Date

Design And Analysis of Algorithms

Tutorial - 1

Name :- Vidushi Tiwari
Section : CST SPL 2
Roll no : 27
University Roll no :- 2017579

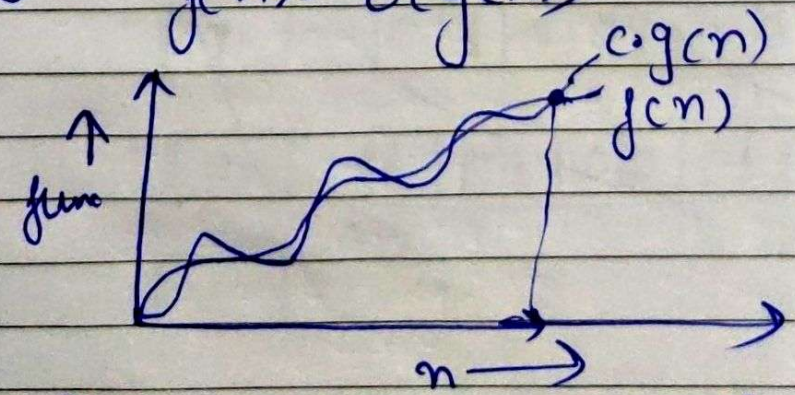
1. what do you understand - - - -- examples

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

Different Asymptotic Notations :-

(1) **Big-Oh (O) :-**

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



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

Date

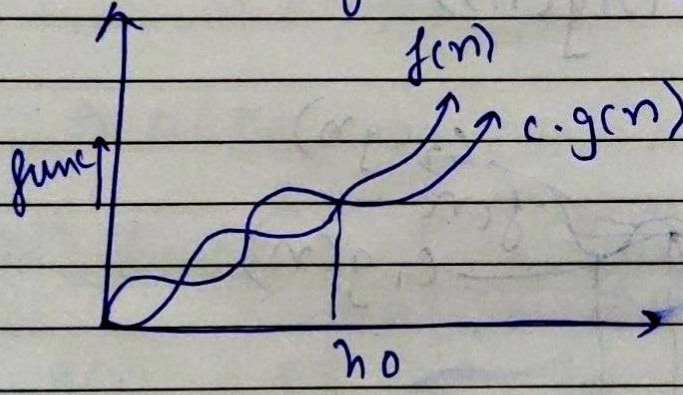
iff
$f(n) \leq c \cdot g(n)$
$\forall n \geq n_0$ & some constant, $c > 0$.

eg:-
```
for(i=1; i<=n; i++)
{
    print(i);   — O(1)
}
```
$T(n) = O(n)$

(i)
1
2
3
.
.
n times
↳ $O(n)$

(ii) **Big Omega (Ω):**

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



g(n) is "tight" lowerbound.

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

if,
$f(n) \geq c.g(n)$
$\forall n \geq n_0$, & some constant $c > 0$.

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

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

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

$\Rightarrow c \leq 2 + \frac{3}{n} + \frac{5}{n^2}$

On putting $n = \infty$, $\Rightarrow \frac{3}{n} \to \infty$, $\frac{5}{n^2} \to \infty$

Date

$\Rightarrow c = 2$,

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

On putting $n = 1$,

$2 \leq 2 + 3 + 5$

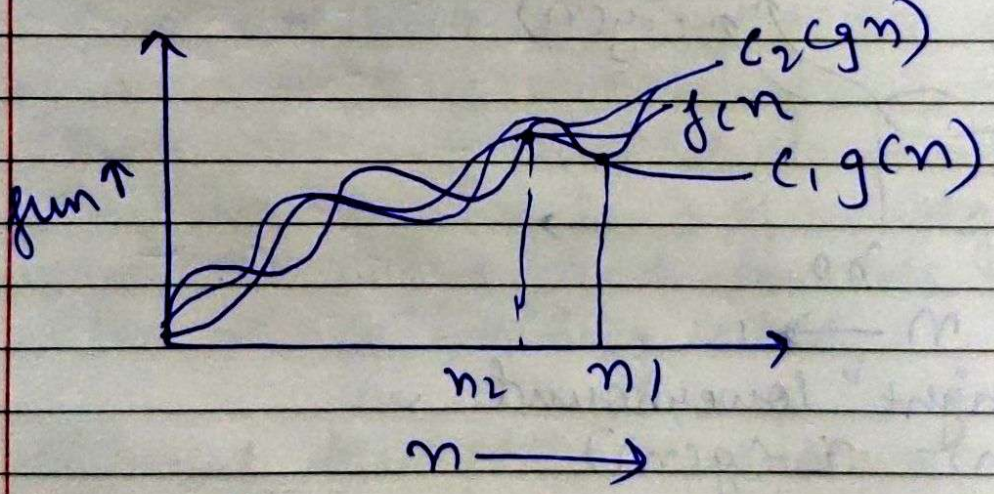
$2 \leq 10$ (True)

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

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

(iii) **Big Theta (Θ):**

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



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

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

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

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

$c_2 = 11$

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

$4 \leq \log_2 n$

$16 \leq n$.

Here,

Date

$\forall\ n \geq 16$
$n_2 = 16$
$c_2 = 11$

$f(n) \geq c_1 \cdot g(n)$
$10 \log_2 n + 4 \geq 2 \log_2 n$
$c_1 = 1$, $n > 0$
$\Rightarrow n_1 = 1$ $\Rightarrow n_0 = \max(n_1, n_2)$
$\Rightarrow n_0 = 16$

$\log_2 n \leq 10 \log_2 n + 4 \leq 11 \log_2 n$
$c_1 = 1$
$c_2 = 11$

$\Rightarrow O(\log_2 n)$

(IV) Small oh (o):

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

$g(n)$ is the upper bound of the function $f(n)$.

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

$g(n)$ when, $f(n) < c \cdot g(n)$
$\forall\ n > n_0$
& $\forall$ constants, $c > 0$

(V) Small Omega (ω):

$f(n) = \omega(g(n))$
$\forall\ n > n_0$
& $\forall\ c > 0$,

Date

Q2. Time complexity of :-

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

⇒ i = 1, 2, 4, 8, 16 ... n

(k terms)

a = 1 , r = 2

⇒ k^{th} term $t_k = ar^{k-1}$

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

$= 2^{k-1}$

take $\log_2$ both sides

⇒ $\log_2 n = \log_2 2^{k-1}$

⇒ $\log_2 n = (k-1) \log_2 2$

⇒ $\log_2 n = (k-1)$

$k = 1 + \log_2 n$

$T(n) = O(k)$

$= O(1 + \log_2 n)$

$= O(\log_2 n)$

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

⇒ ∵ $T(n) = 3T(n-1)$ ——— (1)

put $n = n-1$ in eqn (1)

$T(n-1) = 3T(n-2)$ ——— (2)

put (2) in eqn (1)

$T(n) = 3[3T(n-2)]$ ——— (3)

put $n = (n-2)$ in eq (1)

$T(n-2) = 3T(n-3)$ ——— (4)

Date

put this in eqn (3)

$\Rightarrow T(n) = 9[3T(n-3)]$

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

Generalised form :-

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

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

$T(n) = 3^n$

$\Rightarrow O(3^n)$

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

$\Rightarrow T(n) = 2T(n-1)-1$ — (1)

put $n-1$ in eqn (1)

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

put this in eqn (1)

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

$T(n) = 4T(n-2)-2-1$ — (3)

put $n = n-2$ in eqn (1)

$\Rightarrow T(n-2) = 2T(n-3)-1$ — (4)

put this value in eqn (3)

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

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

$\Rightarrow$ generalised form :-

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

put $n-k=0$

$\Rightarrow n=k$, $T(0)=1$ (given)

Date

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

$= 2^n - 2^{n-1} - 2^{n-2} \cdots - 1$

$= 2^n - (\underbrace{2^{n-1} + 2^{n-2} + \cdots + 1}_{K \text{ terms}})$

$\Rightarrow a = 2^{n-1}, \; r = 1/2$

Sum of G.P $= \dfrac{2^{n-1}(1-(1/2)^{n-1})}{1-1/2} = 2^n - 2$

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

$\underline{O(2) \Rightarrow O(1)}$

So $S = 1, 3, 6, 10, 15 \cdots n$.

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

$\Rightarrow K^{th}$ term,

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

$K = t_k - t_{k-1}$ ——— (1)

$\Rightarrow k = n - t_{k-1}$

(loop runs k times)

S
1
3
6
10
15
⋮
n

K times

Time comp. $= O(1+1+1+n-t_{k-1})$

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

T. comp. $= O(3+n-C)$

$= O(n)$

Date

6. i*i ⇒ $1^2, 2^2, 3^2, 4^2, 5^2, \ldots n$

k terms.

i*i
1^2
2^2
3^2
⋮
n

k^{th} term ⇒

$t_k = k^2$

$k^2 = n$

$k = n^{1/2}$

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

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

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 ⇒ $\frac{n}{2}, \frac{n+2}{2}, \frac{n+4}{2}, \frac{n+6}{2}$ --- upto n.

⇒ $\frac{n+0\times2}{2}, \frac{n+(1*2)}{2}, \frac{n+(2*2)}{2}, \frac{n+(3*2)}{2}$ - upto.

General form :-

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

Date

total terms = $k+1$

$t_{k+1} = n$

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

$n+2k+2 = 2n$

$2k = n-2$

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

i	j	k
$\frac{n}{2}$	$\log_2 n$ times	$(\log_2 n)^2$
$\frac{n+2}{2}$	$\log_2 n$ times	$(\log_2 n)^2$
$\frac{n+4}{6}$	$\log_2 n$ times	$(\log_2 n)^2$
$\left(\frac{n}{2}-1\right)$ times	$\Rightarrow$	$\left(\frac{n}{2}-1\right)\cdot(\log_2 n)^2$

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

$O(n \log^2 n)$

Date

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);
}
```

for function call,

$\underbrace{n, n-3, n-6, n-9 \cdots 1}_{K \text{ terms}}$

⇒ AP with $d = -3$

$\Rightarrow l = a + (K-1)d$

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

$(K-1) = \frac{n-1}{3}$

$K = \frac{(n-1)+3}{3}$

$= \frac{n+2}{3}$

⇒ function gives a recursive call $\frac{n+2}{3}$ times

⇒ Time complexity $= \frac{(n+2)}{3}(n)(n)$

$= n^3$

$\therefore O(n^3)$

Date

i: $n/2, n+2/2, n+4/2, n+6/2$ --- upto n.

$\Rightarrow \frac{n+0\times2}{2}, \frac{n+1\times2}{2}, \frac{n+2*2}{2}, \frac{n+3*2}{2}$ --- upto n

$\Rightarrow$ General form:

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

total terms $= k+1$

$\Rightarrow t_{k+1} = n$

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

$= n+2k+2 = 2n$

$= 2k = n-2$

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

$\Rightarrow$

i	j	k
$n/2$	$\log n$	$(\log n)^2$
$n+2/2$	$\log n$	$(\log n)^2$
$n+4/6$	$\log n$	$(\log n)^2$
|	|	|
n	$\log n$ terms	$(\log n)^2$

$(n/2-1)$ terms

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

$\Rightarrow O(n/2 \log^2 n - \log n)$

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

Date

9. TC of void function (int n)

```
{
  for(i=1 to n)
  {
    for (j=1; j<=n; j++)
      print("*");
  }
}
```

for i=1 → j = 1, 2, 3, 4, ---- n = n
for i=2 → j = 1, 3, 5 ---- n = n/2
for i=3 → j = 1, 4, 7 ---- n = n/3
!
for i=n ⇒ j = 1, ---- , n = 1

$$\Rightarrow \sum_{j=n}^{1} n + n/2 + n/3 + n/4 + \cdots + 1$$

$$\Rightarrow \sum_{j=n}^{1} n[1 + 1/2 + 1/3 + \cdots + 1/n]$$

$$\Rightarrow \sum_{j=n}^{1} n \log n.$$

⇒ T(n) = [n log n]

T(n) = O(n log n)

10. As given n^k & c^n

relation b/w n^k & c^n is $n^k = O(c^n)$

as $n^k \leq ac^n \; \forall \; n \geq n_0$ for (a) constant (n>0)

for $n_0 = 1$

$c = 2$

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

Date

$\therefore n_0 = 1 \text{ & } c = 2.$