Name → Anadi
Section → ML
Roll no → 13

Assignment 1

TCS409

Q1. What do you mean by Asymptotic notations Define different Asymptotic notations with examples

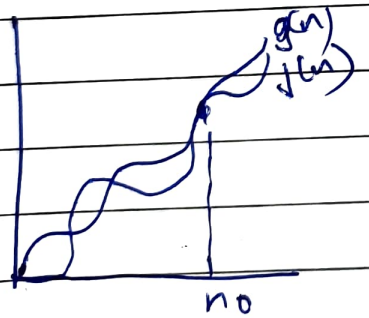
Asymptotic notations are the mathematical tools which are used to tell complexity of an algorithm when i/p is very large.

$O(n^2)$ = no of instructions where n is no of i/p

① **Big O Notation (O)**

It describes tight upper bound of an algorithm's time complexity in worst case scenario

$$f(n) \leq O(g(n))$$



g(n) is tight upper bound of f(n)

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

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

$\forall\ n \geq n_0$

and for some constants.

→ BIG OMEGA (Ω) Notation

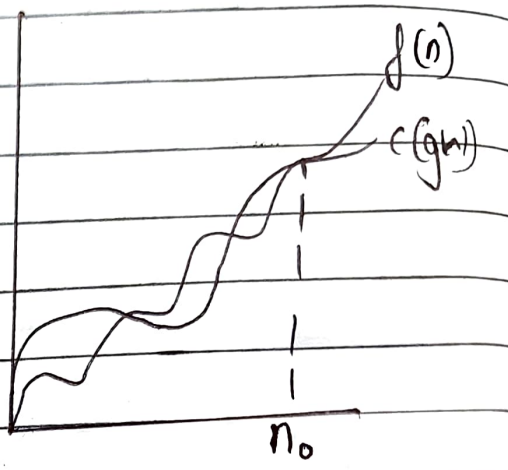
$$f(n) = \Omega(g(n))$$

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

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

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

$\forall\ n \geq n_0$ and for some constants $c > 0$



→ BIG THETA (Θ) Notation

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

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

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

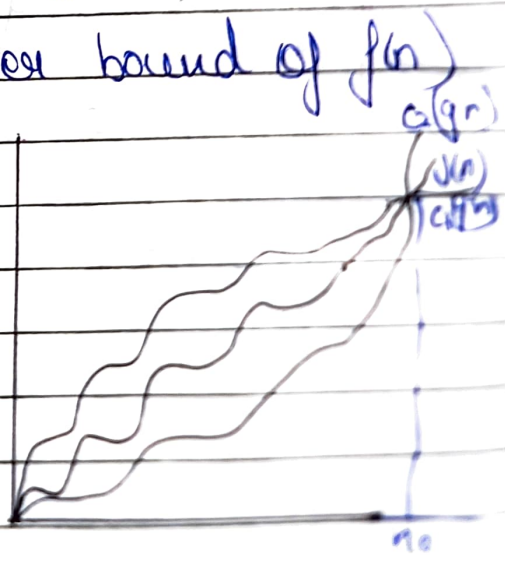
$$f(n) = \Theta(g(n))$$

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

$\forall\ n > \max(n_1, n_2)$ &

for some constants

$(c > 0)$



→ SMALL (o) Notation

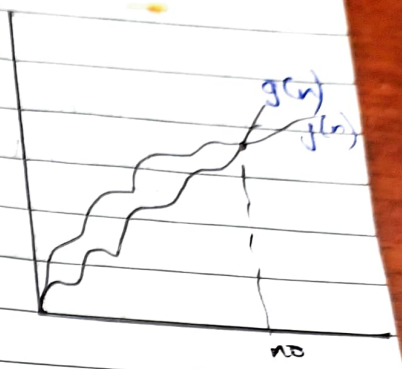
$f(n) = o\,g(n)$

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

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

iff $f(n) < c(g(n))$

$\forall\, n \geq n_0$ and



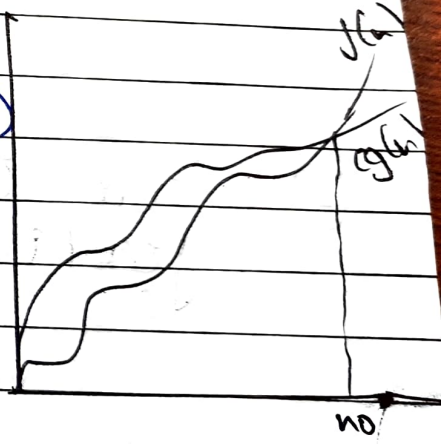
→ SMALL OMEGA (n) Notation

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

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

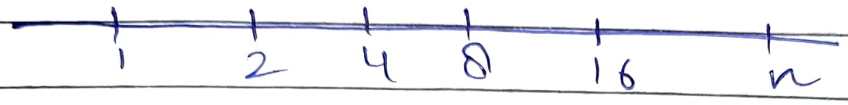
$f(n) = n\,g(n)$

iff $f(n) > c(g(n))$



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

```
sum = 0
for (i=1, i<=n; i*=2 )
{
    sum+ = i
}
```



This is forming a GP

$n = ar^{k-1}$

Where $a = 1$

$r = 4/2 = 2$

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

$n = 2^{k-1}$

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

$2n = 2^k$

taking log on both sides

$\log_2(2n) = k \log_2 2$

$k = \frac{\log_2(2n)}{\log_2(2)}$ - constant

$= \log_2(2n)$

$= \log_2 n + \log_2 2$

$= \log_2 n + (\log_2 2 - 1)$

$= \log_2 n$

Time complexity = $O(\log_2 n)$

Q3) $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

$T(0) = 1$

$3T(n-1)) = ?$

for $T(1)$

$T(1) = 3T0$

for $T(2)$

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

$= 3T(0)$

$= 3*3*1$

$T(3) = 3T(3-1)$

$= 3T(2)$

$= 3*3*3*1$

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

$= 3*3*3*3$

$= 3^n$

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

Q4) $T(n) = \{2T(n-1)-1$ if $n>0$, else $1\}$

$T(0) = 1$

for $T = 1$

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

$= 2T(0)-1$

$= 2-1$

$= 1.$

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

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

⋮

for $T(n)$
$T(n) = 2T(n-1)-1$
$= 2n-(2n-2)- \dots -4-2-1$

$\boxed{T(n) = O(1)}$

Q5)
```
int i=1 ; s=1
while (s>n){
  i++ ;
  s=s+i ;
  print ("#"); }
```

$S_i = S_{i-1} + i$

when $i=1$, $S_1 = S_0 + i \Rightarrow S=1$
when $i=2$, $S_2 = S_1 + 1 = 1+2 \Rightarrow S=3$
when $i=3$, $S_3 = S_2 + 3 = 3+3 \Rightarrow 6$
$1+3+6+10 \dots \dots \dots k=n$

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

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

$O(k^2) = n$ $\boxed{K=\sqrt{n}}$

Q6)
```
void function (int n) {
    int i, c=0
    for (i=1; i*i<=n; i++)
    {
        c++;
    }
}
```

Check $i * i <= n$

$i \times i$ should be less than equal to n

When $i = 1$ $\quad 1 * 1 \leq n \Rightarrow 1 \leq 4$

$i = 2$ $\quad 2 \times 2 = 4 \leq n \Rightarrow 4 \leq n$

$i = n$ $\quad \sqrt{n} \times \sqrt{n} = n \leq n$

So the loop will be

$1, 2, 3 - - - \sqrt{n}$

no of iterations k is bound by $\sqrt{n}$

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

Q7 void fun (int n) {
int i, j, k, c = 0;
for (i = n/2; i <= n; i++)
for (j = 1; j <= n; j = j * 2)
for (k = 1; k <= n; k = k * 2)
c++;
}

1) i iterates from $\frac{n}{2}$ to n. Its time complexity is $O(n)$
2) j iterates from 1 to n with double increment (j = j * 2)
TC = O(log n)
3) k iterates from 1 to n with a double increment (k = k * 2)
TC = O(log n)

$O(n) \times O(\log n) \times O(\log n) = O(n \log^2 n)$

Q8 fun(int n) {
if (n == 1) return;
for (i = 1 to n) { (n times) } $O(n^2)$
for (j = 1 to n) { (n times) }
print ("#");
}
fun(n - 3);

TC of both inner loops $O(n^2)$
$T(n) = T(n-3) + O(n^2)$
as $T(1) = O(1)$
$TC = O(n^2)$

Q9
```
void fun (int n)
{
    for (i=1 to n) {
        for (j=1; j<=n; j=j+i)
            printf ("*");
    }
}
```

for $j = n/1 + n/2 + n/3 + \dots + n/n$

$n = 1e^{K-1} \Rightarrow n = 2^K/2 \Rightarrow 2n = 2^K$

taking log both side

$\log 2n = \log 2^K$

$TC = O(\log_2 n)$

TC is $O(n \log n)$

Q10 For function n^k & c^n, what is asymptotic relationship b/w these function Assume that $k >= 1$ & $c > 1$ are constant find value of c & no for which relation holds.

n^k grows polynomially with n

c^n " exponentially with n

thus $c^n = n^k$

so, n^k is $O(c^n)$

find value of c & no

$\log n^k = \log (c^n)$

$\Rightarrow c \geq e$ & $no = k$