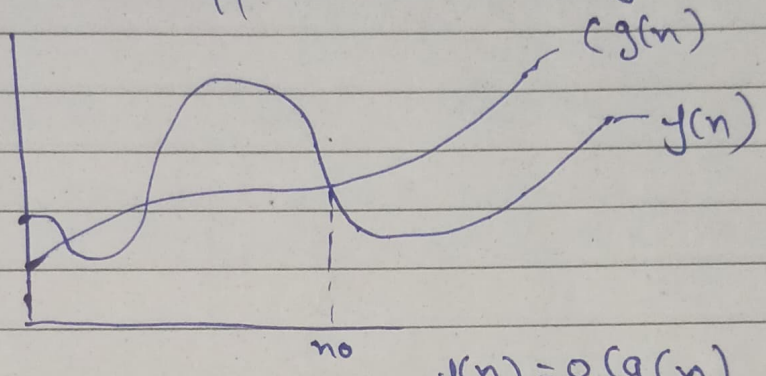
(Tutorial-1) Ritish Sharma
CST-14

Q1) Asymptotic Notation : They are the Mathematical notation used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

There are mainly three asymptotic notations:

(1) provide worst complexity
(2) provide upper bound of an running time algo.

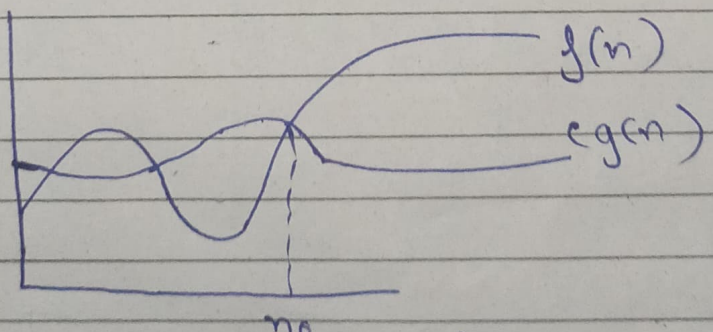


$f(n) = O(g(n))$

$O(g(n)) = \{ f(n) :$ there exist +ve constant C & n_0 such that $0 \leq f(n) \leq g(n)$ for all $n \geq n_0 \}$

(11) Omega Notation (Ω)

- provide best case complexity
- rep lower bound of running time algo.



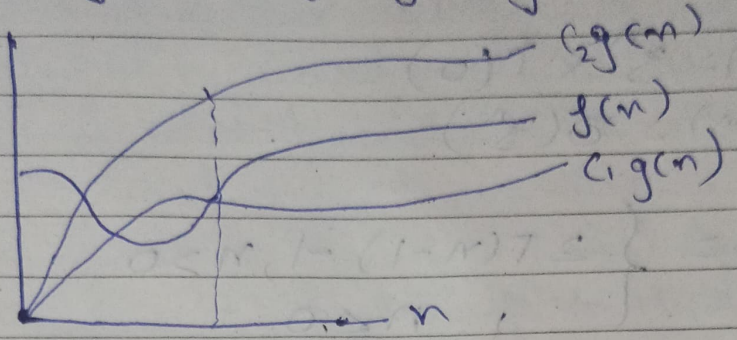
$f(n) = \Omega(g(n))$

$\Omega(g(n)) = \{ f(n) :$ there exist positive constant c and n_0 such that $0 \leq (g(n)) \leq f(n)$ for all $n \geq n_0 \}$

Ritik Sharma

(4) theta Notation (Θ notation)

⇒ used for analysing avg. time complexity



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

$\Theta(g(n)) = \{ f(n)$: there exist positive constant c_1, c_2 no such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0 \}$

Q2
```
for (i = 1 to n)
  { j = i * 2;
  }
```

$j = 1, 2, 2^2 \ldots 2^k$

$2^k \le n \Rightarrow k = \log_2 n$

$2^k \le n \Rightarrow k = \log_2^n$

$\therefore \sum_{j=1}^{k} 2 \Rightarrow 1 + 1 + 1 \ldots$ k times

$T(n) = O(\log n)$

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

using forward substitution

Partition Sharma

$T(0) = 1 \quad \ldots \quad O(1)$

$T(1) = 3T(0)$

$T(2) = 3^2 T(0)$

$\vdots$

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

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

Q4 $T(n) = \begin{cases} 2T(n-1) - 1, & n > 0 \\ 1, & n \leq 0 \end{cases}$

using forward sub.

$T(1) = 2T(0) - 1 \; ; \quad T(0) = 1$

$= 1$

$T(2) = 2 \times T(1) - 1 = 1$

$\vdots$

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

$\therefore \quad T(n) = O(1)$

Q5

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

```
for(i=1)          | for(i=2)
s=1+2;            | j=1+2+3
```

for (i=k)

$1+2+\ldots+k <= n$

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

$= \left(\frac{k^2+k}{2}\right) \leq n$

Ritish Sharma

$O(k^2) <= n \Rightarrow k = O(\sqrt{n})$

$\therefore T(n) = O(\sqrt{n})$ Ans

Q6
```
void function (int n) {
    int i, count = 0;
for (int i=1, i*i <= n, i++)
    count ++ →  O(1)
}
```

let 'k' be max +ve value such that

$k^2 \leq n$

$\therefore k = \sqrt{n}$

$i^2 \leq n$

$\therefore \sum_{i=1} 1 \Rightarrow 1+1+\ldots$ k times

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

Q7
```
void function (int n){
    int i, j, k, count = 0;
for (i = N/2, i <= n, i++)
    {
    for (j=1, j<=n, j = i*2)
        {
        for (k=1; k<=n; k = k*2)
        count ++;
        }
    }
```

let 'm' be highest value of 2^m such that

$2^m <= n$ $\therefore m = \log_2 n$

Ritik Sharma

⇒ for $i = n/2$. $j = \log n$ $k = \log$
$i = (n/2 + 1)$ " "
⋮
$i = n$ " "

$\therefore \sum_{i=n/2}^{n} j \times k$

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

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

Q8 func. (int n)
{
if (n == 1) return;
for (i = 1 to n)
{
for (j = 1 to n) {
print (...) → O(1)
}
}
func (n-3);
}

for : for (i = 1 to n)
we get j = n times every turn

$\therefore i \times j = n^2$.

Now, $T(n) = n^2 + T(n-3)$;
$T(n-3) = T(n-6) + (n-3)^2$
$T(n-6) = T(n-9) + (n-6)^2$
⋮
$T(1) = 1$ } k terms

Now subs. each value in $T(n)$

Ritish Sharma

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

Let

$(n-3k) = 1$

$\therefore \quad k = (n-1)/3$

total terms $= k+1$

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

$T(n) \simeq n^2 + n^2 + n^2 \ldots$ (k times + 1)

$T(n) \simeq kn^2$

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

$\therefore T(n) = O(n^3)$

Q9)

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

for i: $i=1$, $i=2$, $i=3$, ...

$j = 1+2 \ldots (n \geq j+1)$

$j = 1+3+5 \ldots$ "

$j = 1+4+7 \ldots$ "

k^{th} term of AP is

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

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

Rishi Sharma

$(n-1)/d = m$

$\therefore$ for $i=1$ $\quad j$ $\quad (n-1)/1$ times

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

$i=3$ $\quad (n-1)/3$ times

$i = n-1$ $\quad 1$

we get.

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

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

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

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

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

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

$\therefore T(n) = O(n \log n)$

Q10) we have given

n^k & c^n

as $k=1$ & $c>1$

$\Rightarrow$ for values $k \geq 1$, $c > 1$

we have $c^n > n^k$

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

$\forall\ n \geq n_0$, & some constant $k_0 > 0$

Ritika Sharma

Date
Page

$\Rightarrow k_0 c^n \geq n^k$.

for $c > 1$ & $n = 1$.
we get

$\Rightarrow k_0 c \geq 1$

$\therefore$

$\boxed{c > 1 \ \& \ n_0 = 1}$ Ans