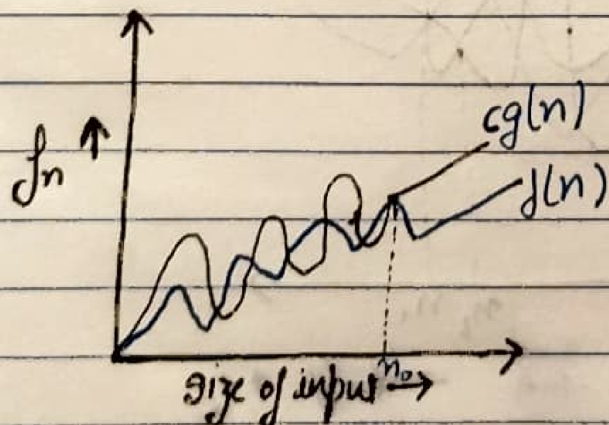
Que 1 Asymptotic Notations

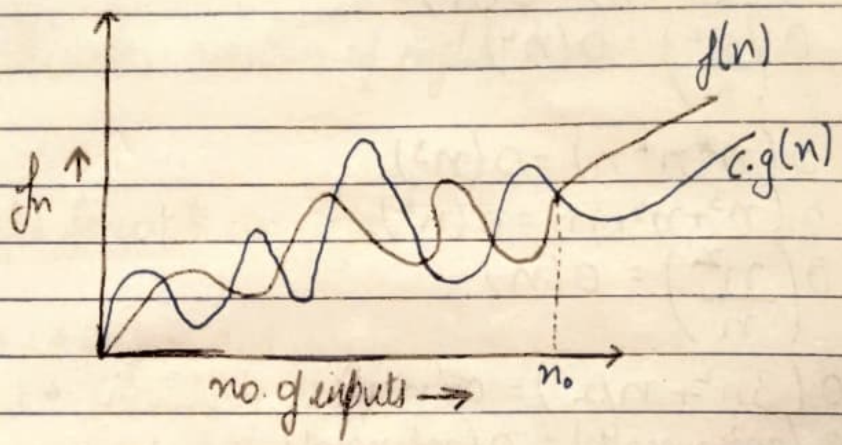
Means Tending to infinity
They help you find the complexity of an algorithm when input is very large.

1. Big O(o)



$f(n) = O(g(n))$
iff $f(n) \le c(g(n))$
$\forall n \geq n_0$
for some constant $c > 0$
$\Rightarrow$ $g(n)$ is tight upper bound of $f(n)$

2. Big Omega (Ω)



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

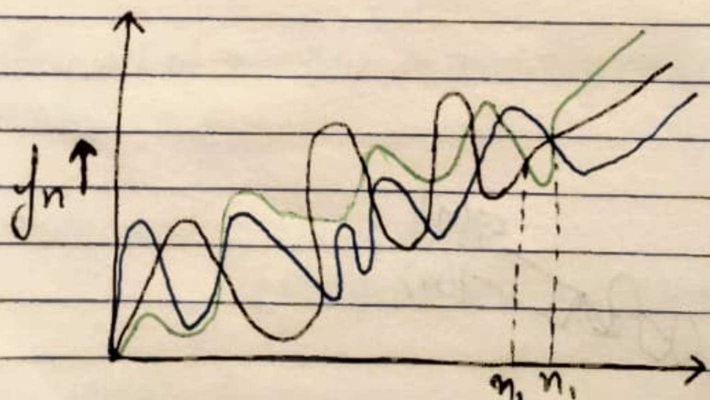
$g(n)$ is tight lower bound of $f(n)$

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

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

$\forall$ $n \geq n_0$ for some constant $c > 0$.

3. Theta (θ)



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

$g(n)$ is both 'tight' upper & lower bound of function $f(n)$

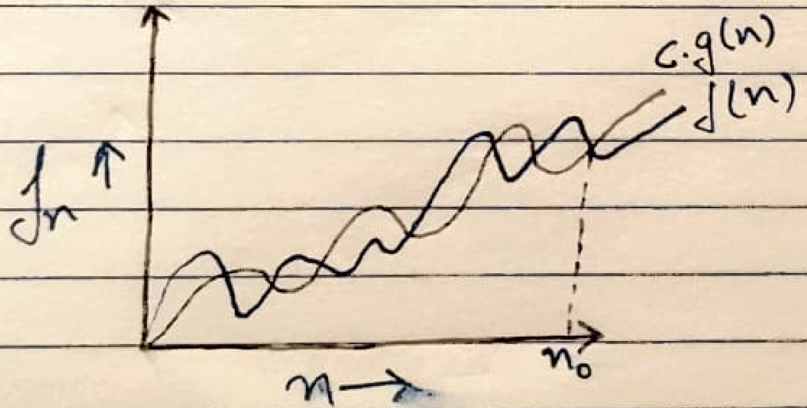
$f(n) = \theta(g(n))$

iff $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

$\forall$ $n \geq \max(n_1, n_2)$

for some constant $c_1 > 0$ & $c_2 > 0$

4. Small o(o)



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

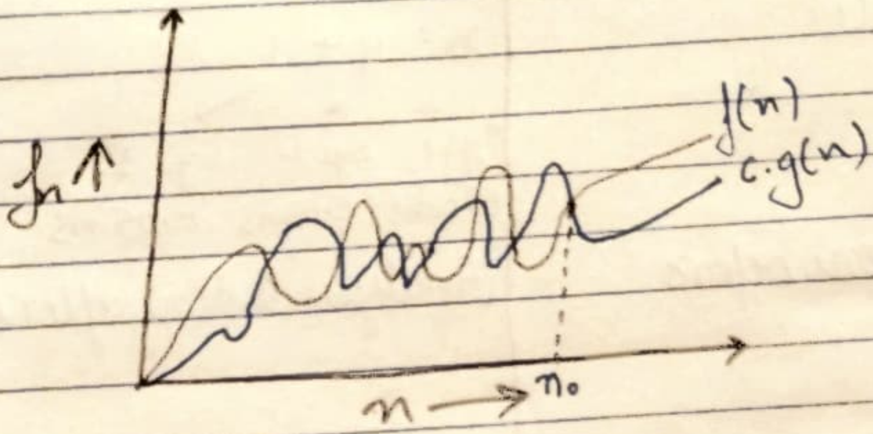
g(n) is upper bound of fn f(n)

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

when $f(n) < c.g(n)$

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

5. Small omega (ω)



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

$g(n)$ is ~~upper bound of f~~ lower bound of fn $f(n)$

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

when $f(n) > c.g(n)$

$\forall n > n_0$

& $\forall c > 0$

Date.
Page No.

Que 2 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)

$\Rightarrow \sum_{i=1}^{n} 1+2+4+8+.....n$

G.P k^{th} value $\Rightarrow T_k = ar^{k-1}$

$\Rightarrow 1\times 2^{k-1}$

$\Rightarrow n = 2^{k}$

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

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

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

$\Rightarrow \log n + 1 = k.$

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

$= O(\log n)$

Date.
Page No.

Que 3 $T(n) = \{3T(n-1)$ 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

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

$9T(n-2)$ — 3

putting $n = n-2$ in — 1

$T(n) = 3(T(n-3))$ — 4

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

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

putting $n-k=0$

$n=k$

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

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

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

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

Que 4 $T(n) = \{2T(n-1)-1$ if $n>0$, otherwise $1\}$

$T(n) = 2T(n-1) - 1$ — 1

Let $n = n-1$

$T(n-1) = 2T(n-2) - 1$ — 2

from 1 & 2

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

$T(n) = 4T(n-2) - 2 - 1$ — 3

Let $n = n-2$

Date.
Page No.

$T(n-2) = 2T(n-3) - 1$ —— (4)

from (3) & (4)

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

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

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

$G.P = 2^{k-1} + 2^{k-2} + 2^{k-3} + \dots 1$

$a = 2^{k-1}$

$r = 1/2$

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

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

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

$= 2^k - 1$

Let $n-k = 0$

$n = k.$

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

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

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

$T(n) = O(1)$

Que 5 What should be time complexity of.

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

Date.
Page No.

Sum of $S = 1 + 3 + 6 + 10 + \ldots\ldots T_n$ — (1)
also $S = 1 + 3 + 6 + 10 + \ldots\ldots T_{n-1} + T_n$ — (2).
from (1) − (2)

$0 = 1 + 2 + 3 + 4 + \ldots\ldots n - T_n$
$T_k = 1 + 2 + 3 + 4 + \ldots\ldots k$
$T_k = \frac{1}{2} k(k+1)$

for K iterations
$1 + 2 + 3 + \ldots\ldots + k <= n$
$\frac{k(k+1)}{2} <= n$
$\frac{k^2 + k}{2} <= n$
$O(k^2) <= n$
$k = O(\sqrt{n})$
$T(n) = O(\sqrt{n})$

Que 6 Time complexity of -

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

as $i^2 <= n$
$i <= \sqrt{n}$
$i = 1, 2, 3, 4 \ldots\ldots, \sqrt{n}$
$\sum_{i=1}^{n} 1 + 2 + 3 + 4 + \ldots\ldots + \sqrt{n}$

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

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

$T(n) = O(n)$

Que 7 Time complexity of.

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

for $k = k*2$

$k = 1, 2, 4, 8, \ldots n$

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

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

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

$n \Rightarrow 2^k$

$\log n = k$

i	j	k
1	log n	log n * log n
2	log n	log n * log n.
⋮		⋮
n	log n	log n * log n

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

$O(n \log^2 n)$

Date.
Page No.

Que 8 Time complexity of

```
function (int n)
{
  int (n==1)
  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)
}
```

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

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

$c = \log_3 1 = 0$

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

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

Que 9 Time complexity of -

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

for $i=1 \Rightarrow j=1,2,3,4,\ldots\ldots n = n$
for $i=2 \Rightarrow j=1,3,5\ldots\ldots n = n/2$.
for $i=3 \Rightarrow j=1,4,7,\ldots\ldots n = n/3$.

for $i=n$

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

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

$$\sum_{j=n}^{1} n(\log n)$$

$T(n) = [n\log n]$.
$T(n) = O(n\log n)$.

Que 10 for functions, n^k & c^n, what is the asymptotic relation between these functions?
assume that $k>=1$ & $c>1$ are constant.
Find out the value of c & no for which relation holds.

as given n^k & c^n
relation b/w n^k & c^n is

$n^k = O(c^n)$
as $n^k \leq a c^n$
$\forall n \geq n_0$ & some constant $a>0$.
for $n_0 = 1$
$c = 2$

Date.

Page No.

$1^{k} \leq a_{2}^{1}$

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