Design and Analysis of Algorithms

NAME : AKSHIT SINGH

SECTION : F

ROLL NO : 42

COURSE : B. Tech CSE

Tutorial - 1

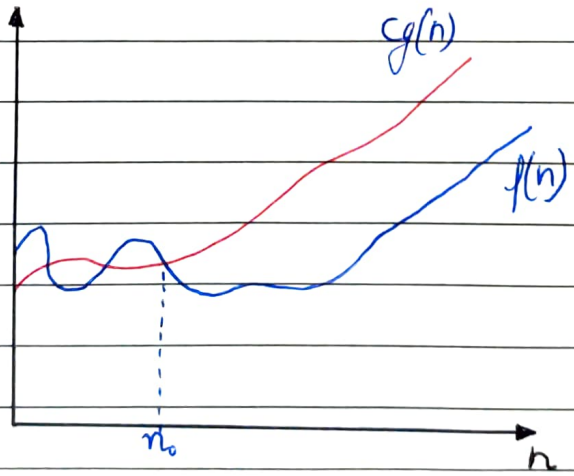
Design And Analysis of Algorithm

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

Asymptotic notations are used to represent the complexities of algorithms for Asymptotic analysis. These notations are mathematical tools to represent the complexities. There are three notations ~~are~~ that are commonly used.

- Big Oh Notation

Big-Oh (O) notation gives an upper bound for a function $f(n)$ to within a constant factor.



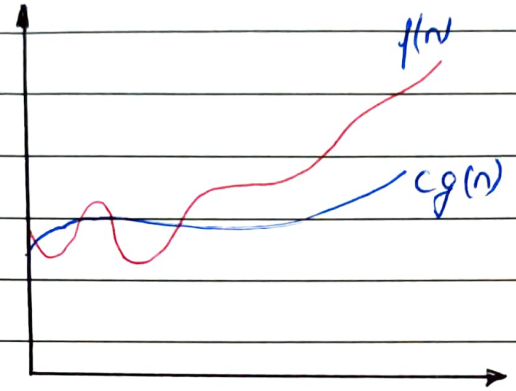
We write $f(n) = O(g(n))$, If there are positive constants and c such that, to the right of no. the $f(n)$ always lies on or below $c * g(n)$.

$O(g(n)) = \{ f(n) :$ There exist positive constant c and no such that $0 \leq f(n) \leq cg(n)$, for all $n \geq no. \}$

eg:- ① $f(n) = n^2 + n$ $\quad n^2 + n \leq c * n^3$)
$g(n) = n^3$ $\quad n^2 + n \geq O(n^3$

- **Big Theta Notation**

Big-Omega(Ω) notation gives a lower bound for a function $f(n)$ to within a constant factor



Ex: $f(n) = n^3 + 4n^2$

$g(n) = n^2$

i.e., $f(n) \geq c * g(n)$

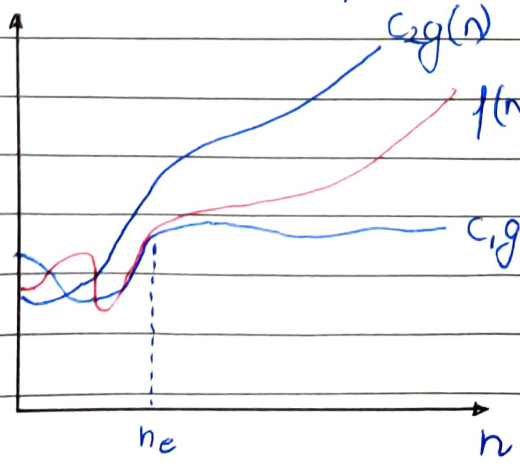
$n^3 + 4n^2 = \Omega(n^2)$

We write $f(n) = \Omega(g(n))$, if there are positive constants and c such that, to the right of the $f(n)$ always lies on or above $c*g(n)$

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

- **Big Theta Notation**

Big-Theta (Θ) notation gives bound for a function $f(n)$ to within a constant factor.



Ex: $3n+2 = \Theta(n)$ as

$3n+2 \geq 3n$ if

$3n+2 \leq 4n$ for n_1

$C_1 = 3, C_2 = 4$ & $n_0 = 2$

We write $f(n) = \Theta(g(n))$, if there are positive constants and c1 and c2 such that, to the right of the $f(n)$ always lies b/w $c1 * g(n)$ and $c2 * g(n)$ inclusive.

$\Theta(g(n)) = \{ f(n)$: There exist positive constant c1, c2 and such that $0 \leq c1 g(n) \leq c2 g(n)$, for all $n \geq n_0 \}$

iv) Small O (o)

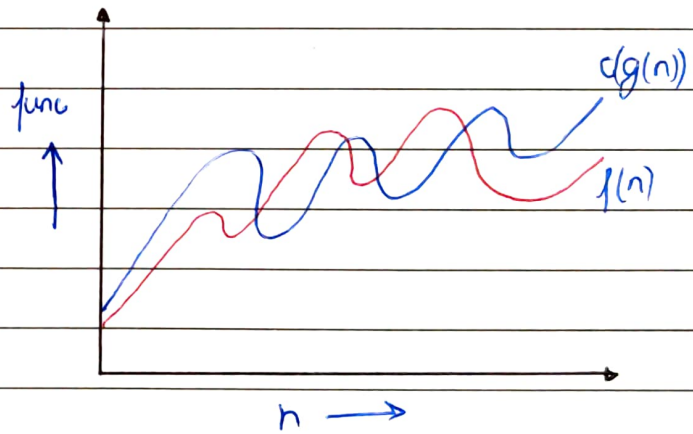
When $f(n) = Og(n)$ gives the upper bound

i.e. $f(n) = og(n)$

if and only if

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

$\forall\ n > n_0$ & $n > 0$



Ex: $f(n) = n^2$; $g(n) = n^3$

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

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

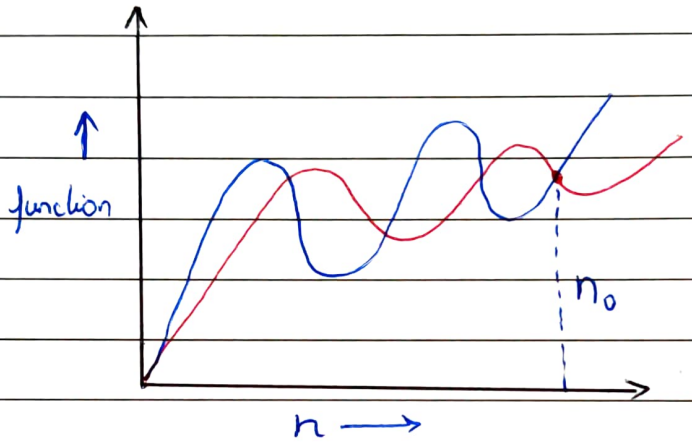
Ex: $f(n) = n^2$; $g(n) = n^3$

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

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

v) Small Omega (ω)

It gives the 'lower bound' i.e., $f(n) = \omega(g(n))$ where $g(n)$ is lower bound of $f(n)$ if and only if $f(n) > c * g(n)$ $\forall$ $n > n_0$ if some constant, $c > 0$



2. What should be Time Complexity of:

```
for (int i = 1 to n)
{
    i = i * 2;        ⟶ O(1)
}
```

Sol- for $i \Rightarrow 1, 2, 4, 6, 8 \ldots\ldots$ n times

i.e., Series is a GP

So, $a = 1$, $r = 2/1$

K^{th} value of GP:

$$t_K = a r^{K-1}$$
$$t_K = 1(2)^{K-1}$$
$$2n = 2^K$$

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

$\log_2 2 + \log_2 n = k$

$\log_2 n + 1 = k$ (Neglecting '1')

So, Time Complexity $T(n) \Rightarrow O(\log_2 n)$

3. $T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0 \\ \text{otherwise } 1 & \end{cases}$

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

$T(n) = 1$

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

$T(n-1) = 3T(n-2)$ ——— (2)

put (2) in (1)

$T(n) = 3 \times 3T(n-2)$

$T(n) = 9T(n-2)$ ——— (3)

put $x = x - 2$ in (1)

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

put in (3)

$T(n) = 27T(n-3)$ ——— (4)

Generalising Series,

$T(k) = 3^k T(n-k)$ ——— (5)

for K^{th} term, let $n-K=1$ (Base Case)

$K = n-1$
put in (5)

$T(n) = 3^{n-1} T(1)$
$T(n) = 3^{n-1}$ (neglecting '3')
$T(n) = O(3^n)$

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

$T(n) = 2T(n-1)-1$ ——— (1)

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

put in (2)
$T(n) = 2 \times (2T(n-2)-1)-1$
$= 4T(n-2)-2-1$ ——— (3)

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

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

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

Generalising term

$T(n) = 2^k T(n-K) - 2^{k-1} - 2^{k-2} \ldots 2^0$

= K^th term

Let n - k = 1

k = n - 1

$$T(n) = 2^{n-1} T(1) - 2^k \left(\frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^n}\right)$$

$$= 2^{n-1} - 2^{n-1} \left(\frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^n}\right)$$

i.e., Series of GP

$a = 1/2$, $r = 1/2$

So,

$$T(n) = 2^{n-1} \left(1 - \left(\frac{1/2 \left(1 - (1/2)^{n-1}\right)}{1 - 1/2}\right)\right)$$

$$= 2^{n-1} \left(1 - 1 + (1/2)^{n-1}\right)$$

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

$$T(n) = O(1)$$ Ans

5. What should be time complexity of

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

Sol ⇒ i = 1 2 3 4 5 6 ...

s = 1 + 3 + 6 + 10 + 13 + ...

Sum of s = 1 + 3 + 6 + 10 + ... + n ——— (1)

Also, S = 1 + 3 + 6 + 10 + ... + T_{n-1} + T_n ——— (2)

0 = 1 + 2 + 3 + 4 + ... n − T_n

T_K = 1 + 2 + 3 + 4 + ...

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

for K iterations

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

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

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

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

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

6. Time Complexity of

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

As $i^2 = n$

$i = \sqrt{n}$

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

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

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

$T(n) = O(n)$ Ans

7. Time Complexity of

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

Sol - Since, for $k = k^2$

$k = 1, 2, 4, 8, \ldots, n$

∴ Series is in GP

So, $a = 1$, $r = 2$

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

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

$n = 2^k - 1$

$n + 1 = 2^k$

$\log_2(n) = k$

i	j	k
1	$\log(n)$	$\log(n) * \log(n)$
⋮	⋮	
2	$\log(n)$	$\log(n) * \log(n)$
⋮	⋮	
n	$\log(n)$	$\log(n) * \log(n)$

$T.C = O(n * \log n * \log n)$

$= O(n \log^2(n))$ — Ans

8. Time complexity of -

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

Sol - for (i = 1 to n)
we get j = n times every time
∴ i * j = n^2

k^th, Now,

$T(n) = n^2 + T(n-3)$;
$T(n-3) = (n-3)^2 + T(n-6)$;
$T(n-6) = (n-6)^2 + T(n-9)$;
and $T(1) = 1$;

Now, subtract each value in $T(n)$
$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$
Let

$n - 3k = 1$
$k = (n-1)/3$ total terms = $k+1$

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

$T(n) \simeq kn^2$

$T(n) \simeq (k-1)/3 * n^2$

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

9. Time complexity of -

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

Sol -

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

$i = 2$ $\quad j = 1 + 3 + 5 + \dots (n \geq j + i)$

$i = 3$ $\quad j = 1 + 4 + 7 + \dots (n \geq j + i)$

n^{th} term of AP is

$T(n) = a + d * m$

$T(n) = a1 + d * m$

$(n-1)/d = n$

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

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

$i = n-1$

we get,

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

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

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

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

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

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

$T(n) = O(n \log n)$ ⟶ Ans

10. For the functions, n^k and c^n, what is the asymptotic relationship between these functions?
Assume that $k >= 1$ and $c > 1$ are constants. Find out the value of c and no. for which relation holds.

Sol – As given n^k and c^n

Relationship b/w n^k & c^n is

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

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

$\forall\ n > n_0$ & constant, $a > 0$

for $n_0 = 1$; $c = 2$

$\Rightarrow 1^k < a^2$

$\Rightarrow$ $n_0 = 1$ & $c = 2$ ⟶ Ans