Tutorial 1

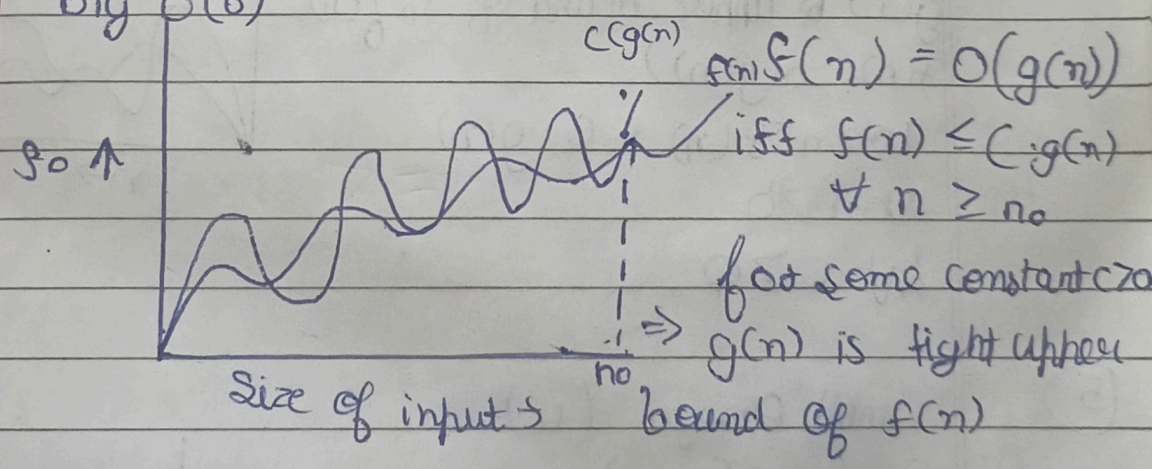
Name -> Vineet Joshi
Section -> CST
Semester -> 4th
Class Roll Number -> 36
University Roll Number -> 2017581
Date -> 10-03-2022.

Name -> Vineet

(I) Asymptotic Notations

Asymptotic Notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

(a) Big O(o)

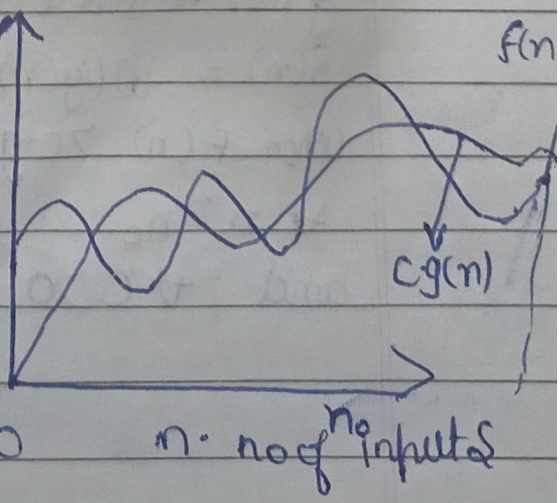


(II) Big Omega (Ω)

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

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

$f(n) = \Omega(g(n))$ iff $f(n) \geq g(n)$ $\forall n \geq n_0$ for some $c > 0$



iii) Theta (Θ)

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

$g(n)$ is both tight Upper and lower bound of f_n

$f(n)$

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

iff

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

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

for some constant $c_1 > 0$

and $c_2 > 0$.

Name → Vineet

④ Small o(0)

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

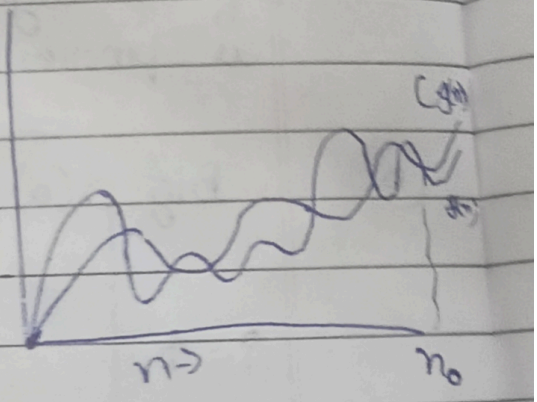
$g(n)$ is Upper bound of f_n $f(n)$

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

when $f(n) < c \cdot g(n)$

$\forall\ n > 0$

$\&\forall\ c > 0$



⑤ Small Omega (ω)

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

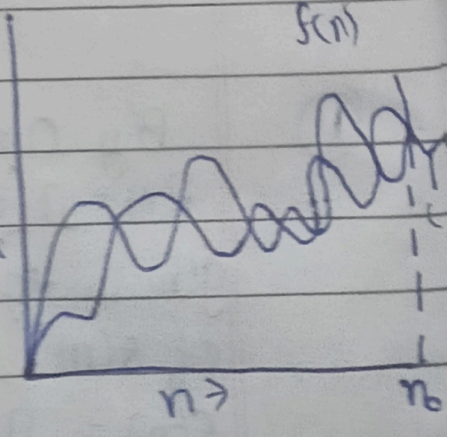
$g(n)$ is lower bound of f_n $f(n)$

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

when $f(n) > c \cdot g(n)$

$\forall\ n > n_0$

and $\forall\ c > 0$



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)

⇒ $\sum_{i=1}^{n}$ 1+2+4+8+ .... + n.

GP the value $T_K = a r^{K-1}$
$= 1 \times 2^{K-1}$
⇒ $\boxed{n = 2^{K-1}}$

⇒ $2^K = 2n$
⇒ $\log 2n = K \log 2$
⇒ $\log_2 + \log n = K \log 2$.
⇒ $\log n + 1 = K$

Name → Vineet

⇒ $O(K) = O(\log n + 1) = O(\log n)$.

Q3 $T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0, \\ 1 & \text{otherwise.} \end{cases}$

$T(n) = 3T(n-1)$ —— ①
put $n = n-1$
$T(n-1) = 3T(n-2)$ —— ②
from ① and ②

⇒ $T(n) = 3(3T(n-2))$
$= 9T(n-2)$ —— ③

Putting $n = n-2$ in (1)

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

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

$\Rightarrow T(n) = 3^K (T(n-K))$

Putting $n-K=0 \Rightarrow \boxed{n=K}$

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

$= 3^n T(0)$

$= 3^n \times 1 = O(3^n)$

Name → Vineet

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

Using backward substitution.

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

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

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

$\vdots$

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

$T(0) = 1$

Substituting value of $T(n-1)$ then $T(n-2)$ ... till $T(1)$ in eqn $T(n)$ we get

$T(n) = 2^n \times T(0)$

$= 2^n \times 1 = O(2^n)$

⑤

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

i = 1 2 3 4 5 6 . . . . . .

s = 1+3+6+10+15 . . . . . .

Sum of s = $1+3+6+10+\cdots+n$ — ①

also s = $1+3+6+10+\cdots+(m+n)$ — ②

from ① − ②

$0 = 1+2+3+4+\cdots+n-(m+n)$

$\Rightarrow T_K = 1+2+3+4+\cdots+K$

$T_K = \frac{1}{2}K(K+1)$

Name → Vineet

for k iterations.

$1+2+3+\cdots K <= n$

$\frac{K(K+1)}{2} <= n$

$\frac{K^2+K}{2} <= n$

$O(K^2) <= n$

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

$\Rightarrow T(n) = O(\sqrt{n})$

(7) Time complexity of

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

for (i=1 ; j <= n ; j = j*2)
  for (k=1; k <= n; k = k*2)
     count++;
```

Name ⇒ Vineet

for $k = k*2$

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

GP ⇒ $a = 1$, $r = 2$

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

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

$n \Rightarrow 2^k$

$= \boxed{k = \log n}$

i	j	k
1	logn	logn * logn
2	logn	logn * logn
.	:	
.	:	
.	:	
n	logn	logn * logn

⇒ $O(n * \log n * \log n)$

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

(8) $T(n) = T\left(\frac{n}{3}\right) + n^2$

Using Master's Method

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

$c = \log_3 1 = 0$

$n^0 = 1 < f(n) \quad [f(n) = n^2]$

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

(10) As given n^k and c^n Name > Vineet

Relation b/w n^k and c^n is

$n^k = O(c^n)$

As $n^k \leq ac^n$

$\forall\ n \geq n_0$ and some constant $a > 0$

for $n_0 = 1 \quad c = 2$

$\Rightarrow$ $1^k \leq a_2$

$\Rightarrow$ $\boxed{n_0 = 1}$ and $\boxed{c = 2}$