TUTORIAL 1

NAME → Sajal Mittal.
SEC → CST
SEMESTER → IV
R. NO. → 16.
UNI ROLL NO. → 2017540.
DATE → 10 - March - 2022.

Sajal

Design And Analysis of Algorithms.

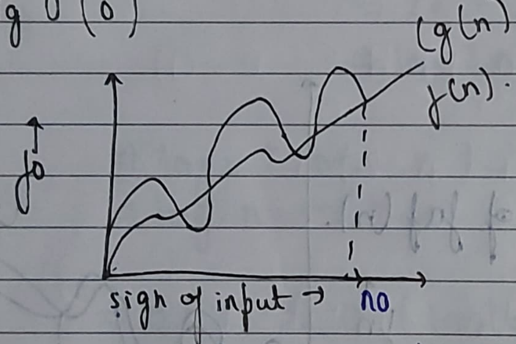
Tutorial-1

Ques 1:

Asymptotic Notations: Asymptotic means tending to infinity. They help you find the complexity of an algorithm when input is very large.

Different asymptotic notations with examples are:-

1. Big O (o)

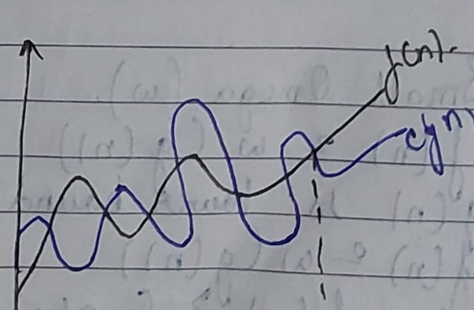


$f(n) = O(g(n))$

iff $f(n) \le c\,g(n)$
$\forall\ n \ge n_0$
for some constant $c > 0$

⇒ g(o) is tight upper bound of f(n)

2. Big Omega (Ω).

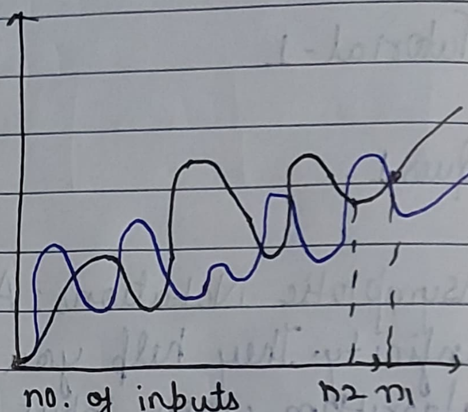


$f(n) = \Omega(g(n))$
g(n) is 'tight' lower bound of f(n)
$f(n) = \Omega(g(n))$
iff $f(n) \ge c\,g(n)$
$\forall\ n \ge n_0$ for some constant $c > 0$.

3. Theta (θ).

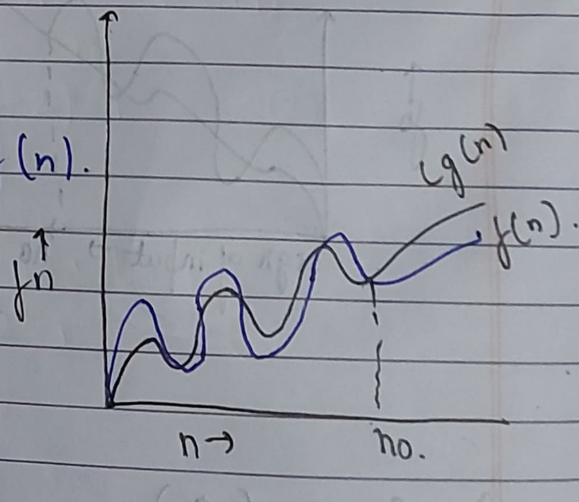
$f(n) = \theta(g(n))$

g(n) is both 'tight' upper & lower bound of fn of f(n).
$f(n) = \theta(g(n))$
iff.
$c_1 g(n) \leq f(n) \leq c_2 . g(n)$
$\forall n \geq \max(n_1, n_2)$
for some constant $c_1 > 0$ & $c_2 > 0$

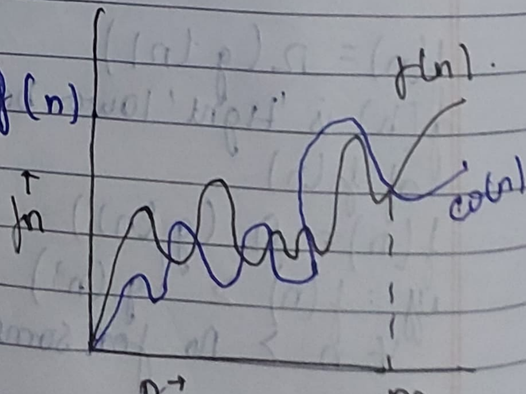


4. Small o(o)

$f(n) = o(g(n))$
g(n) is upper bound of fn f(n).
$f(n) = o(g(n))$
when $f(n) < c . g(n)$
$\forall n > n_0$
& $\forall c > 0$



5. Small Omega (ω).
$f(n) = \omega(g(n))$
g(n) is lower bound of fn f(n)
$f(n) = \omega(g(n))$.
when $f(n) > c . g(n)$.
$\forall n > n_0$.
& $\forall c > 0$.



Date
Page

Question 2: What should be time comple. of for. (i = 1 to n) (i = i x 2).

for (i = 1 to n) " i = 1, 2, 4, 8 - - - - n.
{i = i x 2} " o(1).

$\Rightarrow \{1+2+4+8+ - - - - - n\}$.
i = 1

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

$\Rightarrow 1 \times 2^{k-1}$

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

$\Rightarrow 2n = 2^k$

$\Rightarrow \log 2n = k \log 2$

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

$\Rightarrow \log n + 1 = k$

$\Rightarrow O(k) = O(1 + \log n)$

$= O(\log n)$

Question 3: T(n) = {3T (n - 1) if n > 0 otherwise 1}.

$T(n) = 3T(n-1)$ — ①.

put $n = n-1$

$T(n-1) = 3T(n-2)$ — ②.

from ① & ②.

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

$= 3T(n-2)$ — ③.

Putting $n = n-2$ in ①.

Date
Page

$T(n) = 3(T(n-3))$ — 4

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

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

putting $n-K=0$

$\Rightarrow n = K$

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

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

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

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

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

$T(n) = 2T(n-1) - 1$ — 1

Let $n = n-1$

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

from 1 & 2

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

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

Let $n = n-2$

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

from 3 & 4

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

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

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

Sajal

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

$a = 2^{k-1}$

$k = 1/2$

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

$= \frac{2^{x-1}(1-(12)^n)}{1/2}$

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

$= 2^k - 1$

let $n - k = 0$

$\Rightarrow n = k$

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

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

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

Question 5. What should be time complexity of

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

Date
Page

$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \; - - -$

$s = 1 + 3 + 6 + 10 + 15 + 21 - - - n$

Sum of $s = 1 + 3 + 6 + 10 + \; - - - - + T_n$ — (1)

also $s = 1 + 3 + 6 + 10 + \; - - - + T_n$ — (2)

from (1) & (2).

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

$\Rightarrow T_K = 1 + 2 + 3 + 4 + - - - K$

$\Rightarrow T_K = \frac{1}{2} K (K + 1)$

$\Rightarrow$ for K interations

$1 + 2 + 3 + - - - - + K <= n$

$\Rightarrow \; \frac{K(K+1)}{2} <= n$

$\Rightarrow \frac{K^2 + K}{2} <= n$

$\Rightarrow O(K^2) <= n$

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

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

Question 6: Time complexity of -

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

?

as $i^2 <= n$

$\Rightarrow i <= \sqrt{n}$

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

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

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

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

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

Question 7: Time complexity of -

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

  for (i=1; j <= n ; j = j x2)
  for ( k = 1; k <= n ; k = k x 2 )
      count ++;
```

for $k = k \times 2$

$k = 1, 2, 4, 8 ------ n$

$\Rightarrow GP \rightarrow a = 1, r = 2$

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

$n \Rightarrow 2^k$

$\Rightarrow \log n \rightarrow k.$

Date
Page

⇒ i	j	k.
1	log n	log n * log n
2	log n	log n * log n.
⋮	⋮	⋮
n	log n	log n * log n.

⇒ $O(n * \log n * \log n)$.
⇒ $O(n \log^2 n)$.

Question 8 : Time complexity of

```
function (int n).
{  int (n = = 1)
   1 , return;

   for (i = 1 to n)          // i = 1, 2, 3, 4 ... n ⇒ O(n)
   { for (j = 1 to K)        // j = 1, 2, 3, 4, - - - n ⇒ O(n^2).
      {
        print ("*");
      }
   }.
   function (n-3);   T(n/3).
```

⇒ $T(n) = T(n/3) + n^2$
⇒ $a = 1$, $b = 3$, $f(n) = n^2$

$c = \log_3 1 = 0.$

⇒ $n^0 = 1 > (T(n) = n^2)$
⇒ $T(n) = O(n^2)$

Date
Page

Question 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(1)
    }
}
```

for $i = 1 \Rightarrow j = 1, 2, 3, 4 \ldots n = n$

for $i = 2 \Rightarrow j = 1, 3, 5 \ldots n = n/2$

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

⋮

for $i = n \Rightarrow j = 1 \ldots 1$

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

$$\Rightarrow \sum_{j=n} n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots \frac{1}{n}\right].$$

$$\sum_{j=n} n[\log n].$$

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

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

Question 10: for functions, n^k & c^n, what is asymptotic relation b/w these functions?
Assume that $k >= 1$ & $c > 1$ 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 $n^k \leq a c^n$
$\forall\ n \geq n_0$ & some constant $a > 0$

for $n_0 = 1$
$c = 2$

$\Rightarrow 1^k \leq a\, 2^1$
$\Rightarrow n_0 = 1$ & $c = 2$