Name: Krishna Pratap Singh University R.No: 2016821
Section: D Sem: 4th
Roll No: 17

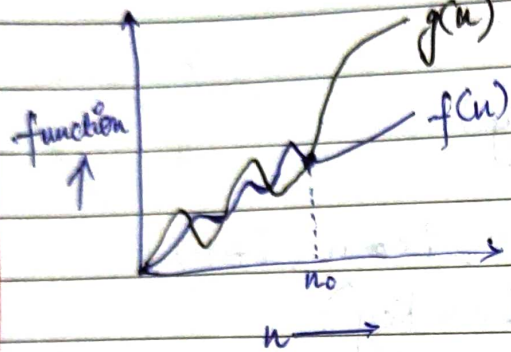
Asymptotic Notation

↳ tending to infinity

These notations are used to tell the complexity of an algo 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

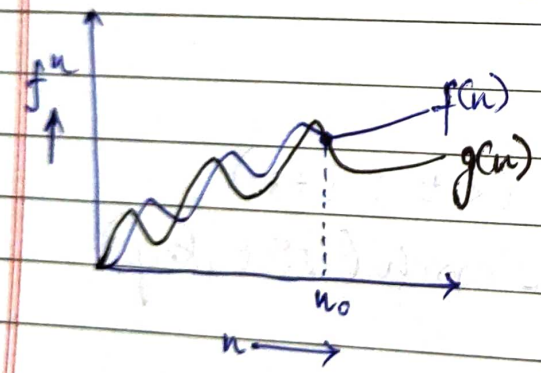
$f(n) \leq c\,g(n)$

$\forall\ n \geq n_0$

(and some const, $c > 0$)

② Big Omega (Ω)

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



g(n) is "tight" lower bound

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

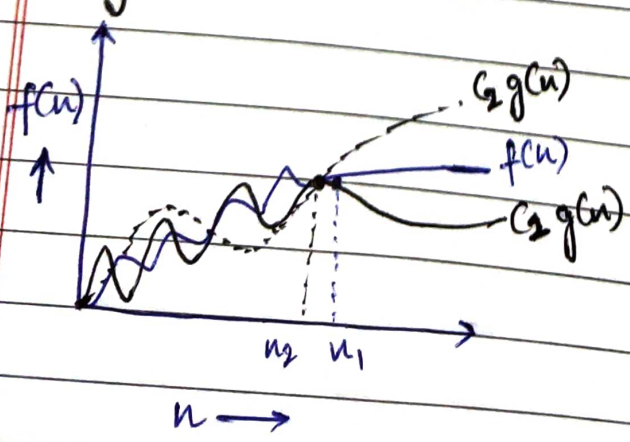
iff

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

$\forall\ n \geq n_0$

& some const, $c > 0$

③ Big theta (Θ)



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

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

iff

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

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

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

(Q2) What should be time complexity of
for(i=1 to n) { i=i+2 }

Ans) for (i=1 to n) // i = 1, 2, 4, 8 . . . , n
{i = i×2 } // O(1)

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

GP of K^{th} value $\rightarrow$ $T_K = ar^{K-1}$

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

$n = 2^{K-1}$

$2n = 2^K$

$\Rightarrow \log 2n = K \log 2$

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

$\Rightarrow \log n + 1 = K$

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

$= O(\log n)$

(Q3) $T(n) = \{3T(n-1)$ 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

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

$= 9T(n-2)$ —— ③

putting $n = n-2$ in ①

$T(n-2) = 3T(n-3)$ —— ④

putting $T(n-2)$ in ③

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

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

putting $n-k = 0$

$k = n$

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

$= 3^n \times 1$

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

(Ans 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

$T(n) = 4T(n-2)-1$ —— ③

put $n = n-2$

$T(n-2) = 2T(n-3)-1$ —— ④

from ③ & ④

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

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

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

$n-k = 0$

$n = k$

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

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

$T(n) = O(1)$

(Q5) What should be time complexity of

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

Ans) Sum of $s = 1+3+6+10+\cdots+T_n$ — ①

also $s = 1+3+8+10+\cdots+T_{n-1}+T_n$ — ②

from ① & ②

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

$T_K = 1+2+3+4+\cdots K$

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

for K iterations

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

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

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

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

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

(Q6) Time complexity of: void fn(int n) //O(n)

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

Ans) $i*i = \underbrace{1^2, 2^2, 3^2, 4^2, \ldots, n}_{K \text{ terms}}$

$\Rightarrow$ K^{th} term :

$t_K = K^2$

$\Rightarrow K^2 = n$

$K = n^{1/2}$

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

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

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

(Q7) 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++;
  }
 }}
```

Ans) for k = k*2

k = 1, 2, 4, 8 . . . . n

GP ⇒ a=1, r=2

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

$$n = 2^k$$

$$\log n = k$$

i → 1, 2, ... , n

j → logn, logn, ... , logn

k → logn * logn, ... , logn * logn

⇒ O(n * logn * logn)

⇒ O(n $\log^2 n$)

(Q8) 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);
}
```

Ans) for function call,

$$\underbrace{n, n-3, n-6, \ldots, 1}_{K \text{ terms}}$$

$\Rightarrow$ AP with $d = -3$

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

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

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

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

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

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

⇒ Time complexity = $\frac{(n+2)}{3}(n)(n)$

$= n^3$

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

PAGE
DATE

(89) Time complexity of

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

Ans)

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

$\vdots$

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

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

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

$$\Rightarrow \sum_{j=n}^{1} n \log n$$

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

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

(810) Ans10) As given, n^k & c^n

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

as $n^k \leq a c^n \quad \forall \quad n \geq n_0$ for (a) constant ($a > 0$)

for $n_0 = 1$

$c = 2$

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

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