Name: Nishtha Jain
Section: H
Roll No.: 39

Design And Analysis Of Algorithms (Tutorial-1)

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

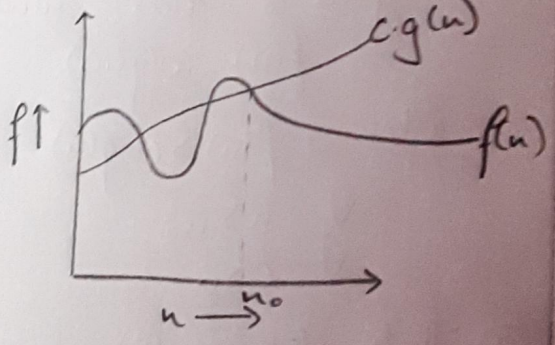
⇒ Asymptotic notations are those notations that describes the limiting behaviour of a function.
There are five different types of notations.

* Big O (O)
* Big Omega (Ω)
* Theta (Θ)
* Small O (o)
* Small Omega (ω)

(i) Big O Notation (O)

It gives an upper bound for a function $f(n)$ to within a constant factor.

$f(n) = O(g(n))$
$g(n)$ is "tight" upper bound
$f(n) = O(g(n))$
iff $f(n) \leq c \cdot g(n)$
$\forall\ n \geq n_0$.



Eg:
```
for (i=1 ; i≤n ; i++)
{
    Sum+=i;        ⇒ O(1+n+n+n)
}                  = O(n)
```

(ii) Small O Notation (o)

$f(n) = O(g(n))$
$g(n)$ is upper bound of function $f(n)$
$f(n) = O(g(n))$

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

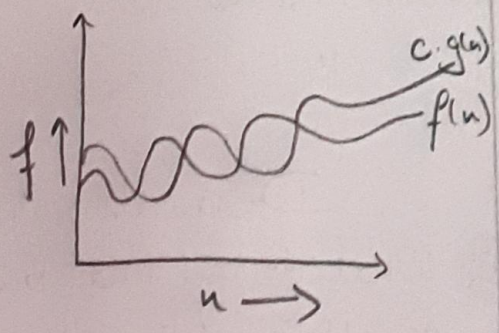
$g(n)$ is "upper" bound of $f(n)$

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

When $f(n) < C \cdot g(n)$

$\forall \quad n \geq n_0$

$\forall \quad C > 0$



(III) Big Omega Notation (Ω)

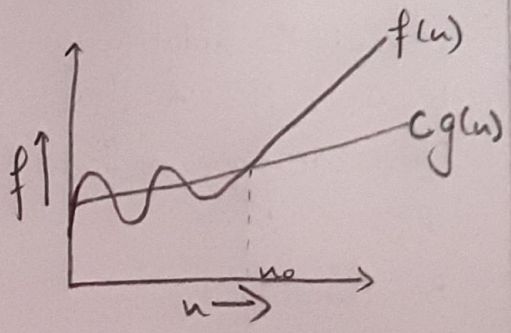
$f(n) = \Omega(g(n))$.

$g(n)$ is "tight" lower bound.

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

iff $f(n) \geq C g(n)$

$\forall \quad n \geq n_0$.



(IV) Small Omega Notation (ω)

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

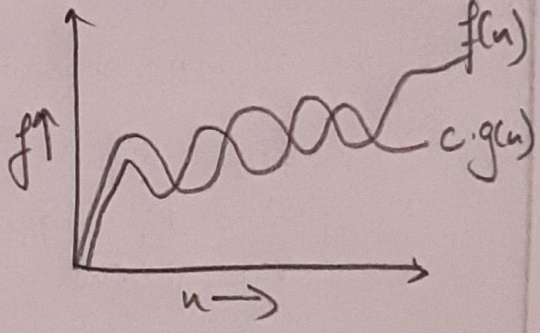
$g(n)$ is "lower" bound of $f(n)$

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

When, $f(n) > C \cdot g(n)$

$\forall \quad n > n_0$

$\forall \quad C > 0$



(V) Theta Notation (Θ)

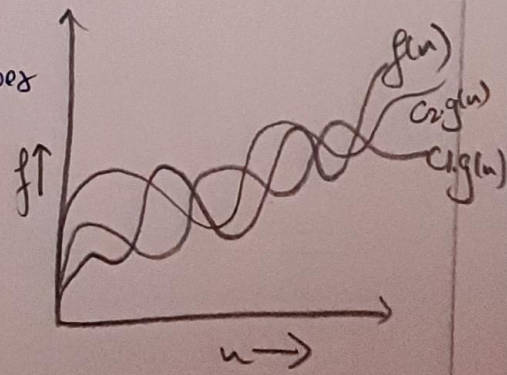
$f(n) = \Theta(g(n))$

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

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

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

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



Q2 What should be time complexity of:

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

⇒ `for(i=1 to n)` // i = 1, 2, 4, 8, ... n

`{ i=i*2}` // O(1)

⇒ $\sum_{i=1}^{} 1+2+4+8+ \ldots +n$

k^{th} term of G.P. ⇒ $T_k = ar^{k-1}$

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

$n = 2^{k-1}$ ⇒ $n = \frac{2^k}{2}$

$2^k = 2n$

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

$k = \log_2(2n)$

$k = \log_2 2 + \log_2 n$ ⇒ $k = 1 + \log_2 n$

~~$k = 1 + \log_2 n$~~ $O(\log_2 n)$

⇒ $O(n)$

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

⇒ $T(n) = 3T(n-1)$

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

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

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

$\vdots$

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

$= 3^n T(0)$

$= 3^n$ ⇒ $O(3^n)$

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

⇒ $T(n) = \{2T(n-1)-1\}$

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

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

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

$= 2^2 T(n-3) - 2^2 - 2^1 - 2^0$

$\vdots$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} - \ldots - 2^2 - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - \ldots - 2^2 - 2^1 - 2^0$

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

$T(n) = 1 \Rightarrow \underline{O(1)}$

Q5 What should be the time complexity of:

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

$\Rightarrow$ $i = 1, 2, 3, 4, 5, \ldots k$

$S = 1+2+3+4+5 \ldots k$

When $S \geq n$, then loop will stop at k^{th} iteration.

$\Rightarrow S \geq n$

$S = n$

$\Rightarrow 2 + 2+3+4+ \ldots +k = n$

$= 1 + (k*(k+1))/2 = n$

$= k^2 = n \Rightarrow k = \sqrt{n}$

$\Rightarrow \underline{O(\sqrt{n})}$

Q6 Time complexity of:

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

⇒ As $i^2 <= n$

$i <= \sqrt{n}$

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

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

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

⇒ $T(n) = O(n)$

Q7 Time complexity of:

```
void function (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++
}
```

⇒ For $k = k*2$

$k = 1, 2, 4, 8, \ldots n$ G.P. ⇒ $a=1$, $r=2$

$\frac{a(r^n-1)}{r-1}$ ⇒ $\frac{1(2^k-1)}{1}$

$n \Rightarrow 2^k$

$\log n = k$

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

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

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

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);
}
```

$\Rightarrow T(n) = T(n/3) + n^2$

$a=1, \quad b=3, \quad f(n)=n^2 \quad c = \log_3 1 = 0$

$n_0 = 1 \quad > (f(n)=n^2)$

$T(n) = \theta(n^2)$

Q9 Time complexity of:

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

⇒ for $i=1$ ⇒ $j = 1, 2, 3, 4, \ldots n$

for $i=2$ ⇒ $j = 1, 3, 5, 7, \ldots n$

for $i=3$ ⇒ $j = 1, 4, 7, \ldots n$

for $i=n$ ⇒ $j = 1 \ldots \ldots i$

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

⇒ $\sum_{j=n}^{1} n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}\right]$

⇒ $\sum_{j=n} n[\log n]$ ⇒ $T(n) = (n \log n)$

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

Q10 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 n_0 for which relation holds.

⇒ As given n^k and c^n

Relation between n^k and c^n is

$n^k = O(c^n)$ $\quad (n^k \leq c^n)$

$\forall\ n \geq n_0$ and some constant $a > 0$

for $n_0 = 1$, $c = a$

⇒ $1^k \leq a\, 2^1$

⇒ $n_0 = 1$ & $c = 2$.