Name - Anshika Semwal
Sec - CST
Roll no. - 26

Tutorial-1

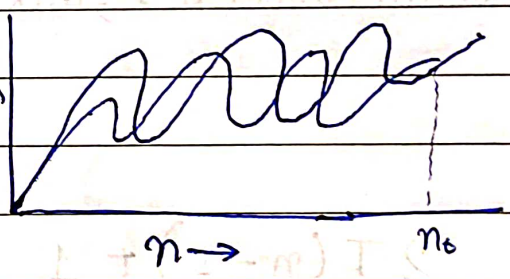
Q1- What do you understand by Asymptotic notations. Define different notation with example.

Sol Asymptotic Notations :- They are the mathematical notations used to describe the running time of an algorithm when the input ends towards a particular value or a limiting value.

Different asymptotic notations -

1) Big O(n)

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



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

if $f(n) \leq c.g(n)$ $\forall n \geq n_0$.

for $c > 0$

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

ex. $f(n) = n^2 + n$

$g(n) = n^3$

$n^2 + n \leq c\, n^3$

$n^2 + n = O(n^3)$

(ii) Big Omega (Ω)

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

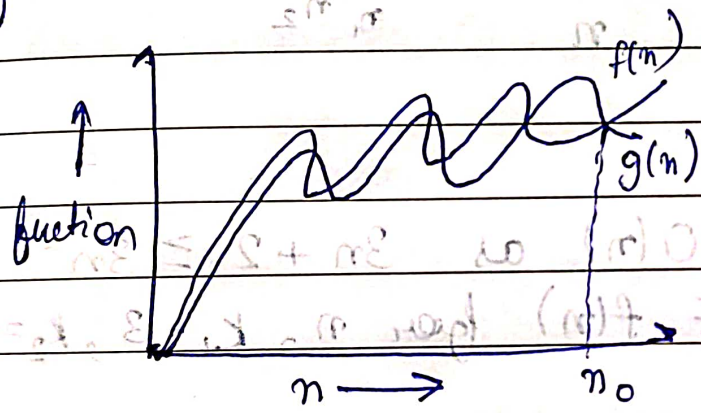
$g(n)$ is "tight" lower bound of function $f(n)$.

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

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

$\forall\, n \geq n_0$

for some const $c > 0$



Ex. $f(n) = n^3 + 4n^2$

$g(n) = n^2$

$= n^3 + n^2$

$= \Omega(n^2)$

(iii) Big theta (Θ)

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

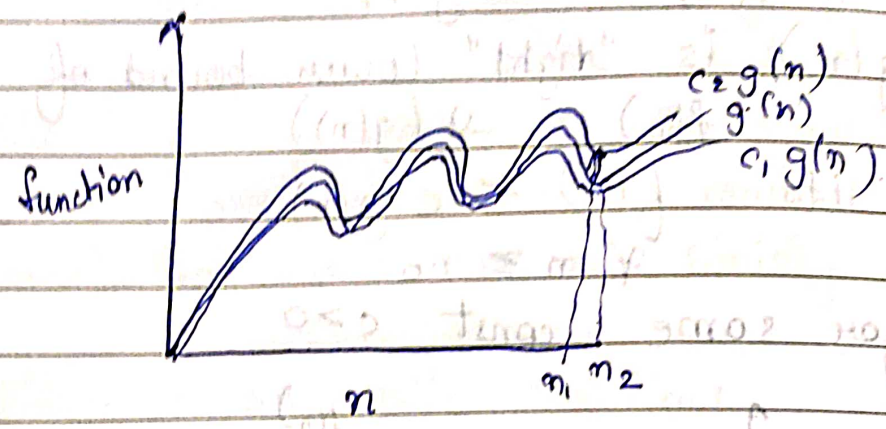
$g(n)$ is both "tight" upper and "lower bound" of function $f(n)$.

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

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

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

for some constant $c_1 > 0$ & $c_2 > 0$



Ex.

$3n+2 = O(n)$ as $3n+2 \geq 3n$ & $3n+2 \leq f(n)$ for n, $k_1 = 3$, $k_2 = 4$ & $n = 2$

(iv) Small O (O)

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

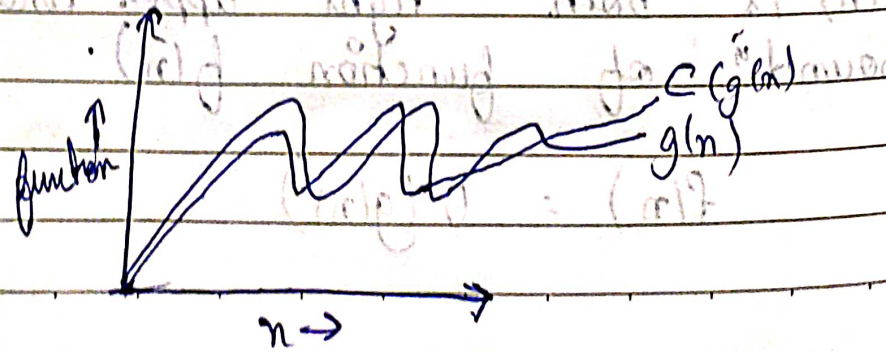
$g(n)$ is upper bound of function $f(n)$

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

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

$\forall\ n > n_0$

and $\forall$ constants, $c > 0$



Ex. - $f(n) = n^2$

$g(n) = n^3$

$n^2 = O(n^3)$

(v) Small Omega (n)

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

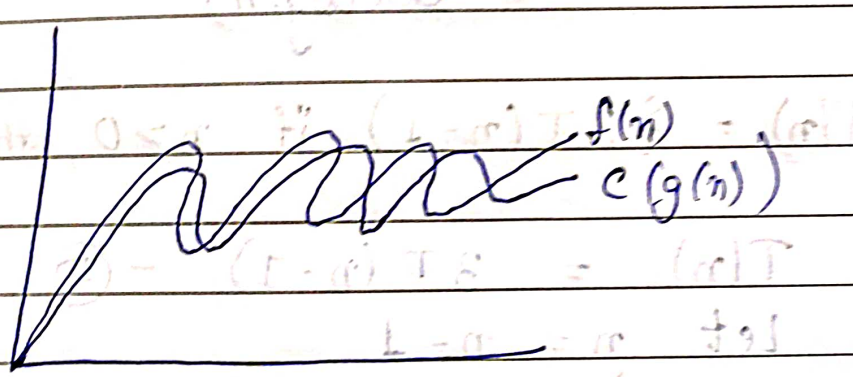
$g(n)$ is lower bound of $f(n)$

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

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

$\forall\ n > n_0$

and $\forall$ constants, $c > 0$



$f(n) = 4n + 6 \qquad g(n) = (1)$

Q2- What should be time complexity of

for ($i = 1$ to n $i = i * 2$; }

for ($i = 1$ to n)

$p = i * 2$

$i = 1, 2, 4, 8, 16 \cdots n$ G.P

$a = 1 \quad r = 2$

G.P. k^{th} value $= t_k = a^{k-1}$

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

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

$2n = 2^k$

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

$k = \log_2 2n$

$k = \log_2 2 + \log_2 n$

$k = 1 + \log n$

Time comp $= O(1 + \log_2 n)$

$= O(\log_2 n)$

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

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

Let $n = n-1$

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

Put (2) in (1)

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

Put $n = n-2$

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

Put (4) in (3)

$T(n) = 3 \times 3 \times 3T(n-3)$ - (5)

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

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

$= O(3^n)$

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

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

$T(n) = 2T(n-1) - 1$ — (1)

replacing n with $n-1$

$T(n-1) = 2T(n-2) - 1$ — (2)

replacing n with $n-2$

$T(n-2) = 2T(n-3) - 1$ — (3)

from (1) (2) (3), we get

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

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

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

$T(n) = 2^3 T(n-3) - 2^2 - 2 - 1$ using (3)

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

we know that

$T(0) = 1$

$n - k = 0$

$k = n$

on putting $k = n$, we get

$T(n) = 2^n T(0) + 2^{n-1} + 2^{n-2} + \cdots 2 + 1$

$= \underbrace{2^n + 2^{n-1} + 2^{n-2} + \cdots + 2 + 1}$

G.P

$$T(n) = \frac{1 \cdot (2^{n+1} - 1)}{2 - 1}$$

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

Time Complexity = $O(2^n)$

5. What should be time complexity of

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

Sol. s depends on i, so we make case

At i=1, s=1	2	3	4	...	n
	3	6	10	...	k

at $i = n$, $s = k$ & k breaks the while condition so we can clearly see the s is just sum of n natural no. so

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

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

$$k^2 = n$$

$$k = \sqrt{n}$$

Page No.: YOUVA
Date:

Time complexity = $O(\sqrt{n})$

6. Time complexity of

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

$i = 1, 2, 3, \ldots n$

$i^2 = 1, 4, 9, \ldots n$

So $i^2 <= n$ or $i <= \sqrt{n}$

$a_n = a + (k-1)d$

$a = 1$; $d = 1$

$a_k <= \sqrt{n}$

$\sqrt{n} = 1 + (k-1)1$

$\sqrt{n} = k$

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

7. Time Complexity of

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

All are independent loop,

so time complexity = $\frac{n}{2} * \log_2 n * \log_2 n$

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

8. Time complexity of

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

$T(n) = T(n-3) + n^2$ — (1)

$T(1) = 1$ — (2)

put $n = n-3$ in (1)

$T(n-3) = T(n-6) + (n-3)^2$ — (3)

Put (3) in (1)

$T(n) = T(n-6) + (n-3)^2 + n^2$ — (4)

put $n = n-6$ in (1)

$T(n-6) = T(n-9) + (n-6)^2$ — (5)

Date:

Put (5) in (9)

$$T(n) = T(n-9) + (n-6)^2 + (n-3)^2 + n$$

Generalizing

$$T(n) = T(n-3k) + (n-3(k-1))^2 + (n-3(k-2))^2 + \ldots + n^2$$

let $n - 3k = 1$

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

$$T(n) = T(1) + \left(n - 3\left(\frac{n-1}{3} - 1\right)\right)^2 + \left(n - 3\left(\frac{n-1}{3}\right)\right)^2 + \ldots n^2$$

$$T(n) = T(1) + (n-(n-1)-3)^2 + (n-(n-1-6))^2 + (n-(n-1-9))^2 + \ldots n^2$$

$$T(n) = 1 + (3+1)^2 + (6+1)^2 + \ldots n^2$$

$$T(n) = 1^2 + 4^2 + 7^2 + \ldots n^2$$

$$T(n) = n^2 + \ldots 1$$

$$T = O(n^2)$$

9. Time Complexity of

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

for $i=1 \rightarrow j=1$ to $n \rightarrow$ n times
$i=2 \rightarrow j=1$ to $n \rightarrow$ $n/2$ times
$i=3 \rightarrow j=1$ to $n \rightarrow$ $n/3$ times
⋮
$i=n \rightarrow j=1$ to $n \rightarrow$ 1 times

So total $= n + n/2 + n/3 + n/4 + \cdots + 1$

$$n\underbrace{\left(1 + 1/2 + 1/3 + 1/4 + \cdots 1/n\right)}_{\log n}$$

$T(n) = n \log n$

Time complexity $= O(n \log n)$

Date: YOUVA

10. For the function, n^k and c^n, what is the asymptotic relationship b/w these function?
Assume that $k >= 1$ and $c > 1$ constants.
Find out the value of c and no. for which relationship holds?

Sol As given n^k d c^n
relation b/w n^k d c^n is
$n^k = O(c^n)$
as $n^k \leq d\, c^n$
if $n \geq n_0$ d some conitant $a > 0$
for $n_0 > 1$
$c > 2$
$1^k \leq d_2$
$n_0 = 1$
$c = d\, 2$