Name: T Rishita Reddy
Roll no: 47
Sec: I

Page No. ______
Date ______

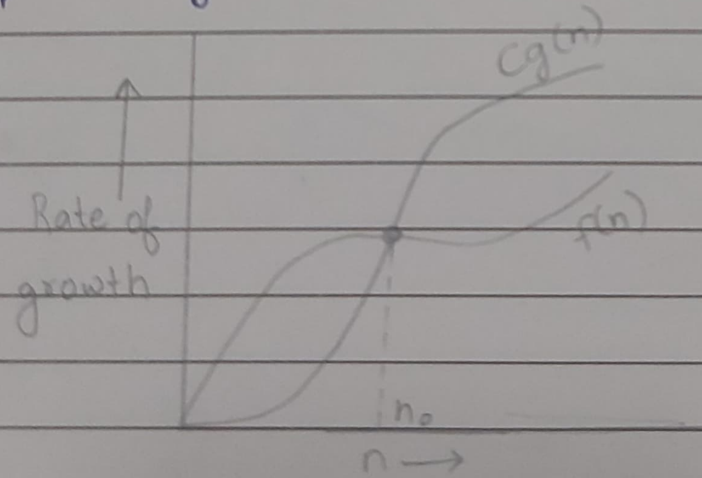
Tutorial-01

1. What do you understand by Asymptotic notations. Define different Asymptotic notation with examples.

Ans⇒ Asymptotic Notations are used to analyze an algorithm when input is large. The efficiency of an algorithm depends on the amount of time, storage and other resources required to execute the algorithm. The efficiency is measured with the help of asymptotic notations.

Different types of Asymptotic notations are:

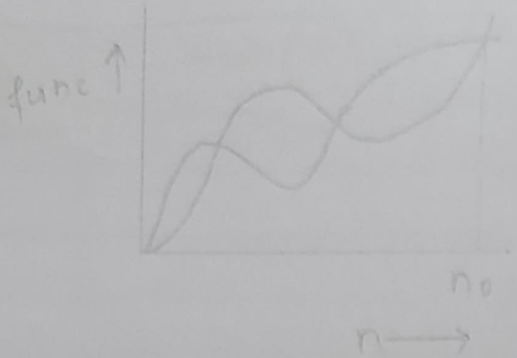
① Big-O Notation (O-notation): Big-O notation represents the upper bound of the running time of an algorithm. Thus it gives the worst-case complexity of an algorithm.



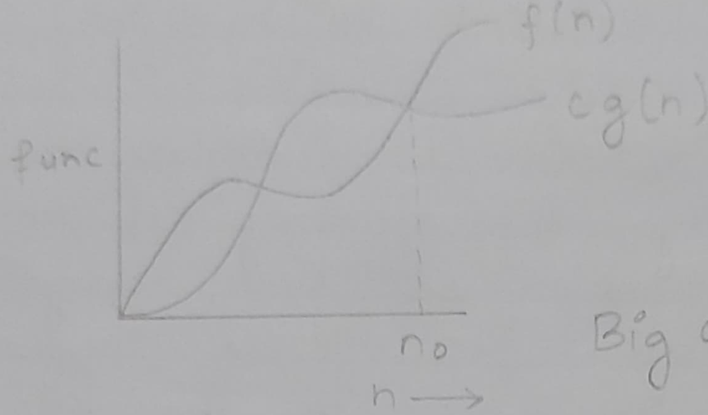
$f(n) = O(g(n))$
if $0 \le f(n) \le cg(n)$
if $n \ge n_0$ for some constant c

g(n) is "tight" upperbound on f(n)

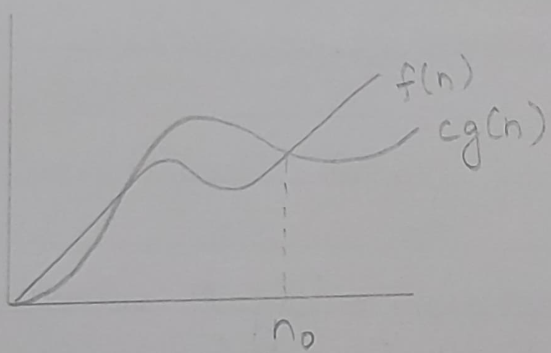
Date



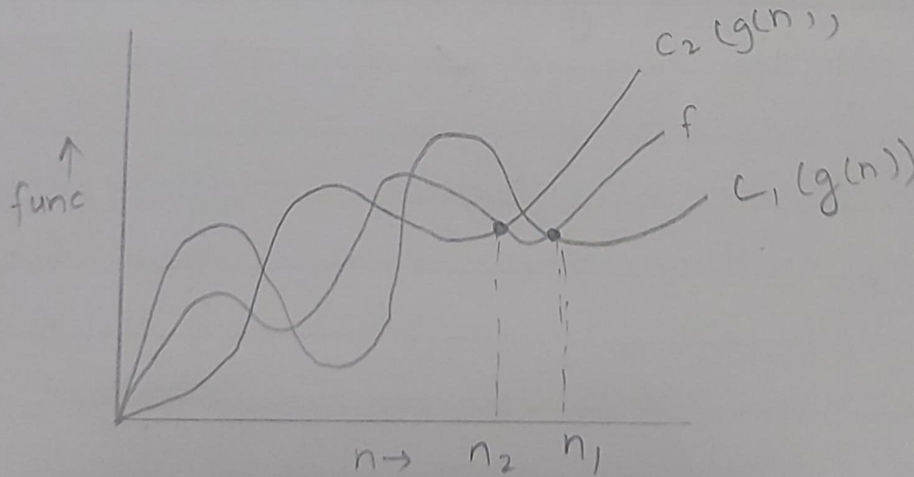
Small O



Big omega



Small omega



Theta

② Small-oh (o):
$f(n) = o(g(n))$ if $f(n) < c\,g(n)$ $\forall$ $n > n_0$ & $\forall$ $c > 0$
This gives us upper bound.

③ Big Omega (Ω):
$f(n) = \Omega(g(n))$ if $f(n) \geq c\,g(n) \geq 0$
$\forall$ $n \geq n_0$ & some constant $c > 0$
$g(n)$ is the tight lower bound of $f(n)$

④ Small omega (ω):
$f(n) = \omega(g(n))$ if $f(n) > c\,g(n)$ $\forall$ $n > n_0$ & $\forall$ $c > 0$
$g(n)$ is lower bound on $f(n)$.

⑤ Theta (Θ):
$f(n) = \Theta(g(n))$ iff $c_1(g(n)) \leq f(n) \leq c_2(g(n))$ $\forall$ $n \geq$ $\max(n_1, n_2)$ & some constant c_1 & $c_2 > 0$.
Theta gives the tight upper and lower bound both.

Q.2 What should be time complexity of-
```
for(i=1 to n)
{ i=i*2;
}
```
$i = 1, 2, 4, 8 \ldots n$
$n = ar^{k-1}$
$n = 2^{k-1}$
$\log n = k-1$
$k = \log_2 n + 1$

Date : Page :
Topic :

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

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

$T(n) = 3T(n-1) \qquad T(0) = 1$

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

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

$T(n) = 3.3T(n-2)$

$T(n) = 3.3.3T(n-3)$

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

let $n-k=0$, $n=k \Rightarrow T(n) = 3^n T(0)$

as $T(0)=1$

$T(n) = 3^n$

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

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

$T(n) = 2T(n-1) \qquad T(0) = 1$

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

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

$T(n) = 2.2T(n-2) -2 -1$

$T(n) = 2.2(2T(n-3)-1)-2-1$

$T(n) = 2.2.2T(n-3) - (1+2+2.2)$

$\vdots$

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

let $n-k=0$

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

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

Signature..........................

Date : ________ Page : ________
Topic : ________

Q.5 Time complexity of

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

for $i=1$, $s=1$

$i=2$, $s=1+2$

$i=3$, $s=1+2+3$

Sum of n natural numbers so,

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

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

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

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

Q.6 Time complexity

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

$i = 1, 2, 3, 4 \ldots n$

$i^2 = 1, 4, 9, 16 \ldots n$

so, $i^2 <= n \Rightarrow i <= \sqrt{n}$

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

$i <= \sqrt{n}$

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

$k <= \sqrt{n}$

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

Q.7

```
void fun(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 ++;
}
```

Signature ....

Date : Page :
Topic :

i	j	k
$n/2$	$\log_2 n$	$\log_2 n$
⋮	⋮	⋮
n	$\log_2 n$	$\log_2 n$

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

$$O(i*j*k) = O\left(\left(\frac{n}{2}+1\right) \times \log n \times \log n\right) = O(n(\log n)^2)$$

Q.8 Time complexity -

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

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

let $n-3k = 1$

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

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

$$T(n) = T(1) + [n - (n-1-3)]^2 + \_\_ + n^2$$
$$T(n) = 1 + (3+1)^2 + (4+1)^2 + \_\_ + n^2$$
$$T(n) = 1 + 4^2 + 7^2 + \_\_ + n^2$$
$$T(n) = \frac{n(n+1)(2n+1)}{6}$$

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

Signature..........................