Design and Analysis of Algorithms

TUTORIAL - 1

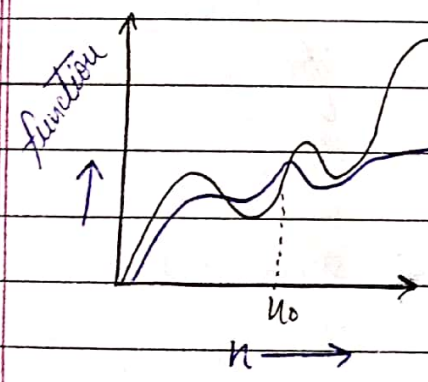
Ques (1) What do you understand by Asymptotic notations. Define different Asymptotic notation with Example.

→ Asymptotic Notation (tending to infinity)

↳ These notations are used to tell the Complexity of an algorithm when input is very large

1) Big O (O)

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



g(n) is "tight" upper bound

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

iff

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

($\forall\ n \geq n_0$ and some Const, $c > 0$)

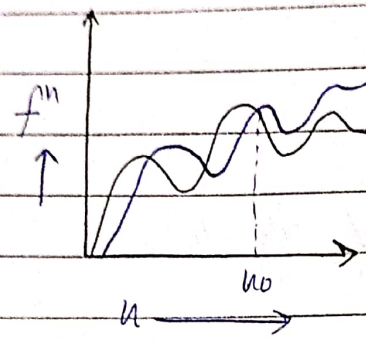
Ishika Gupta Utlavsi
'E_15'
DAA

Ishika

PAGE No.
DATE

(2) Big Omega (Ω)

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



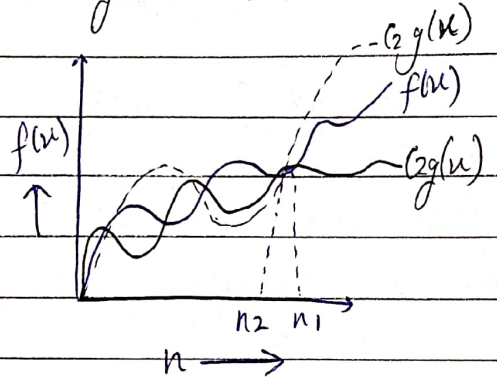
$g(n)$ is "tight" lower bound

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

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

$\forall \quad n \geq n_0$
& some Const, $c > 0$

(3) Big theta (Θ)



$g(n)$ is both "tight" upper & lower bound of $f(n)$

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

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

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

& some Const, $c_1 > 0, c_2 > 0$

Ques (2) What should be time Complexity of

for ($i = 1$ to n) { $i = i + 2$ }

→ for ($i = 1$ to n) // $i = 1, 2, 4, 8 - - - -, n$

{ $i = i \times 2$ } // $O(1)$

Ishika

Ishika Gupta Attavar

'E-15'

DAA

$\Rightarrow \sum_{(i=1)}^{n} 1+2+4+8+\cdots\cdots 1n$

GP of k^{th} Value $\rightarrow T_k = ar^{k-1}$

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

$n = 2^{k-1}$

$2n = 2^k$

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

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

$\log n + 1 = k$

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

$= O(\log n)$

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

$T(n) = 3T(n-1)$ — (1)

put $n = (n-1)$

$T(n-1) = 3T(n-2)$ — (2)

from (1) and (2)

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

$= 9T(n-2)$ — (3)

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

$T(n-2) = 3T(n-3)$ — (4)

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

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

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

putting $n-k = 0$

$k = n$

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

$= 3^n \times 1$

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

Ishika

Ishika Gupta Attavar

E-15

DAA

Ques(4) $T(n) = \begin{cases} 2T(n-1)-1 & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$

$T(n) = 2T(n-1)-1$ — (1)

Let $n = n-1$

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

from (1) and (2)

$T(n) = 4T(n-2)-1$ — (3)

put $n = n-2$

$T(n-2) = 2T(n-3)-1$ — (4)

from (3) and (4)

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

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

General $\Rightarrow 2^k T(n-k) - (2^k-1)$

$n-k=0$

$n=k$

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

$= 2^n \times 1 - 2^n + 1 = 1$

$T(n) = O(1)$

Ques(5) What should be the time Complexity of

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

Ishika

Ishika Gupta Allavas 'E-15'
DAA

PAGE No.
DATE

→ Sum of S = 1+3+6+10+ ------ + Tn -(1)

also S = 1+3+8+10+ ------ + Tn-1 + Tn -(2)

from (1) and (2)

0 = 1+2+3+4+ --- + n - Tn

Tk = 1+2+3+4+ ---+k

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

for k iterations

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

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

$O(k^2) <= n$

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

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

Ques 6) Time Complexity of : Void f(n) {int w) //O(n)

{ int i, count = 0; //O(1)

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

Count ++ //O(1)

}

→ i*i = $\underbrace{1^2, 2^2, 3^2, 4^2, \text{-----}, n}_{k \text{ terms}}$

⇒ k^{th} term :

$t_k = k^2$

$\Rightarrow k^2 = n$

$k = n^{1/2}$

$T(n) = O(1+1+1+n^{1/2}+1)$

$= O(n^{1/2})$

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

Ishika

Ishika Gupta
E-15
DAA

Ques (19) 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 * 2)
    { for (k=1 ; k <= n ; k = k * 2)
        count ++ ;
    }
  }
```

→ for $k = k * 2$

$k = 1, 2, 4, 8 \text{ ---- } n$

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

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

$n = 2^k$

$\log n = k$

$i \rightarrow 1, 2, \text{ ---- }, n$

$j \rightarrow \log n, \log n, \text{ ---- }, \log n$

$k \rightarrow \log n + \log n, \text{ ---- }, \log n * \log n$

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

$\Rightarrow O(n \log^2 n)$

Ishika Gupta 11615

DAA

Ques 2) Time Complexity of ;

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

→ for function Call,

$\underbrace{n, n-3, n-6, \dots, 1}_{K \text{ Terms}}$

⇒ AP with $d = -3$

$\Rightarrow l = a + (k-1)d$

$1 = n + (k-1)(-3)$

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

$\Rightarrow k-1 = \frac{n-1}{3}$

$\left(k = \frac{n+2}{3}\right)$

⇒ function gives a recursive call $\frac{n+2}{3}$ times

⇒ Time Complexity = $\left(\frac{n+2}{3}\right)(n)(n)$

$= n^2$

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

Ishika

Ishika Gupta Atlavar

DAA E-16

Ques (9) Time Complexity of Void function (int n)

```
{  for (i=1 to n)
   {  for (j=1; j<=n; j=j+i)
         print ("*");
   }
}
```

→

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 $i=n \rightarrow j = 1,\ldots,n = 1$

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

$\Rightarrow \sum_{j=n}^{1} n\,[1 + 1/2 + 1/3 + \ldots + 1/n]$

$\Rightarrow \sum_{j=n}^{1} n \log n \Rightarrow T(n) = [n \log n]$

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

Ques (10) → As Given, n^k & c^n

relation b/w n^k & c^n is $\boxed{n^k = O(c^n)}$

as $n^k \leq ac^n \;\forall\; n \geq n_0$ for (a) Constant $(a>0)$

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a2^1$

$\therefore \boxed{n_0 = 1 \text{ & } c = 2}$

Ishika

Ishika Gupta Sharma

E-15

DAA