Tutorial-1

① **Asymptotic Notations:** These are used to tell the complexity of any algorithm when the input size is very large. These notations tend towards infinity. Different types of asymptotic notations are:

(i) **Big Oh (O):**

A function 'f(n)' is said to be 'O(g(n))' iff

$$f(n) \leq c \cdot g(n) \quad \forall \; n \geq n_0 \;\&\; c > 0$$

g(n) is tight upper bound of f(n).

(ii) **Big Omega (Ω):**

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

iff $f(n) \geq c \cdot g(n) \quad \forall \; n \geq n_0 \;\&\; c > 0$

Here, g(n) is tight lower bound of f(n).

(iii) **Big theta (Θ):** (It gives both tight upper & lower bound)

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

if $c_1 g(n) \leq f(n) \leq c_2 \cdot g(n)$

$\forall \; n \geq \max(n_1, n_2), \; c_1 c_2 > 0$

(iv) **Small Oh (o):**

It gives upper bound.

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

if $f(n) < c \cdot g(n) \quad \forall \; n > n_0 \;\&\; c > 0$

(V) **Small Omega (ω):**

It gives lower bound.

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

if $f(n) > c \cdot g(n) \quad \forall \; n > n_0 \;\&\; c > 0$

(3) $T(n) = 3T(n-1)$ — (i) , $n > 0$

$T(1) = 1$

Putting $n = n-1$ in (i)

$T(n-1) = 3T(n-2)$ — (ii)

from (i) & (ii)

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

$T(n) = 3^2 T(n-2)$ — (iii)

Putting $n = n-2$ in (i)

$T(n-2) = 3T(n-3)$ — (iv)

from (iii) & (iv)

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

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

$\Rightarrow$ for $n = n-k$

$T(n) = 3^{k-1} [3T(n-k)]$

$T(n) = 3^k T(n-k)$ — (v)

if $n - k = 1$

$\Rightarrow$ $k = n-1$

$\Rightarrow T(n) = 3^{(n-1)} T(1)$

$\Rightarrow$

Time Complexity $= O(\frac{3^n}{3})$

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

(2)

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

$\boxed{TC = O(\log_2 n)}$

(4) $T(n) = 2T(n-1)$ — (I) $n>0$
$T(1) = 1$

Putting $n = n-1$ in (I) -
$T(n-1) = 2T(n-2)$ — (II)
from (I) & (II)
$T(n) = 2[2T(n-2)]$.
$T(n) = 2^2 T(n-2)$ — (III)

Putting $n=n-2$ in (I) -
$T(n-2) = 2T(n-3)$ (IV)
from (III) & (IV)
$T(n) = 2^2 [2T(n-3)]$
$T(n) = 2^3 T(n-3)$

Similarly, for $n = n-k$
$T(n) = 2^k T(n-k)$
if $n-k=1$
$\Rightarrow k = n-1$
$\Rightarrow T(n) = 2^{n-1} T(1)$
$\Rightarrow$ Time Complexity $= O(2^{n-1}) = O(2^n)$

(5) $\because$ $s = s+i$ and i starts from 1.
$\Rightarrow s = 1+2+3+4+ \dots + k$
$s = \dfrac{k(k+1)}{2} = \dfrac{k^2}{2} + \dfrac{k}{2}$
& $k^2 <= n$
$\Rightarrow k <= \sqrt{n}$
$\Rightarrow$ $\boxed{\text{Time Complexity} = O(\sqrt{n})}$

(7) for (i=n/2; i<=n; i++) $\Rightarrow \frac{n}{2}$ times

for (j=1; j<=n; j=j*2) $\Rightarrow \log_2 n$ times

for (k=1; k<=n; k=k*2) $\Rightarrow \log_2 n$ times

//O(1)

Solⁿ

$$\sum_{i=n/2}^{n}\left(\sum_{j=1\,(j=j*2)}^{n}\left(\sum_{k=1\,(k=k*2)}^{n} 1\right)\right)$$

$$= \sum_{i=n/2}^{n} \log_2 n \; \log_2 n$$

$$= \left(n-\frac{n}{2}+1\right)\left(\log_2 n \; \log_2 n\right) = n \log_2 n \log_2 n$$

$$\boxed{TC = O(n \log_2^2 n)}$$

(9) for (i=1 to n)

for (j=1; j<=n; j=j+i)

//O(1)

for i=1 , j runs n times

for i=2 , j runs n/2 times

for i=3 , j runs n/3 times

⋮

for i=n , j runs n/n times

$$\text{sum} = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + \frac{n}{n}$$

$$= n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right]$$

Since, above eq'n is in harmonic Progression, we will find its upper bound,

$$1 + \underbrace{\left(\frac{1}{2} + \frac{1}{3}\right)}_{\text{I}} + \underbrace{\left(\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7}\right)}_{\text{II}} + \cdots$$

upper bound of I $= \frac{1}{2}$

" " " II $= \frac{1}{4}$

$\Rightarrow$ replacing the elements with their upper bounds respectively.

$\Rightarrow$ Sum $= 1 + \left(\frac{1}{2} + \frac{1}{2}\right) + \left(\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}\right) + \cdots$

Sum $= 1 + 1 + 1 + \cdots$ log (n) times

[as for every 2^k terms, sum is only 1)

$\Rightarrow$ Sum $= n \log n$

$\Rightarrow \boxed{TC = O(n \log n)}$

⑥
```
for (i=1; i*i<=n; i++)
    count++;
```

$i*i = n$

$i = \sqrt{n}$

$\Rightarrow \boxed{\text{Time Complexity } = O(\sqrt{n})}$

⑧ function (int n){
if (n==1) return; //O(1)
for(i=1 to n){ //O(n)
for(j=1 to n){ // O(n)
//O(1)
}
}
function (n-3); // O($\log_2 n$)
}

T.C $= O(n^2 + \log_2 n)$

$\boxed{TC = O(n^2)}$

⑩ let $f(n) = n^k$ $\quad k \geq 1$

$g(n) = c^n$ $\quad c > 1$

if $n=1$, $f(n) = 1$

$g(n) = c$

for $c > 1$, $g(n) \geq f(n)$

$\Rightarrow 0 \leq f(n) \leq g(n)$ for $c > 1$ & $n \geq 1$

$\because$ $g(n)$ is tight upper bound of $f(n)$

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

or $\boxed{n^k = O(c^n)}$ where $c \geq 1$ & $n_0 = 1$