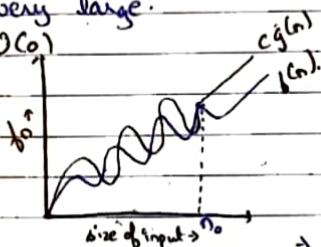


# Asymptotic Notations

→ Tending to infinity

They help you find the complexity an algorithm when input is very large.

## i) Big O ( $O$ )



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

$$f(n) \leq c \cdot g(n)$$

$$\forall n \geq n_0$$

for some constant  $c > 0$

→  $g(n)$  is 'tight' upper bound of  $f(n)$

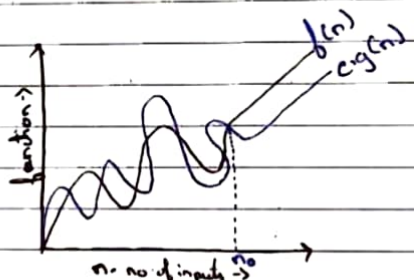
## ii) Big Omega ( $\Omega$ )

$f(n) = \Omega(g(n))$   
 $g(n)$  is 'tight' lower bound of  $f(n)$

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

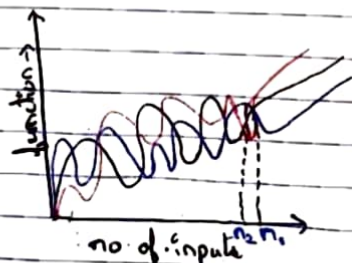
$$f(n) \geq c \cdot g(n)$$

$\forall n \geq n_0$  for some constant  $c > 0$



## iii) Theta ( $\Theta$ )

$f(n) = \Theta(g(n))$   
 $g(n)$  is both 'tight' upper & lower bound of  $f(n)$   
 $f(n) = \Theta(g(n))$



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

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

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

## small o ( $o$ )

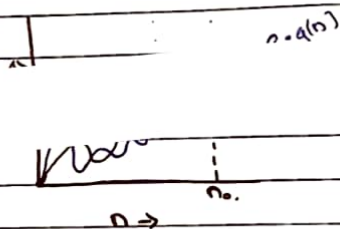
$f(n) = o(g(n))$   
 $g(n)$  is upper bound

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

when  $f(n) < c \cdot g(n)$

$\forall n > n_0$

$\Delta \forall c > 0$



## small omega ( $\omega$ )

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

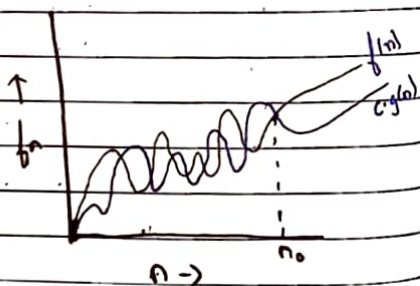
$g(n)$  is lower bound of  $f(n)$

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

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

$\forall n > n_0$

$\Delta \forall c > 0$



Ques 2 What should be time complexity of  
for( $i=1$  to  $n$ ) { $i=i+2$ }

for( $i=1$  to  $n$ ) //  $i=1, 2, 4, 8, \dots, n$   
{ $i=i+2$ } //  $O(1)$

$$\Rightarrow \sum_{i=1}^n 1+2+4+8+\dots+n$$

GP kth value  $\Rightarrow T_k = ar^{k-1}$   
 $\Rightarrow 1 \times 2^{k-1}$

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

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

$$\Rightarrow \log_2 2n = k \log_2 2$$

$$\Rightarrow \log_2 + \log_2 n = k \log_2 2$$

$$\Rightarrow \log_2 n = k$$

$$\Rightarrow O(k) = O(1 + \log_2 n)$$
  
$$= \underline{O(\log_2 n)}$$

Q-3  $T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0 \\ \text{otherwise } 1 \end{cases}$

$$T(n) = 3T(n-1) \quad \text{--- (1)}$$

~~Recursion~~

put  $n = n-1$

$$T(n-1) = 3T(n-2) \quad \text{--- (2)}$$

from 1 & 2

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

$$= 9T(n-2) \quad \text{--- (3)}$$

putting  $n = n-2$  in ①

$$T(n) = 3(T(n-2)) \quad - \textcircled{4}$$

$$\Rightarrow T(n) = 27(T(n-3))$$

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

putting  $n-k=0$

$$\Rightarrow n=k$$

$$\Rightarrow T(n) = 3^n [T(0-0)]$$

$$\Rightarrow T(n) = 3^n T(0)$$

$$\Rightarrow T(n) = 3^n \times 1 \quad [T(0)=1]$$

$$\Rightarrow \underline{\underline{T(n) = O(3^n)}}$$

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

$$T(n) = 2T(n-1) - 1 \quad - \textcircled{1}$$

Let  $n = n-1$

$$\Rightarrow T(n-1) = 2T(n-2) - 1 \quad - \textcircled{2}$$

$\Rightarrow$  from ① & ②

$$\Rightarrow T(n) = 2[2T(n-2) - 1] - 1$$

$$\Rightarrow T(n) = 4T(n-2) - 2 - 1 \quad - \textcircled{3}$$

Let  $n = n-2$

$$\Rightarrow T(n-2) = 2T(n-3) - 1 \quad - \textcircled{4}$$

from ③ & ④

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

$$\Rightarrow T(n) = 8T(n-3) - 4 - 2 - 1$$



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

$$\Rightarrow GP = 2^{k-1} + 2^{k-2} + 2^{k-3} + \dots$$

$$a = 2^{k-1}$$

$$r = 1/2$$

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

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

$$= 2^k (1 - (1/2)^k)$$

$$= 2^k - 1$$

$$\text{Let } n-k = 0$$

$$\Rightarrow n = k$$

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

$$\Rightarrow T(n) = 2^n \cdot 1 - (2^n - 1)$$

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

$$\Rightarrow T(n) = O(1)$$

25-5 what should be time complexity of

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

i: 1 2 3 4 5 6 ... n  
s: 1 2 3 6 10 15 21 ... n

Sum of  $s = 1 + 3 + 6 + 10 + \dots + T_n$  - (1)  
also  $s = 1 + 3 + 6 + 10 + \dots + T_{n-1} + T_n$  - (2)  
from (1) - (2)

$$0 = 1 + 2 + 3 + 4 + \dots + n - T_n$$

$$\Rightarrow T_n = 1 + 2 + 3 + 4 + \dots + n$$

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

~~$\Rightarrow T_n = \frac{1}{2} n(n+1)$~~

$\Rightarrow$  for  $k$  iterations.

$$1 + 2 + 3 + \dots + k \leq n$$

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

$$\Rightarrow \frac{k^2 + k}{2} \leq n$$

$$\Rightarrow O(k^2) \leq n$$

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

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

Ques-5 Time complexity of -

```
void fn(int n)
```

```
{ int i, count = 0;
```

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

```
    count++
```

// OCS

```
}
```

as  $i^2 \leq n$   
 $\Rightarrow i \leq \sqrt{n}$

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

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

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

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

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

Ques-7 Time complexity of :-

```
void fn(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  $k = 1, 2$

$k = 1, 2, 4, 8, \dots, n$

$\Rightarrow$  G.P.  $a=1, r=2$

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

$$n \Rightarrow 2^k$$

$$\Rightarrow \underline{\log n = k}$$

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

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

$$\Rightarrow \underline{O(n \log^2 n)}$$



8) Time Complexity of

```
function f(int n)
{ int m=1;
```

```
  return;
```

```
  for (i=1 to n)
```

```
    for (j=1 to n)
      print('*');
```

```
    }
```

```
  }
```

```
  function f(n-3);
```

```
}
```

// O(1)

// i = 1, 2, 3, 4, ..., n  $\Rightarrow O(n)$

// j = 1, 2, 3, 4, ..., n  $\Rightarrow O(n^2)$

$T(n/3)$

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

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

$$c = \log_3 1 = 0$$

$$\Rightarrow n^0 = 1 > (f(n) = n^2)$$

$$\Rightarrow T(n) = \Theta(n^2)$$



Ques 9

Time complexity of -  
void function (int n)

{ for (i=1 to n)

// O(n)

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

print ("\*")

// O(n)

}

}

for i=1  $\Rightarrow j = 1, 2, 3, 4, \dots, n = n$   
for i=2  $\Rightarrow j = 1, 3, 5, \dots, n = n/2$   
for i=3  $\Rightarrow j = 1, 4, 7, \dots, n = n/3$   
:  
:  
for i=n  $\Rightarrow j = 1, \dots, 1$

$$\Rightarrow \sum_{j=1}^n n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \dots + 1$$

$$\Rightarrow \sum_{j=1}^n n \left[ 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} \right]$$
$$\sum_{j=1}^n n [\log n]$$

$$\Rightarrow T(n) = [n \log n]$$

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

Q-10 for functions,  $n^k$  &  $c^n$ , what is the asymptotic relation between these functions?

assume that  $k \geq 1$ , &  $c > 1$  are constant.

Find out the value of  $c$  &  $n_0$  for which relation holds

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

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

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

as  $n^k \leq ac^n$   
 $\forall n \geq n_0$  & some constant ~~where~~  
 $a > 0$

$$\text{for } n_0 = 1 \\ c = 2$$

$$\Rightarrow 1^k \leq a \cdot 2^1$$

$$\Rightarrow \underline{n_0 = 1} \text{ & } c = 2$$