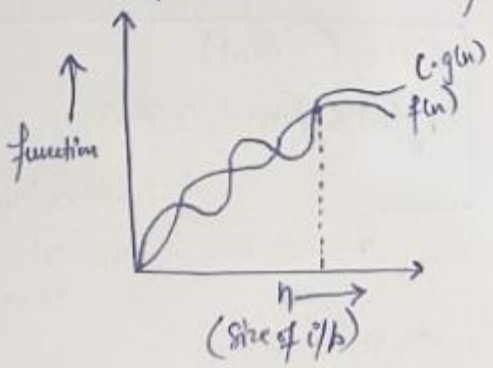
Assignment -> 1

(DAA) Design and Analysis of Algorithms

Ans-1 **Asymptotic Notation :-** These notation are used to tell the complexity of an algorithm, when input is very large. These are mathematical notations used to describe running time of an algorithm when the input tends towards a particular value or a limiting value.

→ **different Asymptotic Notation are :-**

▷ **Big-Oh (O) :-** $f(n) = O(g(n))$



→ g(n) is "tight" upper bound $f(n) = O(g(n))$

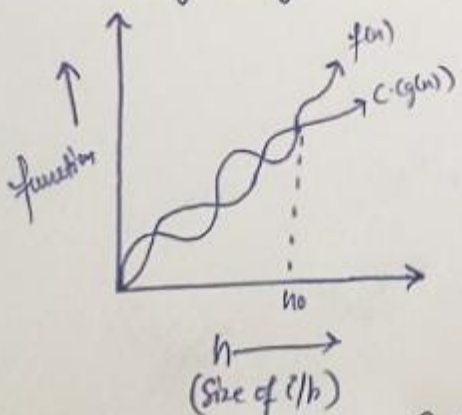
iff $f(n) \leq c \cdot (g(n))$ $\forall\ n \geq n_0$

and for some Constant, $c > 0$

Eg :-
```
for(i=1, i<=n; i++)
  { print(i); }   ---> O(1)
```

$T(n) = O(n)$

2) Big-Omega (Ω) :- $f(n) = \Omega(g(n))$



→ g(n) is "tight" lower bound $f(n) = \Omega(g(n))$

iff $f(n) \geq c \cdot (g(n))$ $\forall\ n \geq n_0$

and for some Constant, $c > 0$

Eg :- $f(n) = 2n^2 + 3n + 5$ $\quad g(n) = n^2$

$0 \leq c \cdot g(n) \leq f(n)$

$\Rightarrow 0 \leq c \cdot n^2 \leq 2n^2 + 3n + 5$

Yashraj

$\Rightarrow c \leq 2 + \frac{3}{n} + \frac{5}{n^2}$ $\left\{\because \text{On putting } n=\infty, \Rightarrow \frac{3}{n} \to \infty, \frac{5}{n^2} \to \infty\right\}$

$\Rightarrow c = 2$

$\Rightarrow 2n^2 \leq 2n^2 + 3n + 5$

Let, $n = 1$

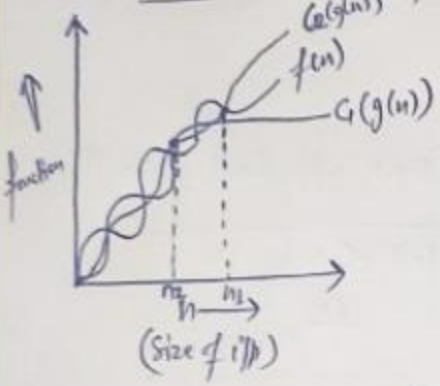
$2 \leq 10$ (True)

$c = 2, \; n = n_0 = 1$

$0 \leq 2n^2 \leq 2n^2 + 3n + 5$

(iii) Theta (Θ)

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



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

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

$C_1 g(n) \leq f(n) \leq C_2 g(n)$

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

and for some constant $C_1 > 0$ & $C_2 > 0$

eg:- $f(n) = 10 \log_2 n + 4$ $\quad g(n) = \log_2 n$

$f(n) \leq ((2 \cdot g(n))$

$\Rightarrow 10\log_2 n + 4 \leq 10 \log_2 n + \log_2 n$

$10 \log n + 4 \leq 11 \log n$

$C_2 = 11$

$4 \leq 11 \log_2 n - 10 \log_2 n$

$4 \leq \log n$

$16 \leq n$

here, $\forall n \geq 16$

$n_2 = 16$

$C_2 = 11$

$f(n) \geq C_1 \cdot g(n)$

$10 \log_2 n + 4 \geq 2 \log_2 n$

$C_1 = 1, \; n > 0$

$n_1 = 1 \Rightarrow n_0 = \max(n_1, n_2)$

$n_0 = 16$

$\log_2 n \leq 10 \log_2 n + 4 \leq 11 \log_2 n$

$C_1 = 1$

$C_2 = 11$

$\Rightarrow \Theta(\log_2 n)$

Verified ✓

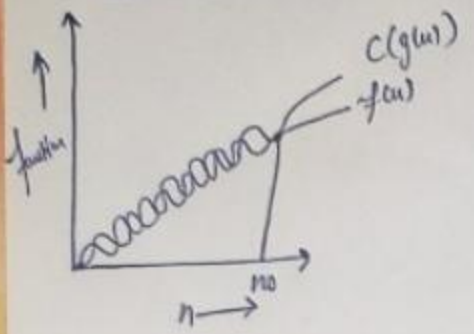
4) Small Oh (o) :- $f(n) = o(g(n))$

→ $g(n)$ is upper bound of function of $f(n)$

iff

$f(n) = O(g(n))$ when $f(n) < c(g(n))$ $\forall n > n_0$

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

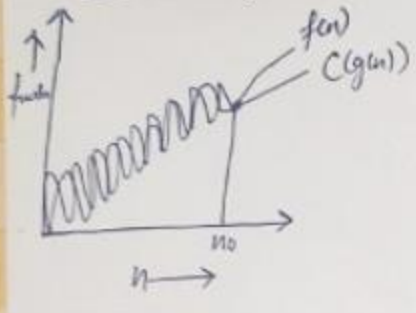


5) Small omega (ω) :- $f(n) = \omega(g(n))$

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

iff $f(n) = \omega(g(n))$ when $f(n) > c(g(n))$ $\forall n > n_0$

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



Ans-2 Time Complexity of → for (i=1 to n) $\{i = i * 2\}$

$i = \underbrace{1, 2, 4, 8, 16 \cdots n}_{k \text{ terms}}$ $\quad a = 1, \quad r = \frac{t_2}{t_1} = 2$

GP of k^{th} value $\Rightarrow t_k = ar^{k-1}$

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

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

$2n = 2^k$ $\quad \{\log \text{ on both side}\}$

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

$\log(2n) = k \log(2)$ $\quad \{\log(2) = 1\}$

$\log(2n) = k$

$k = \log 2 + \log n$

$k = 1 + \log n$

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

Yashwant

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

$\Rightarrow \therefore T(n) = 3T(n-1)$ —①

put, $n = n-1$ in eq ①

$T(n-1) = 3T(n-2)$ —②

put, ② in eq ①

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

put, $(n-2) = n$ in eq ①

$T(n-2) = 3T(n-3)$ —④

put, this in eq ③

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

$= 9T(n-3)$

generalised form,

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

putting $(n-k=0) \Rightarrow T(n) = 3^n T(0)$

$T(n) = 3^n$

$O(3^n)$

=

Ans4 $T(n) = \{2T(n-1)-1$; if $n>0$, otherwise $1\}$

$\Rightarrow T(n) = 2T(n-1) - 1$ —①

put $n = n-1$ in eq ①

$T(n-1) = 2T(n-2) - 1$ —②

put eq ② in eq ①

$T(n) = 4T(n-2) - 2 - 1$ —③

put $n = n-2$ in eq ②

$T(n-2) = 2T(n-3) - 1$ —④

put eq ④ in eq ③

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

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

generalised form:

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

put $n-k = 0$

$n = k$, $T(0) = 1$

Yashraj

$\Rightarrow T(n) = 2^n T(0) - 2^{n-1} - 2^{n-2} \cdots - 1$

$= 2^n - \underbrace{(2^{n-1} + 2^{n-2} + \cdots + 1)}_{k \text{ terms}}$ $\{a = 2^{n-1} ; r = 1/2\}$

Sum of GP $= \dfrac{2^{n-1}(1 - (1/2)^{n-1})}{1 - \frac{1}{2}} = 2^n - 2$

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

$O(2) \Rightarrow O(1)$

Ans-5 $S = 1, 3, 6, 10, 15 \cdots, n$

$\Rightarrow k^{th}$ term

$t_k = t_{k-1} + k$

$\Rightarrow k = t_k - t_{k-1}$ — (1) $\{t_k = n\}$

$k = n - t_{k-1}$ (loop runs k times)

Time Complexity $= O(1 + 1 + \cdots + n - t_{k-1})$

but, $t_{k-1} = C$ (Constant)

Time Complexity $= O(3 + n - c)$

$= O(n)$

Ans-6 $i * i \Rightarrow \underbrace{1^2, 2^2, 3^2, 4^2 \cdots n}_{k \text{ terms}}$

k^{th} term $\Rightarrow t_k = k^2$

$k^2 = n$

$k = n^{1/2}$

Time Complexity,

$O(1 + 1 + 1 + n^{1/2} + 1)$

$= O(n^{1/2}) = O(\sqrt{n})$

Yashraj

Ans-7 Time Complexity of

```
Void func (int n)                     ..... O(1)
{ int i, j, k, Count = 0;             .... O(1)
  for(i = n/2 ; i <= n ; i++)
    for(j = 1 ; j <= n ; j = j*2)     .... log(n)
      for(k = 1 ; k <= n ; k = k*2)   .... log(n)
        Count++;                      .... O(1)
}
```

$i = \frac{n}{2}, \frac{n+2}{2}, \frac{n+4}{2} \ldots$ upto n

general form: $t_k = \frac{n + k*2}{2}$

total terms = k+1

$t_{k+1} = n$

$\frac{n + (k+1)*2}{2} = n$

$2n = n + 2k + 2$

$\boxed{k = \frac{n}{2} - 1}$

i	j	k
$\frac{n}{2}$	$\log n$	$(\log n)^2$
$\frac{n+2}{2}$	$\log n$	$(\log n)^2$
$\frac{n+4}{2}$	$\log n$	$(\log n)^2$
⋮	⋮	⋮
n times	$\log n$ times	$(\log n)^2$

$\left(\frac{n}{2} - 1\right)$ times $\Rightarrow \left(\frac{n-1}{2}\right)(\log n)^2$

$O\left(\frac{n}{2}\log n - \log^2 n\right)$

$O(n \log^2 n)$

Ans-8 Time Complexity of

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

Yashraj

for function Call,

$\underbrace{n, n-3, n-6, n-9, \ldots 1}_{k \text{ terms}}$

⇒ AP with $d = -3$

$l = a + (k-1)d$

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

$k = (n+2)/3$

→ function gives a recursive call $(n+2)/3$ times

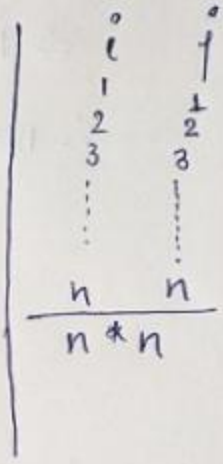
Time Complexity $= (n+2)/3 \cdot (n) \cdot (n)$

$\therefore O(n^3)$

=

9) Time Complexity of

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



Time Complexity, $O(n*n)$

$O(n^2)$

=

Yashaj