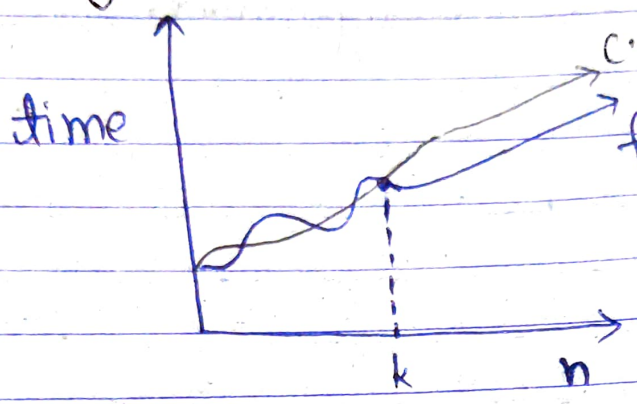
Asymptotic Notation : It's a notation to calculate how much time an algo takes to run over n input.

① Big - oh (O)



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

$f(n) \leq c \cdot g(n)$ [in small o $f(n) < c \cdot g(n)$]

$\forall\ c > 0$

$n \geq k$

$k \geq 0$

tight upper bound.
/least (means the least value for which the function hold true)

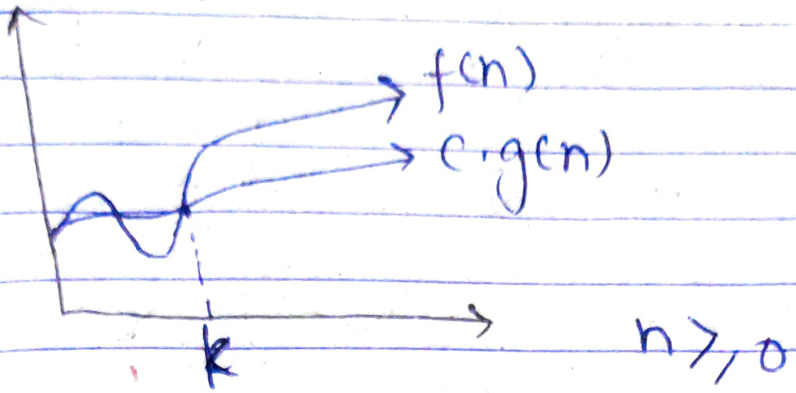
eg ⇒ $f(n) = 2n^2 + 3n$

Alg to def. $f(n) \leq c \cdot g(n)$

$2n^2 + 3n \leq c \cdot g(n)$

Ans $2n^2 + 3n \leq c \cdot g(n^2)$

2) Big - Omega (Ω)



greatest lower bound.

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

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

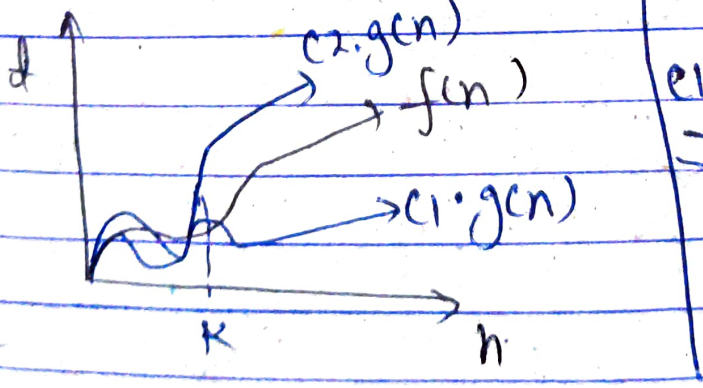
(in small omega $f(n) > c \cdot g(n)$.)

eg $f(n) = 2n^2 + 4n$

$2n^2 + 4n \geq c \cdot g(n)$

$2n^2 + 4n \geq c \cdot n^2$.

3) Theta Notation



$c1 \cdot g(n) \leq f(n) \leq c2 \cdot g(n)$

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

$f(n) \geq c1 g(n)$ and $f(n) \leq c2 \cdot g(n)$

Date ___/___/_____ Page No.: _____

② What should be time complexity of :-

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

$2^k = n$

Taking log

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

$k = \log_2 n$

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

③ $T(n) = 3T(n-1)$ if $n>0$; otherwise 1. — ①

for $n = n-1$

$T(n-1) = 3T(n-2)$ — ②

from ① and ②

$T(n) = 3^2 T(n-2)$ — ③

from ② put $n = n-1$

$T(n-2) = 3T(n-3)$ — ④

from ③ and ④

$T(n) = 3^3 T(n-3)$ — ⑤

Generalising $T(n)$

$T(n) = 3^k T(n-k)$ — ⑥

put $n-k=1$

"No real change in history has ever been acheived by discussions." —Subhash chandra bose

Date ___/___/_____ Page No.: _____

$k = n-1$

put $k = n-1$ in (6)

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

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

from (1)

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

put $n = 1$

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

Alq

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

$T(1) = 3$

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

$T(n) = 3^n$

$T(n) = \Theta(3^n)$

Diy

Date ___/___/_____ Page No.: _____

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

for $n = n-1$

$T(n-1) = 2T(n-2) - 1$ — (2)

from (1) and (2)

$T(n) = 2^2 T(n-2) - 1 - 2$ — (3)

from (2) put $n = n-1$

$T(n-2) = 2T(n-3) - 1$ — (4)

from (3) and (4)

$T(n) = 2^3 T(n-3) - 1 - 2 - 4$ — (5)

generalising the term of eq (5)

$T(n) = 2^k T(n-k) - 1 - 2 - 4 \ldots - 2^k$ — (6)

put

$n-k = 1$

$k = n-1$

put $k = n-1$ in eq (6)

$T(n) = 2^{n-1} T(1) - 1 \left[\frac{1 - 2^{n-1}}{1-(1-1)} \right]$

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

$T(n) = O(1)$ ∴ $O(1)$

Ajay

Chitra

"No real change in history has ever been acheived by discussions." —Subhash chandra bose

Date ___/___/___ Page No.: ___

(5) Time complexity of:-

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

i	s
1	1
2	3
3	6
4	10
5	15
.	.
k	$\frac{k(k+1)}{2}$ for k iteration

loop terminates when $\frac{k(k+1)}{2} > n$

Ajay

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

(6) Time complexity of -

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

Ajay

Chitra

"To manifest the divinity is the right of everyone." –Sri Ramakrishna Paramahamsa

Date ___/___/_____ Page No.: _____

i	count
1	0
2	1
3	2
⋮	⋮
k	(k-1)

since $k*(k-1) \leq n$; ignoring constant and lower order terms.

$\therefore k \leq \sqrt{n}$

$\therefore O(\sqrt{n})$.

7→ Time complexity of -

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

Ideally

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

"No real change in history has ever been acheived by discussions." —Subhash chandra bose

Chitra

Page No.: _____

Date ___/___/_____

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

$O(n * (\log n)^2)$

(8) Time complexity of :-

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

$a = n-3 \qquad d = -3$

$1 = (n-3) + (k-1)(-3) \qquad (\because a_n = a + (n-1)d)$

$1 = (n-3) + (-3k) + 3$

$3k = n-1$

$k = \frac{n-1}{3}$

T.C. $= O(n * n^2)$

$= O(n^3)$.

Date ___/___/___ Page No.: _____

9→

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

i = 1 j = n times

i = 2 j = n/2 times

⋮

i = k j = n/k times

∴ i = n & j = n/n times

So Time Complexity = $(n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n})$

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

$n \log n$

So $T_c = \Theta(n \log n)$.

Ajay

Date ___/___/_____ Page No.: _____

10. $f(n) = n^k$ $\quad g(n) = c^n$

where $k=1$, $c=2$

$\Rightarrow$ $f(1) = (1)^1$

and $g(1) = 2^1$

$\because f(1) < g(1)$

$\Rightarrow$ $f(2) = (2)^1$ and $g(2) = 2^2 = 4$

$f(2) < g(2)$

it satisfies O notation

ie $f(n) \leq g(n)$

So $f(n_0) \leq c_0 \cdot g(n_0)$

$n_0^k = c_0 * c^{n_0}$

$k=1$, $c=2$

$n_0^1 = c_0 * 2^{n_0}$

$\left[\frac{n_0}{c_0}\right]^1 = (2)^{n_0}$

comparing $n_0 = 1$ and $\frac{n_0}{c_0} = 2$

$c_0 = \frac{1}{2}$

$\therefore f(n) \leq 0.5(g(n))$

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

"No real change in history has ever been acheived by discussions." –Subhash chandra bose

Chitra