Divyansh Sharma
CST CORE - Sem IV
Roll no 46.

DESIGN & ANALYSIS OF ALGORITHMS

TUTORIAL - 1.

Q1). ASYMPTOTIC NOTATION: Tends to infinity.

These notations help us to find complexity of an algorithm when input is very large.

Different notations :

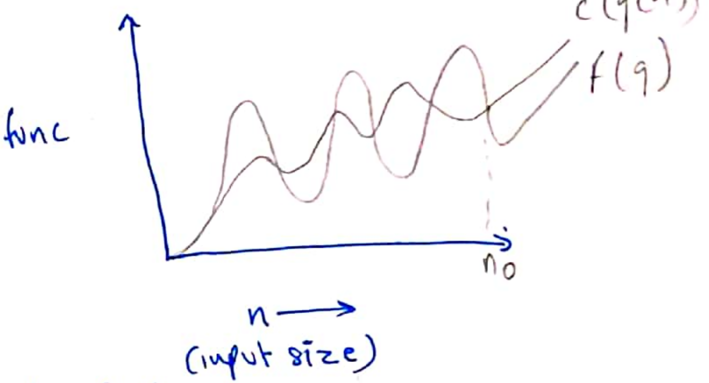
•) Big O (O)

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

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

$\forall \quad n \ge n_0$

for some constant, $c > 0$.



$g(n) =$ "tight" upper bound of $f(n)$

•) Big Omega (Ω)

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

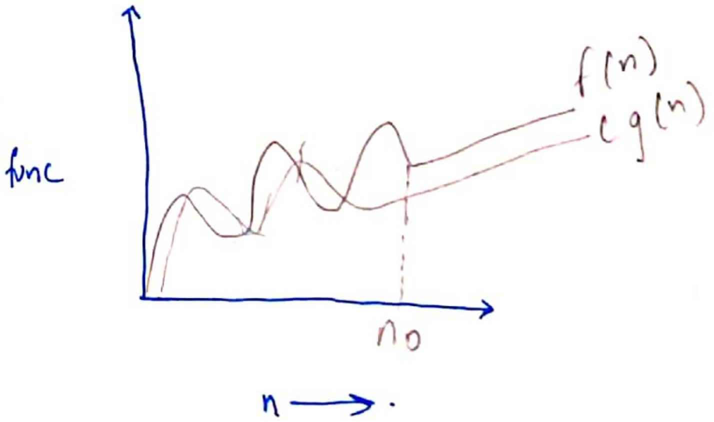
$g(n) =$ "tight" lower bound of $f(n)$

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

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

$\forall \; n \ge n_0$

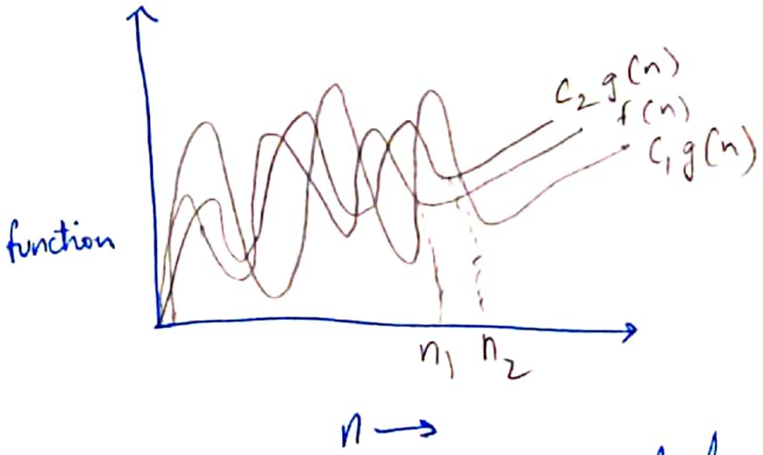
for some constant, $c > 0$



•) Theta (Θ)

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

$g(n) =$ both "tight" upper & lower bound of function $f(n)$



①

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

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

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

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

•)- Small o (θ):

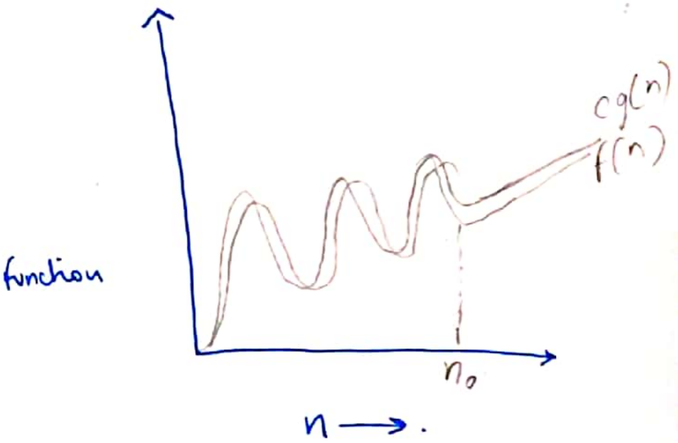
$f(n) = \theta g(n)$

$g(n)$ = upper bound of function $f(n)$

$\therefore f(n) = \theta g(n)$ when $f(n) < g(n)$

$\forall n > n_0$

and $\forall$ constraints , $c > 0$



•) Small omega (ω).

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

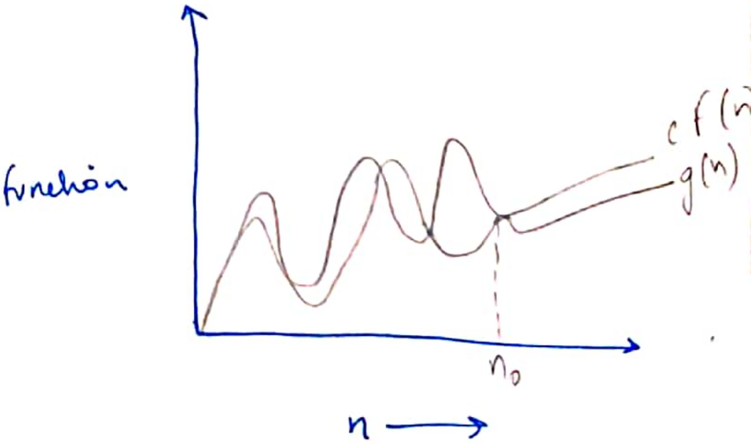
$g(n)$ is lower bound of function $f(n)$

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

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

$\forall n > n_0$

and $\forall$ constants , $c > 0$.



Q2 T.C=? for = for (i=1 to n) { 1=i*2;}.

Ans. now: for (i=1 to n) — i=1, 2, 4, 8 ......n

$1 = i * 2$ —— O(1).

Thus $\sum_{i=1}^{n}$ $1+2+4+8+ \ldots n$

now, GP with k^{th} value : ar^{k-1}

$a = 1$ & $r = 2$ ∴ $1 \times 2^{k-1}$

$\Rightarrow 2n = 2^{k-1}$ → $2n = \frac{2^k}{2}$

→ $2n = 2^{k}$.

$\log 2n = k \log 2$ → $\log 2 + \log n = k \log 2$

→ $\log n + 1 = k$

Thus $O(k) = O(1 + \log n) = \boxed{O(\log n)}$.

Q3 $T(n) = \{3T(n-1)$, if $n > 0$ otherwise $1\}$.

Time Complexity =?

Ans). $T(n) = 3T(n-1)$ — (i).

$n = n-1$

∴ $T(n-1) = 3T(n-2)$ — (ii).

subs in (i) $T(n) = 3 \times 3T(n-2)$

$= 9T(n-2)$ — (iii).

putting $n = n-2$ in (i) : $T(n-2) = 3T(n-3)$ — (iv)

subs in (iii) : $T(n) = 27T(n-3)$ — (v).

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

$T(0) = 1$

③

putting $n-k=0$
$n=k$

$$\therefore \quad T(n) = 3^n (T(n-n))$$
$$= 3^n T(0)$$
$$= 3^n.$$

$$\therefore \boxed{T(n) = O(3^n)} = \text{ans.}$$

Q4) T.C=? For $T(n) = \{ 2T(n-1)-1$ if $n>0$, otherwise $1 \}$

$$\Rightarrow T(n) \begin{cases} 2T(n-1)-1 , & n>0 \\ 1 , & \text{otherwise} \end{cases}$$

Ans). $T(n) = 2T(n-1)-1$ — ①.

$n=n-1 \Rightarrow T(n-1) = 2T(n-2)-1$ — (ii).

sub in eq ① $T(n) = 2[2T(n-2)-1]-1$
$$= 4T(n-2)-2-1$$
$$= 2^2 T(n-2)-3 \quad \text{— (iii)}.$$

Similarly: $T(n) = 2^2 (2T(n-3)-1)-3$
$$= 8T(n-3)-7 \quad \text{— (IV)}$$

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

let $n-k=1$ ($\because$ $n-k$ has to be 1 as it is decreasing one by one)

$$T(n) = 2^k T(1) - (2^k-1)$$
$$= 2^n (T(1)-1)+1$$
$$= O(2^n) = \boxed{O(2^n)}.$$

Q5).

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

Time complexities = ?

Ans. initially $i=1 \quad s=1$

$$\sum_{i=1}^{n} = 1+2+3+4+5 \ldots\ldots$$

$$\sum_{s=1}^{n} = 1+3+6+10+\ldots\ldots$$

series = sum of natural nos

$$\Rightarrow \; k\frac{(k+1)}{2} \geq n.$$

$$O(k^2) \geq n. \quad \Rightarrow \boxed{k = O(\sqrt{n})} = \text{Ans.}$$

Q6). Time complexity = ?

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

Ans). initially

$n=1, \; i=1$

$n=2 \quad i=1$

$n=4 \quad i=2$

$n=9 \quad i=3$

$\therefore n=n \quad i=\sqrt{n}$

$$\therefore \sum_{i=1}^{n} 1+1+2+\ldots\ldots\ldots \sqrt{n} \text{ times}$$

$$\therefore \boxed{O(\sqrt{n})} \Rightarrow \boxed{T(n) = O(\sqrt{n})} \text{ ans.}$$

Q7). Time complexity = ?

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

Ans). for $n=2 \quad i=2$ [initially]

$n=16 \quad i=9$

$\therefore$ for i loop - $\left(\frac{n}{2}+1\right)$ times

for nested loops = $(\log_2 n)$ times

$$T(n) = O(i*j*k)$$

$$= O\left(\left(\frac{n}{2}+1\right)*(\log_2 n)*(\log_2 n)\right)$$

$$\boxed{T(n) = O[n(\log_2 n)^2]} = \text{ANS.}$$

5

Q8). Time Complexity = ?   ans).

```
function (int n)
{  if (n==1) return;         ——  O(1).
   for (i=1 to n)            ——  O(n)
   {  for (i= j to n)        ——  O(n^2).
      { printf("*");
      }
   }
   function (n-3);           ———T(n/3).
}
```

$\therefore\ T(n) = T(n/3) + n^2$ { ignore n for its of lesser power b/w n & n^2 }.

implementing MASTERS METHOD.

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

$\therefore\ c = \log_3 1 = 0$.

& $n^0 = 1 > (f(n) = n^2)$ $\Rightarrow\ \therefore\ \boxed{T(n) = O(n^2)}$

ans.

Q9). Time Complexity = ?

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

ans. initially:

$i = 1$ $j = 1\ 2\ 3 \cdots n$

$i = 2$ $j = 1\ 3\ 5 \cdots n/2$

$i = 3$ $j = 1, 4\ 7 \cdots n/3$

at some point of time -

$i = n$ $j = 1+1+1+ \cdots 1$

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

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

$= n \log(n-1) \quad - \log(n-1).$

$= n \log(n-1)$

$\boxed{T(n) = n \log n.}$ —— Ans.

Q10). what is the asymptotic relationship between n^k & c^n Assume $k >= 1$ & $c > 1$ are constants. Then find value of c and n for which relation holds.

Ans). Relation b/w n^k & c^n can be illustrated by :-

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

as $n^k \leq a\, c^n$ $\left[\forall\, n \geq n_0 \text{ and some constant } a > 0 \right]$

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a\, 2^1$

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