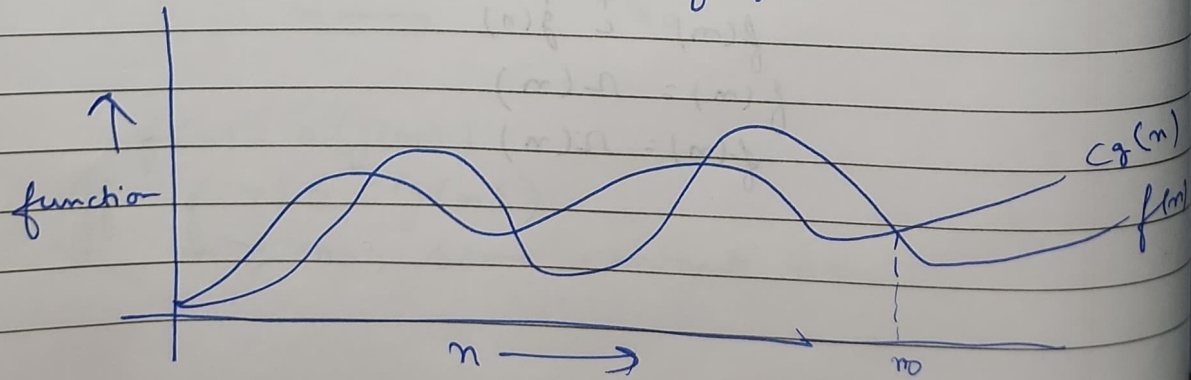


$f(n) = \theta(g(n))$
iff $c_1 g(n) \leq f(n) \leq c_2 g(n)$
$\forall\ n > \max(n_1, n_2)$ and some constant $c_1, c_2 > 0$

Ex:- $f(n) = 2n+3$
$1 * n \leq 2n+3 \leq 5 * n$
$c_1 g(n)$ $f(n)$ $c_2 g(n)$

$f(n) = \theta(n)$
$f(n) = \theta(n^2)$

④ Small Oh (o) :- $f(n) = o(g(n))$
iff $f(n) < c g(n)$
$\forall\ n > n_0$
$g(n)$ is upper bound of $f(n)$ $\forall\ c > 0$



Ex:- $8x^2 \neq O(x^2)$ $(x \to \infty)$

$8x^2 = O(x^3)$ $(x \to \infty)$

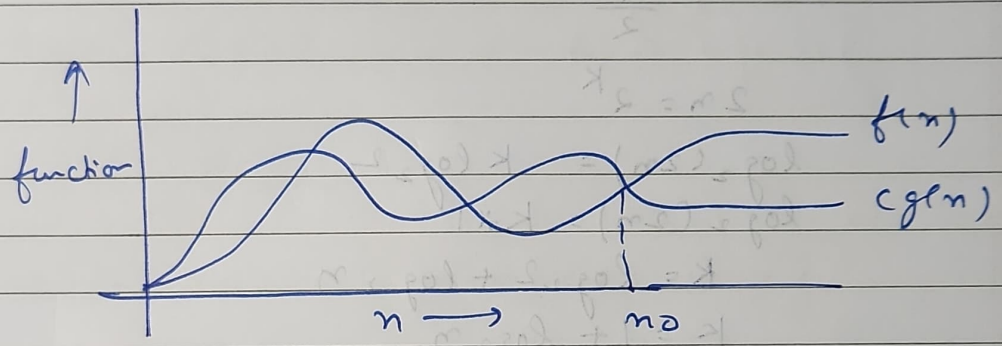
$\lim_{x \to \infty} \frac{f(n)}{g(n)} = \infty$

(5) Small Omega (ω) :- $f(n) = \omega(g(n))$

iff $f(n) > c\,g(n)$

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

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



Ex:- Lower limit

$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$

$3n = \omega(n)$

$f(n) = 3n$

$g(n) = 1$

$\lim_{n \to \infty} \frac{g(n)}{f(n)} = \lim_{n \to \infty} \frac{1}{3n} = \frac{1}{\infty} = 0$ ☐

Q29. What should be time complexity of - for(i=1 to n) {i=i*2;}

$i = \underbrace{1, 2, 4, 8, \ldots, n}_{k \text{ terms}}$

$t_k = a r^{k-1}$

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

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

$2n = 2^k$

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

$\log_2(2n) = k * 1$

$k = \log_2 2 + \log_2 n$

$k = 1 + \log_2 n$

$O(\log_2 n)$ Ans

Q31. $T(n) = \{ 3T(n-1)$ if $n>0$, otherwise $1 \}$

Given $T(0) = 1$

Put $n = 1$

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

$T(1) = 3T(0)$ - (i)

Put $T(0)$ in eq (i)

$T(1) = 3 \times 1$

$T(1) = 3$

Put $n = 2$

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

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

Put $T(1)$ in eq (ii)

$T(2) = 3 \times 3$

Put $n=3$

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

$T(3) = 3T(2)$ —(iii)

Put $T(2)$ in eq (iii)

$T(3) = 3 \times 3 \times 3$

$T(n) = 3 \times 3 \times 3 \times \cdots \times n$ times

$T(n) = 3^n$.

$O(3^n)$ A.

Q42. $T(n) = \{2T(n-1) - 1$ if $n > 0$, otherwise $1\}$

$T(n) = 2T(n-1) - 1$ Given $T(0) = 1$

Put $n=1$

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

$T(1) = 2T(0) - 1$

$= 2 \times 1 - 1$

$T(1) = 1$

Put $n=2$

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

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

$= 2 \times 1 - 1$

$= 1$

Put $n=3$

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

$= 2T(2) - 1$

$= 2 \times 1 - 1$

$= 1$

$T(n) = 1$

$O(1)$ A

Q5. What should be time complexity of

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

s	i
1	1
1+2	2
1+2+3	3
1+2+3+4	4
1	1
1	1
1	1
1+2+3+4+---k	k

$\frac{k(k+1)}{2}$

$\frac{k^2+k}{2} > n$ (Loop terminate)

$k^2 > n$

$k = \sqrt{n}$

$O(\sqrt{n})$ A

Q6:- Time complexity of:-

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

i	i^2
1	1
2	4
3	9
4	16
.	.
.	.
k	.

Assume $i^2 > n$ (to terminate the loop)

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

$i^2 = \frac{k^2+k}{2}$

$i^2 = k^2$

$k^2 > n$

$k > \sqrt{n}$

$O(\sqrt{n})$ Ans

Q7:- Time complexity of -

```
void function (int n) {
  int i, j, k, count = 0;
  for (i = n/2 ; i <= n ; i++)          ---- O(n)
    for (j=1; j<=n; j=j*2)              ---- O(log2 n)
      for (k=1) ; k<=n; k=k*2)          ---- O(log2 n)
        count ++;
}
```

$O(n \log_2^2 n)$ Ans

Q9:- Time complexity of

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

Outer loop will run n times

for (i=1), j will run n times

for (i=2), j will run n/2 times

for (i=n), j will run n/n time

Inner loop will run $= (n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n})$

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

$= n \log n$

$O(n \log n)$ Ans

Q8:- Time complexity of

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

For function (n-3)

$n, n-3, n-6, \ldots$ k^{th} term

$n, n-1\times3, n-2\times3, \ldots, n-k\times3$

k^{th} term $= n+(-k-1)\times3$

$= n-3k-3$

$k = \frac{n-4}{3}$

$\left(\frac{n-4}{3}\right) n^2 = \frac{n^3-4n^2}{3}$

$O(n^3)$ Ans