Date __ / __ / ____

Saathi

## TUTORIAL-1

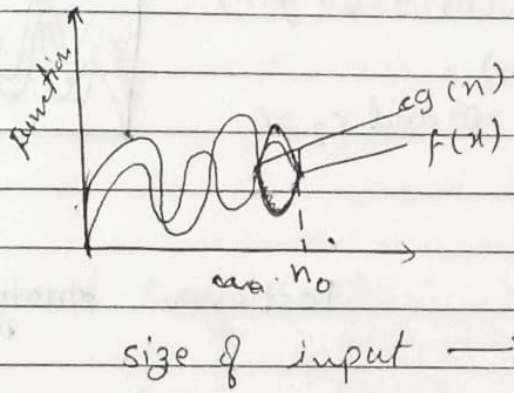
1. **Asymptotic Notation**

Tending to infinity

→ Helps to find the complexity of an algorithm when input is very large.

- Big O

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




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

iff

$f(n) \le c\,g(n)$

$\forall\ n \ge n_0$

for some constant, $c > 0$

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

- Big Omega ($\Omega$)

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

f(n) will not be lower than g(n) after $n_0$

g(n) is '<u>tight' lower</u> bound of function f(n)

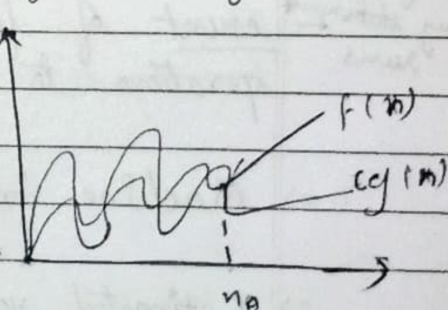
$f(n) = \Omega(g(n))$

iff

$f(n) \ge c\,g(n)$

$\forall\ n \ge n_0$

for some constant, $c > 0$



- Theta ($\theta$)

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

g(n) is both 'tight' upper and lower bound of f(x)

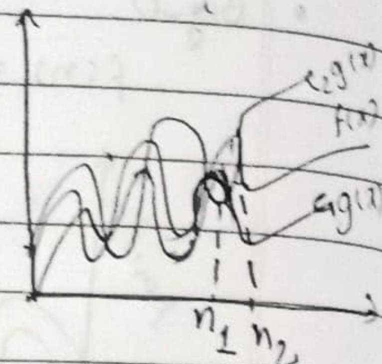
$$f(n) = \theta(g(x))$$

iff

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

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

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



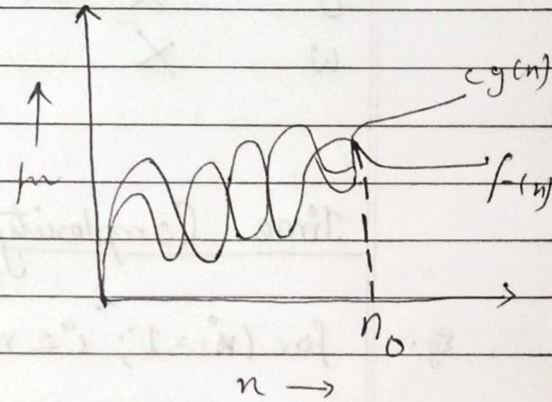
- Small o (o)

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

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

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

$f(n) < c\,g(n)$
$\forall \; n > n_0$
$c > 0$



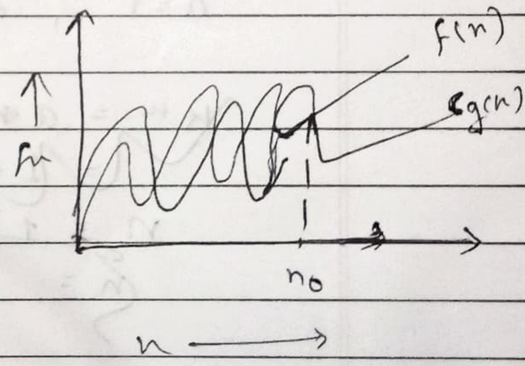

- Small Omega ($\omega$)

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

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

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

$f(n) > c\,g(n)$
$\forall \; n > n_0$
$c > 0$



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

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

$i = 2^0, 2^1, 2^2, 2^3, \ldots, 2^k$

$\Sigma = 1+2+4+8+\ldots n$

$a = 1, \quad r = 2, \quad n = k+1$

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

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

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

$$\cancel{\log n = k+1 \frac{\log 2}{\log 1}}$$

$$n+1 = 2^{k+1}$$

$$\log n+1 = (k+1)\log_2$$

$$\log n = k$$

$$\boxed{O(k) = O(\log n)}$$

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

$T(n) = 3T(n-1)$ — ①

put $n = n-1$ in ①

$T(n-1) = 3T(n-2)$ — ②

put $T(n-1)$ to 1

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

$T(n) = 9T(n-2)$ — ③

put $n = n-2$ in ①

$T(n-2) = 3T(n-3)$ — ④

put $T(n-2)$ in ③

$T(n) = 9[3T(n-3)]$

$T(n) = 27\,T(n-3)$ — ⑤

From, ①, ②, ③ & ④

$T(n) = 3^k T(n-k)$ — ⑥

put $n-k=0$

$n = k$

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

$T(n) = 3^k T(0)$

$T(n) = 3^k$

and $k = n$

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

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

$T(n) = 2T(n-1)-1$ — ①

put $n = n-1$ in ①

$T(n-1) = 2T(n-2)-1$ — ②

put $T(n-1)$ in ①

Saathi

Date __/__/__

$T(n) = 2[2T(n-2)-1]-1$ —— (3)

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

$T(n-2) = 2T(n-3) - 1$ —— (4)

put $T(n-2)$ in (3)

$T(n) = 4[2T(n-3)-1]-2-1$

$T(n) = 8T(n-3)-4-2-1$ —— (5)

From (1), (3) & (5)

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - \dots - 1$ —— (6)

$a = 2^{k-1}$ , $r = \frac{1}{2}$ , $n = k$

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

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

$T(n) = 2^k - 1$ —— (7)

from (6)

$n-k = 0$

$n = k$

From (6) & (7)

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

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

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

$\boxed{T(n) = O(1)}$

Date ___ / ___ / ___

5.
```
int i=1, s=1;
while(s<=n){
    i++; s=s+i;
    printf("#");
}
```

| | $i$ | $s$ |
|---|---|---|
| Initially, | 1 | 1 |
| After iteration, | 2 | 3 |
| | 3 | 6 |
| | 4 | 10 |
| | 5 | 15 |
| | ⋮ | ⋮ |
| | $k$ | $\frac{k(k+1)}{2} > n$ |

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

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

$$2n = k^2$$

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

Therefore,

$$O(k) = O(\sqrt{2}\cdot\sqrt{n})$$

$$\boxed{O(k) = O(\sqrt{n})}$$

Date ___/___/______

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

Given, $i^2 <= n$

$i <= \sqrt{n}$

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

$T(n) = 1+2+3+\cdots+\sqrt{n}$

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

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

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

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

Date ___/___/___

Outer loop (i): $n/2, n/2+1, n/2+2, \dots, n$

$a = n/2, \; d = 1, \; n = n/2$

$T(n) = a + (n-1)d$

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

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

$T(n) = \frac{2n-2}{2}$

$T(n) = n-1$

$\boxed{T(n) = n}$ — 1

Inner loop (j, k): 1, 2, 4, 8, ......, k

$a = 1, \; r = 2, \; n = k$

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

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

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

$n + 1 = 2^k$

$\log n + 1 = k \log 2$

$\boxed{O(k) = O(\log n)}$ — 2

Date ___/___/___

From ① · ⑧ ②

| $i$ | $j$ | $k$ |
|---|---|---|
| $n/2$ | $\log n$ | $\log n * \log n$ |
| $n/2 + 1$ | $\log n$ | $\log n * \log n$ |
| ⋮ | ⋮ | ⋮ |
| $n$ | $\log n$ | $\log^n * \log n$ |

$\therefore T(n) = n * \log n * \log n$

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

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

for i & j loop: 1, 2, 3, ..., n

$\therefore T(n) = O(n)$ — ①

Page No.

Date __/__/__

for recursive call,

$\Rightarrow T(n/3)$ - ②

From ① & ②

$$T(n) = T(n/3) + n*n$$

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

$c = \log_b a$
$= \log_3(1) = 0$

$n^0 = 1 > f(n)$

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

9. void function(int n){
   for(i=1 to n){
      for(j=1; j<=n; j=j+i){
         print("*");
      }
   }
}

| | $i$ | $j$ |
|---|---|---|
| Initially: | 1 | 1 |
| After iteration: | 2 | 2 |
| | 3 | 5 |
| | 4 | 9 |
| | ⋮ | ⋮ |
| | n | |

Date __/__/__

at $i=1: \; j=1,2,3,\dots,n$ $\qquad O(n)$

$i=2: \; j=1,3,5,7,\dots,n$ $\qquad O(n/2)$

$i=3: \; j=1,4,7,10,\dots,n$ $\qquad O(n/3)$

$\vdots$ $\qquad\qquad \vdots$

$i=n: \; j=1$ $\qquad O(1)$

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

$$T(n) = n\left[1 + 1/2 + 1/3 + \dots + 1/n\right]$$

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

$$\boxed{T(n) = O(n \log n)}$$

10. function, $n^k$ and $c^n$

Given, $k >= 1$ and $c > 1$

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

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

as $n^k \le a c^n$ $\quad \forall n \ge n_0$ where $[a > 0]$

for, $n_0 = 1$

$c = 2$

$$1^k \le 2a$$

$\therefore$ $\boxed{n_0 = 1}$ and $\boxed{c = 2}$