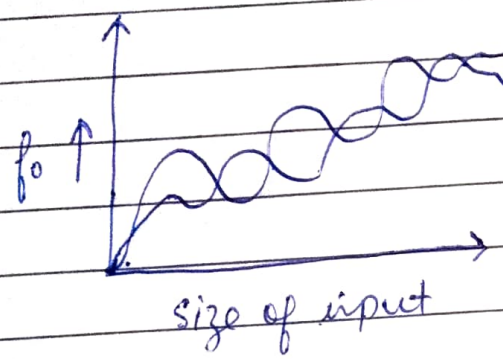
TUTORIAL - 1

NAME: SHARIK KHAN
SECTION: D
SEMESTER: IV
CRNO: 36
UNI. NO: 2017018
DATE: 10-March-2020

Ques ① Asymptotic Notations
└ tending to infinity

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

i) Big O (O)

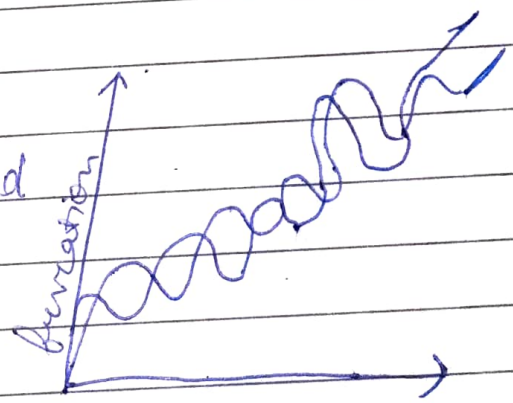


$f(n) = O(g(n))$

iff $f(n) \leq c \cdot g(n)$
$\forall n \geq n$
for some constant
$c > 0$
→ $g(n)$ is tight upper bound of $f(n)$

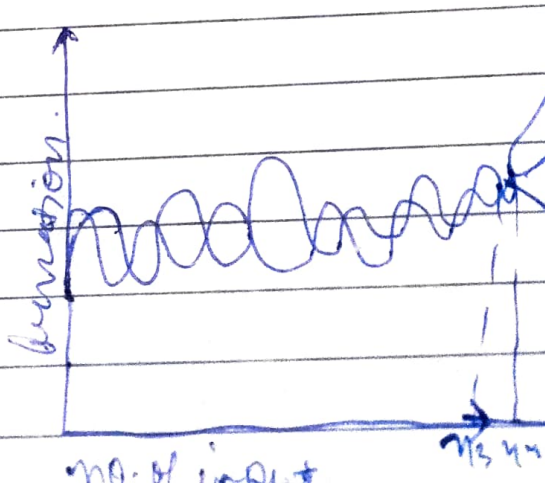
ii) Big Omega (Ω)

$f(n) = \Omega(g(n))$
$g(n)$ is tight lower bound of $f(n)$
$f(n) = \Omega(g(n))$
iff $f(n) \geq (g \cdot (n)$
$\forall n \geq n_0$ for some constant $c > 0$



iii) Theta (Θ)

$f(n) = \Omega(g(n))$
$g(n)$ is both tight upper and lower bound of f or $f(n)$
$f(n) = \Theta(g(n))$



Date / /

4.) Small o (o)

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

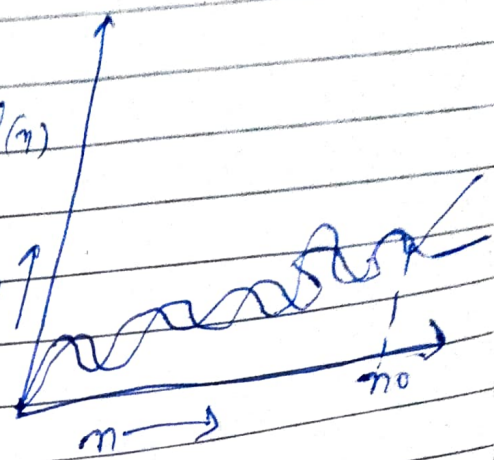
$g(n)$ is upper bound of fn $f(n)$

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

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

$\forall n > n_0$

$\in \forall c > 0$



5.) Small omega (ω)

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

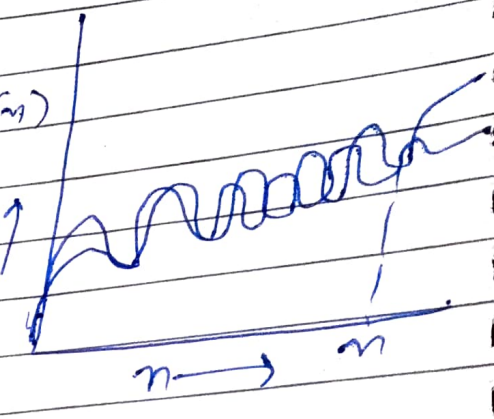
$g(n)$ is lower bound of fn $f(n)$

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

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

$\forall n > n_0$

$\& \forall c > 0$



ues what should be time complexity of -

for $\{i = 1 \text{ to } n\}\ \{i = i * 2\}$

for($i = 1$ to n) // $i = 2, 4, 8 \ldots n$

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

$\sum 1 + 2 + 4 + 8 + \ldots \ldots n$

GP with value $\Rightarrow T_K = a r^{K-1}$

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

$n = 2^{2K-1}$

$2n = 2^K$

$\log 2n = K \log 2$

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

Date / /

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

$= O(\log n)$

ques③ $T(n) = 3T(n-1)$ if $n>0$, otherwise 1

$T(n) = 3T(n-1)$ — ①

put $n = n-1$

$T(n-1) = 3T(n-2)$ — ②

put value of $n-1$ from ② to ①

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

$= 9T(n-2)$ — ③

put $n = n-2$ in ①

$T(n) = 3T(n-3)$ — ④

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

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

putting $n-k = 0$

$n = k$

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

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

$T(n) = 3^n \times 1$ $\quad \therefore T(0) = 1$

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

Date / /

(4) $T(n) = 2T(n-1) - 1$ if $n > 0$, otherwise 1

$T(n) = 2T(n-1) - 1$ — (1)

let $n = n-1$

$T(n-1) = 2T(n-2) - 1$ — (2)

from (1) & (2)

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

$T(n) = 4T(n-2) - 2 - 1$ — (3)

let $n = n-2$

$T(n-2) = 2T(n-3) - 1$ — (4)

from (3) & (4)

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

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

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

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

$a = 2^{k-1}$

$r = 1/2$

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

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

Date / /

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

$= 2^k - 1$

let $n-k=0$

$n=k$

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

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

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

$T(n) = O(1)$

(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 + 3 + 6 + 10 + 15 + 21 - - - n

Sum of $S = 1+3+6+10+\cdots T_n$ — (1)

also $S = 1+3+6+10+\cdots T_{n+1} + T_n$ — (2)

from (1) − (2)

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

Date

$T_k = 1+2+3+4+\cdots+k$

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

for k interations

$1+2+3+\cdots\cdots+k <= n$

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

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

$O(k^2) <= n$

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

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

Time Complexity of -

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

as $i^2 <= n$

$i <= \sqrt{n}$

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

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

Date / /

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

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

$$T(n) = O(n)$$

Ques (7) Time complexity of

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

for $k = k^{*}2$

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

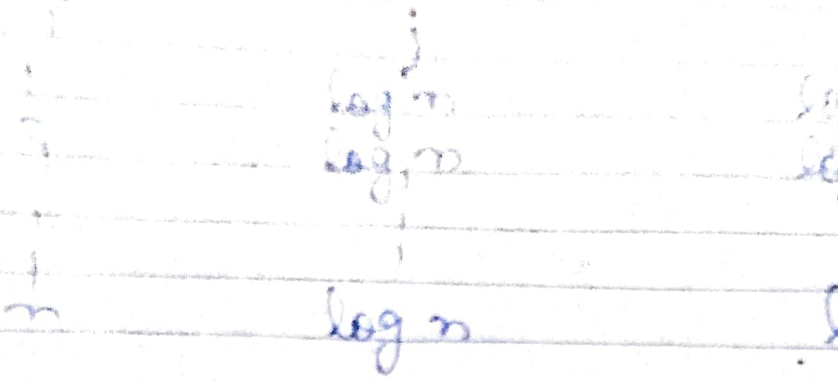
GP $\Rightarrow$ $a = 1$, $r = 2$

$$R_0 = \frac{a(r^n - 1)}{r - 1}$$

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

$n \to 2^k$

$\log n \to k$



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

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

Ans (8) Time complexity of

```
funcation (int n)
{
  int (n==1)
  return;                    // O(1)
  for (i=1 to n)             // i = 1, 2, 3 ---- n = O(n)
  {
    for(j= to n)             // j = 1, 2, 3 ---- n^3 = O(n^3)
    {
      print (*);
    }
    funcation(n-3);          T(n/3)
  }
```

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

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

$c = \log_3^1 = 0$

$n^0 = 1 \quad > (f(n) = n^2) \quad T(n) = O(n^2)$

Ques (4) Time complexity of

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

for i=1 ⇒ j = 1, 2, 3, 4 - - - n = n
for i=2 ⇒ j = 1, 3, 5 - - - - n = n/2
for i=3 ⇒ j = 1, 4, 7 - - - - n = n/3
.
.
.
for i=n ⇒ j = 1 - - - - - - - n

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

$$\sum_{j=n}^{1} n \left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n}\right]$$

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

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

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

Date / /

Ques 10 for function $n^k + c^n$, what is the asymptotic relation between 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 $n_0 = 1$
$c = 2$

$1^k \leq a 2^1$

$n_0 = 1$ & $c = 2$

TUTORIAL - 1

NAME : SHARIK KHAN
SECTION : D
SEMESTER : IV
CRNO : 36
UNI. NO : 2017018
DATE : 10-March-2020