VANSH KALRA
CST
25

DAA

2017576

DATE
PAGE

Tutorial - 1

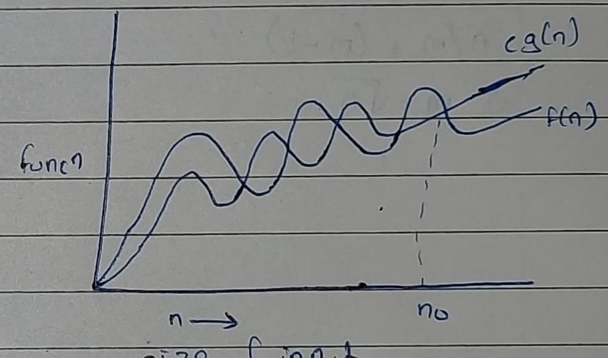
Q1. What do you understand by Asymptotic notations. Define different Asymptotic notation with examples.

A1. Asymptotic notation - They are the mathematical notations used to describe the running time of an algorithm where the input tends towards a particular value or a limiting value.

Different asymptotic notations -

i). Big O(n)

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



$f(n) = O(g(n))$
iff
$f(n) \leq cg(n)$
$\forall\ n \geq n_0$
for some constant, $c > 0$
g(n) is "tight" upper bound of f(n).

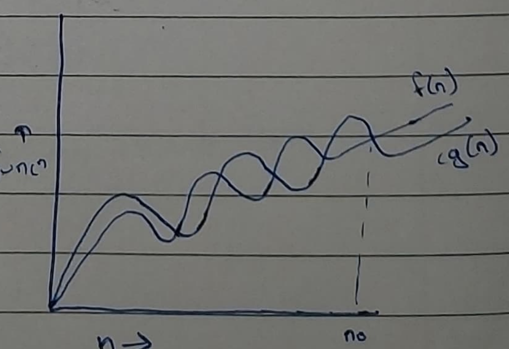
E.g. - $f(n) = n^2 + n$
$g(n) = n^3$
$n^2 + n \leq c.n^3$
$n^2 + n = O(n^3)$

ii). Big Omega (Ω)

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

g(n) is "tight" lower bound of function f(n)

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



iff

$$f(n) \geq c\,g(n)$$
$$\forall\ n \geq n_0$$

for some constant $c > 0$

E.g. - $f(n) = n^3 + 4n^2$

$g(n) = n^2$

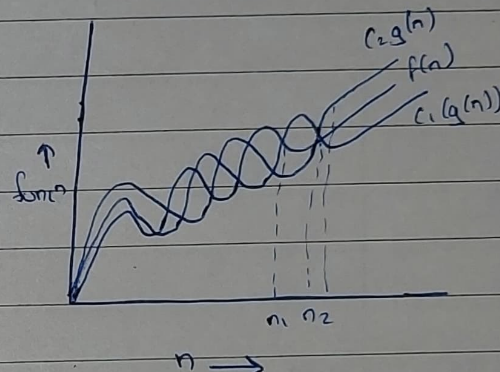
$n^3 + 4n^2 = \Omega(n^2)$

iii). **Big Theta (θ)**

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

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

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



iff

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$
$$\forall\ n \geq \max(n_1, n_2)$$

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

E.g. - $3n + 2 = \theta(n)$ as $3n + 2 \geq 3n$ and

$3n + 2 \leq 4n$ for n, $k_1 = 3$, $k_2 = 4$ & $n_0 = 2$

iv). **Small o (θ)**

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

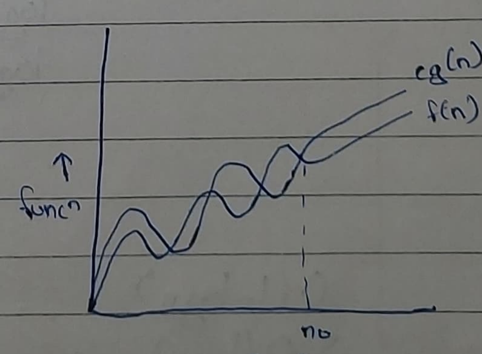
$g(n)$ is upper bound of function $f(n)$.

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

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

$$\forall\ n > n_0$$

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



E.g. - $f(n) = n^2$

$g(n) = n^3$

$n^2 = o(n^3)$

v). <u>Small Omega (ω)</u>

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

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

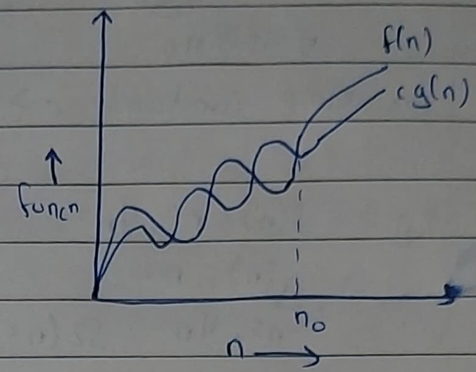
$$f(n) = \omega(g(n))$$

when

$$f(n) > c\,g(n)$$

$$\forall\, n > n_0$$

$$\&\ \forall \text{ constant } c > 0.$$



Q2. what should be time complexity of:

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

A2. $\sum_{i=1}^{n} 1+2+4+8+\dots+n$

G.P. kth value => $T_k = ar^{k-1}$

$$= 1 \times 2^{k-1}$$

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

$$2n = 2^k$$

$$\log 2n = k \log_2 2$$

$$\log_2 2 + \log_2 n = k \log_2 2$$

$$\log n + 1 = k$$

$$k = \log n + 1$$

$O(1 + \log n)$

=> $O(\log n)$ Ans.

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

A3. $T(n) = 3T(n-1)$ — ①

Let $n = n-1$

$T(n-1) = 3T(n-2)$ — ②

Put ② in ①

$T(n) = 3 \times 3T(n-2)$ — ③

Put $n = n-2$

$T(n-2) = 3T(n-3)$ — ④

Put ④ in ③

$T(n) = 3 \times 3 \times 3T(n-3)$ — ⑤

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

$T(n) = 3^n T(0)$

$T(n) = 3^n$

$\boxed{T(n) = O(3^n)}$

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

A4. $T(n) = 2T(n-1) - 1$ — ①

Let $n = n-1$

$T(n-1) = 2T(n-2) - 1$ — ②

Put ② in ①

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

$T(n) = 4T(n-2) - 2 - 1$ — ③

Put $n = n-2$

$T(n-2) = 2T(n-3) - 1$ — ④

Put ④ in ③

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

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

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

DATE
PAGE

$GP = 2^{k-1} + 2^{k-2} + \dots 1$

$a = 2^{k-1}$

$r = 1/2$

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

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

$$= 2^k(1-(1/2)^k)$$

$$= 2^k - 1$$

Let $n - k = 0$

$n = k$

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

$T(n) = 2^n . T(0) - (2^n - 1)$

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

$\boxed{T(n) = O(1)}$

Q5. What should be time complexity of -

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

As. $i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \dots\dots$

$s = 1 + 3 + 6 + 10 + 15 + 21 \dots\dots n$

DATE

PAGE

sum of $s = 1 + 3 + 6 + 10 + \dots + \sqrt{n}$ — ①

also $s = 1 + 3 + 6 + 10 + \dots + \sqrt{n-1} + \sqrt{n}$ — ②

① - ②

$0 = 1 + 2 + 3 + 4 + \dots n - \sqrt{n}$

$T_K = 1 + 2 + 3 + 4 + \dots + k$

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

for k iterations

$1 + 2 + 3 + \dots + k <= n$

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

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

$O(k^2) <= n$

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

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

Q6 Time complexity of -

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

DATE

PAGE

A6.

$i^2 <= n$

$i <= n$

$i = 1, 2, 3, 4, \ldots, \sqrt{n}$

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

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

$T(n) = \frac{n + \sqrt{n}}{2}$

$\boxed{T(n) = O(n)}$

Q7. Time complexity of :

```
void fn (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++;
}
```

A7.

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

DATE

PAGE

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

$\boxed{\Rightarrow O(n(\log n)^2)}$

Q8. Time complexity of -

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

A8. for (i=1 to n)

j=n times every turn

$i \times j = n^2$

Now, $T(n) = n^2 + T(n-3)$

$$\left.\begin{aligned} T(n-3) &= (n^2 3)^2 + T(n-6); \\ T(n-6) &= T(n^2 6)^2 + T(n-5) \end{aligned}\right\} k \text{ times.}$$

$\vdots \qquad \vdots$

$T(1) = 1$

Now substitute each value in $T(n)$

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$

DATE
PAGE

Let

$n - 3k = 1$

$k = (n-1)/3$

Total terms = $k+1$.

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$

$T(n) \approx n^2 + n^2 + n^2 + \dots$ ($k+1$ times)

$T(n) \approx kn^2$

$T(n) \approx \frac{(n-1)}{3} n^2$

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

Q9. Time complexity of -

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

A9. for: $i = 1$ $\quad j = 1+2+\dots\dots$ $(n \geq j+i)$

$i = 2$ $\quad j = 1+3+5+\dots\dots$ "

$i = 3$ $\quad j = 1+4+7+\dots\dots$ "

⋮ ⋮

m^{th} term of AP is

$T(m) = a + d \times m$

$T(m) = 1 + d \times m$

$(n-1)/d = m$

DATE
PAGE

for $i = 1$ $(n-1)/1$ times

$i = 2$ $(n-1)/2$ times

$i = 3$ $(n-3)/3$ times

$i = n-1$ 1

We get

$$T(n) = i_1 j_1 + i_2 j_2 + \dots + i_{n-1} j_{n-1}$$

$$= \frac{(n-1)}{1} + \frac{(n-2)}{2} + \frac{(n-3)}{3} + \dots + 1$$

$$= n + n/2 + n/3 + \dots + n/n-1 \quad -n \times 1 + \dots$$

$$= n\left[1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-1}\right] - n - 1$$

$$= n \times \log n - n + 1$$

$$\because \int \frac{1}{x} = \log x$$

$$\boxed{T(n) = O(n \log n)}$$

S10. For the funcn, n^k & c^n, what is the asymptotic relationship b/w these functions? Assume that $k >= 1$ & $c > 1$ are constants. Find out the value of c & no. for which relation holds.

A10.

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

$n^k \leq a(c^n)$

$\forall\ n \geq n_0$ &

constant, $a > 0$

for $n_0 = 1$

$c = 2$

$1^k \leq a^2$

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