PAGE _____
DATE _____

Design and Analysis of Algorithm

TUTORIAL : 1

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

A1. Asymptotic Notation is used to describe the running time of an algorithm - how much time an algorithm takes with a given input n. There are three different notations : Big O, Big Theta and Big Omega.

* **Big-O** : Function's complexity will not cross the growth of the asymptotic notation in any case.

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

Ex : $f(n) = 3\log n + 100$
$g(n) = \log n$.

* **Big-Omega** : For the best case, or a floor growth rate for a given function

$f(n) = \Omega(g(n))$ implies that $g(n)$ is tight lower bound of $f(n)$.

PAGE
DATE

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

if $f(n) = 4n+3$
$g(n) = n$.

* Theta : Theta gives us tight upper and lower bound both.

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

then $3n^3+6n^2+6000 = \Theta(n^3)$.

Q2. T.C. of : for(i=1 to n) { i = i * 2 };

A2. The loop will run as : 1, 2, 4, 8, ... uptil n

thus, $2^k = n$

$\rightarrow k = \log_2 n$.

Thus, time complexity = $O(\log n)$. Ans.

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

By forward substitution,

Given : $T(0) = 1$.

PAGE
DATE

& $T(1) = 3T(0) = 3$
$T(2) = 3T(1) = 3^1$
$T(3) = 3T(2) = 3^2$
⋮
⋮
$T(n) = 3T(n-1) = 3^{n-1}$

Thus, T.C. of $T(n) = 3T(n-1)$ will be
$O(3^{n-1})$
$= O(3^n)$. Ans.

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

A4. Given, $T(0) = 1$.

By forward substitution,

$T(1) = 2T(0)-1 = 1$
$T(2) = 2T(1)-1 = 1$
⋮

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

Thus, T.C. of $T(n)$ will be $O(1)$. Ans.

PAGE
DATE

Q5. What should be T.C. of :

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

Ans. printf("#") ⟶ O(1).

ATQ : The loop will run as follows –

i=1, s=1
i=2, s=3
i=3, s=6
i=4, s=10
i=5, s=15
... upto n

$\Rightarrow 1+3+6+10+15+\cdots+=n$

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

$\Rightarrow k^2+k = 2n$

$\Rightarrow k^2 = 2n-k$ (Ignoring constants)
$= 2n$

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

∴ Total steps = $\sqrt{n}$

Thus, Time Complexity : $O(\sqrt{n})$ Ans.

6. Time complexity of -

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

Ans 6. Acc to question, i will run till $\sqrt{n}$.

$\because i^2 <= n$

and count++ $\rightarrow O(1)$

$\therefore$ Time complexity = $O(\sqrt{n})$ Ans.

7. T.C. 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++
}
```

Q7. `for(i=n/2; i<=n; i++)` $\longrightarrow O(n/2)$

`for(j=1; j<=n; j*2)` $\longrightarrow O(\log_2 n)$

`for(k=1; k<=n; k*2)` $\longrightarrow O(\log_2 n)$

thus total time : $O(\frac{n}{2}) \times O(\log_2 n) \times O(\log_2 n)$

$= O(n \times \log_2 n \times \log_2 n)$

$= O(n \log^2 n)$. Ans

Q8. T.C. of –

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

Q8. ATQ : $n-3-3-3 \cdots$ upto 1

$\Rightarrow n-3k = 1$

$\Rightarrow k = \left(\frac{n-1}{3}\right)$

for each k, "*" will be printed ~~n×n times~~ $(n-k)$* $(n-k)$ times.

$$\therefore \text{T.C.} = \left(\frac{n-1}{3}\right)(n-k)(n-k)$$

$$= \left(\frac{n-1}{3}\right)\left(n-\left(\frac{n-1}{3}\right)\right)\left(n-\left(\frac{n-1}{3}\right)\right)$$

$$= \left(\frac{n-1}{3}\right)\left(\frac{2n+1}{3}\right)^2$$

$$= \left(\frac{n-1}{3}\right)\left(\frac{4n^2+4n+1}{9}\right)$$

$$= \frac{4n^3+4n^2+n-4n^2-4n-1}{27}$$

Time Compl. = $O(n^3)$ { ∵ all constants can be ignored & retaining the highest order term }.

9. Time Complexity of —

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

Ans. printf("*") → O(1)

for(j=1; j<=n; j=j+i) → $O(\sqrt{n})$

for(i=1 to n) $\longrightarrow O(n)$

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

10. For the functions, n^k and a^n, what is the asymptotic relationship between these functions? Assume that $k>=1$ and $a>1$ are constants. Find out the value of c and n_0 for which relation holds.

Ans. Relation : $n^k = O(a^n)$.