# DESIGN AND ANALYSIS OF ALGORITHMS

## TUTORIAL-1

Name - Srijan
Section - F
Roll No. - 30
Univ. Roll No. - 2017067

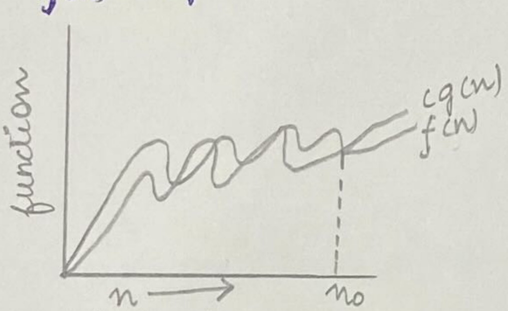
1 Ques →

Solution → Asymptotic Notations → They are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

Different asymptotic notation -

(i) Big O (n)

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




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

iff $f(n) \leq cg(n)$

$\forall\ n \geq n_0$

for some constant, $c > 0$

- $g(n)$ is "tight" upper bound of $f(n)$.

ex. $f(n) = n^2 + n$

$g(n) = n^3$

$n^2 + n \leq cn^3$

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

(ii) Big omega ($\Omega$)

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

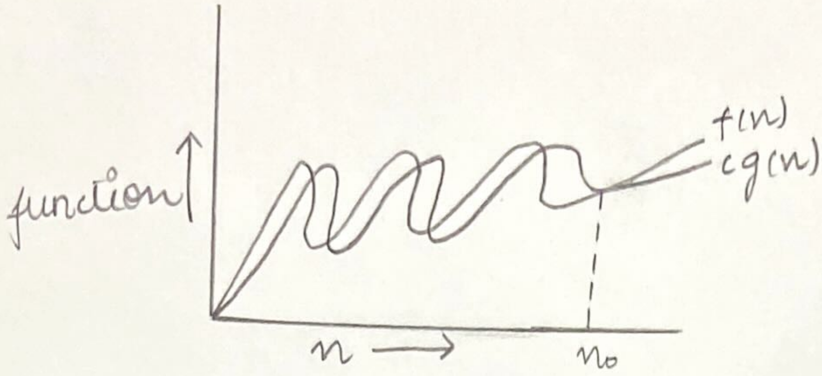
$g(n)$ is "tight" lower bound of function $f(n)$.

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

iff $f(n) \geq cg(n)$

$\forall\ n \geq n_0$

for some constant $c > 0$




ex.

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

(iii) Big Theta ($\theta$)

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

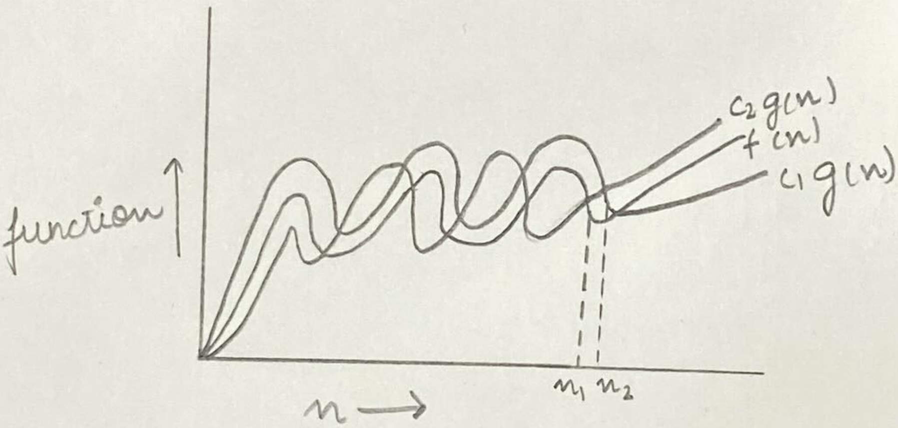
g(n) is both "tight" upper and lower bound of function f(n).

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

iff

$$c_1 g(n) \le f(n) \le c_2 g(n)$$
$$\forall\, n \ge \max(n_1, n_2)$$

for some constant $c_1 > 0$ and $c_2 > 0$




ex.

$3n+2 = \theta(n)$ as $3n+2 \ge 3n$ and
$3n+2 \le 4(n)$ for n, $k_1 = 3$, $k_2 = 4$ & $n_0 = 2$

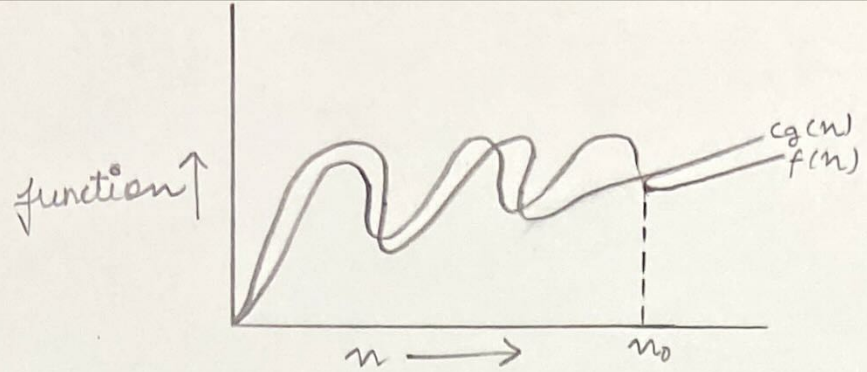
(iv) Small $\theta(\theta)$ -

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

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

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

when $f(n) < cg(n)$
$\forall\, n > n_0$
and $\forall$ constants, $c > 0$




Ex -

$f(n) = n^2$

$g(n) = n^3$

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

(V) Small omega ($\omega$)

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

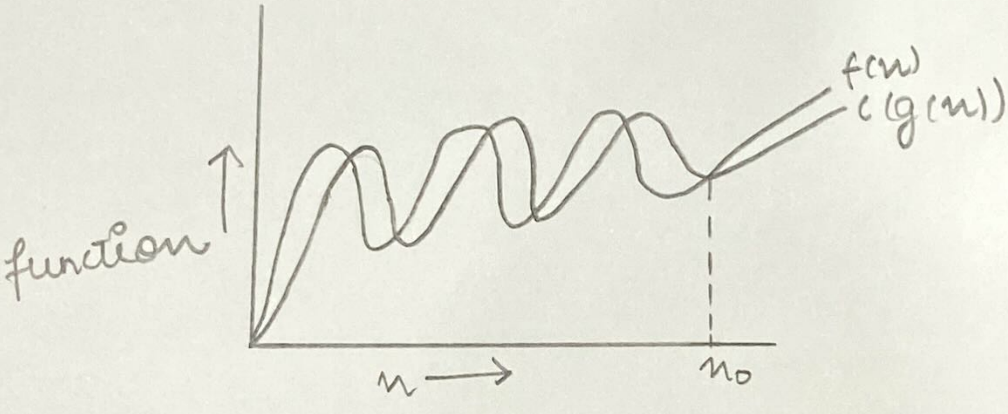
g(n) is lower bound of f(n).

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

when $f(n) > c\,g(n)$

$\forall\ n > n_0$

and $\forall$ constants, $c > 0$




$f(n) = 4n + 6$ $\quad g(n) = (1)$

Ques →

Solution → for ($i = 1$ to $n$)

{ $i = i * 2$ }

→ $i = \underbrace{1, 2, 4, 8, 16, \text{-----}, n}_{k}$ (G.P.)

— $O(k)$

$a = 1$, $r = \frac{2}{1} = 2$.

GP $k^{th}$ value $= t_k = ar^{k-1}$

$n = 1 \times 2^{k-1}$

$n = \frac{2^k}{2}$

$2n = 2^k$

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

$k = \log_2 2n$

$k = \log_2 2 + \log_2 n$

$k = 1 + \log_2 n$

Time comp $= O(1 + \log_2 n)$

$= O(\log_2 n)$

3Ques →

Solution → $T(n) = 3T(n-1)$ — ①

Let $n = n-1$

$T(n-1) = 3T(n-2)$ — ②

Put ② in ①

$T(n) = 3 \times 3T(n-2)$ — ③

Put $n = n-2$

$T(n-2) = 3T(n-3)$ — ④

Put ④ in ③

$T(n) = 3 \times 3 \times 3T(n-3)$ — ⑤

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

$= 3^n T(0)$

$= 3^n$

$= O(3^n)$

4Ques →

Solution → $T(n) = 2T(n-1) - 1$

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

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

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

$- - - - -$

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

$- - - 2^2 - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} - - - - 2^2 - 2^1 - 2^0$

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

$T(n) = 1$

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

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} \cdots 2^2 - 2^1 - 2^0$

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

$T(n) = 1$

**5 Ques →**

**Solution →**

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

$S_i = S_{i-1} + i$

i is incrementing by one step

s is incrementing by value of i

Following will be values after few iterations –

⇒ $i=2, s=3$ 1st iteration

⇒ $i=3, s=6$ 2nd iteration

⇒ $i=4, s=10$ 3rd iteration

Let the value of n be k.

Values of S ⇒ 1, 3, 6, 10 ....

S represents a series of sum of first n natural numbers

for $i=k$, $S = \frac{k(k+1)}{2}$

for stopping loop.

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

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

**6 Ques →**

**Solution →**

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

$i = 1, 2, 3 \cdots n$

$i^2 = 1, 4, 9 \cdots n$

so $i^2 < n$ or $i <= \sqrt{n}$

$a_k = a + (k-1)d.$

$a = 1 \qquad d = 1$

$a_k <= \sqrt{n}$

$\sqrt{n} = 1 + (k-1).1$

$\sqrt{n} = k$

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

7 Ques →

Solution →

```
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 ++;
 }
```

$i = n/2 \qquad j = \log_2 n \qquad k = \log_2 n$

$(\frac{n}{2}+1)$ times $\qquad \log_2 n \qquad \log_2 n$

$O(i * j * k) = O((\frac{n}{2}+1) * \log_2 n + \log_2 n)$

$= O((\frac{n}{2}+1) \times (\log n)^2)$

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

8 Ques →

Solution →

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

$T(n) = T(n-3) - n^2$ — (1)

$T(1) = 1$ — (2)

put $n = n-3$ in 1

$T(n-3) = T(n-6) + (n-3)^2$ — 3

Put 3 in 1

$T(n) = T(n-6) + (n-3)^2 + n^2$ — 4

Put $n = n-6$ in 1

$T(n-6) = T(n-9) + (n-6)^2$ — 5

Put 5 in 4

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

Generalising

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

Let $n - 3K = 1$

$\frac{n-1}{3} = K$

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

$T(n) = T(1) + [n - ((n-1) - 3]^2 + [n - (n-1-6)]^2 + (n - (n-1-9))^2 + \dots n^2$

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

$T(n) = 1^2 + 4^2 + 7^2 \dots n^2$

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

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

9 ques →

Solution →

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

for $i = 1$, $j \to n$ times

for $i = 2$, $j = 1 + 3 + 5 + \dots + n$

$a_n = a + (K-1)d$

$a = 1 \quad d = 2$

$n = 1 + (K-1) \times 2$

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

$$K = \frac{n-1}{2}+1$$

$$\boxed{K = \frac{n+1}{2}}$$ No. of terms

for $i=2$, $j \to \frac{n+1}{2}$ times

for $i=3$, $j = 1+4+7+\ldots\ldots n$

$$n = 1+(K-1)\times 3$$

$$\boxed{\frac{n-1}{3}+1 = K}$$

for $i=3$, $j = \frac{n+2}{3}$ times

Generalising

for $i=n$, $j = \frac{n+K-1}{K}$ times

Time complexity is

$$\underbrace{n + \frac{n+1}{2} + \frac{n+2}{3} + \ldots\ldots + \frac{n+K-1}{K}}_{n \text{ terms}}$$

General term $= \frac{n+K-1}{K}$

$$\sum_{K=1}^{n} \frac{n+K-1}{K} = \frac{\sum_{1}^{n} n + \sum_{1}^{n} K - \sum 1}{K}$$

$$\Rightarrow \frac{\frac{n(n+1)}{2} + nK - n}{K}$$

$$\Rightarrow \frac{n^2 + \frac{n}{2} + nK - n}{K}$$

$$T(n) = \frac{n^2 + \frac{n}{2} + nK - n}{K}$$

Neglecting constant terms

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

10 Ques →

Solution →

as given $n^k$ & $c^n$

relation b/w $n^k$ & $c^n$ is

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

as $n^k \leq a c^n$

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

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a 2$

$n_0 = 1$ & $c = 2$