Nikhil Singh
'I' 20

Design and analysis of Algorithm

1

Tutorial - 1

Ques1. What do you understand by Asymptotic notations. Define different Asymptotic notations with examples.

Asymptotic notations means towards infinity. They are used to tell the complexity of an algorithm having input size very large. It is priory analysis.

Different types of asymptotic notations are:

[i] Big Oh notatation (O)

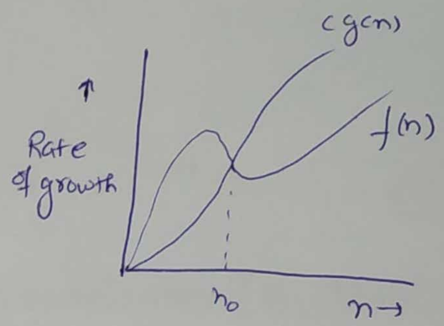
$f(n) = O(g(n))$, if $0 \le f(n) \le c(g(n)) \; \forall \; n \ge n_0$

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

Example:

```
for (int i=0 ; i<n; i++)
{  cout<<i <<endl;
}
```

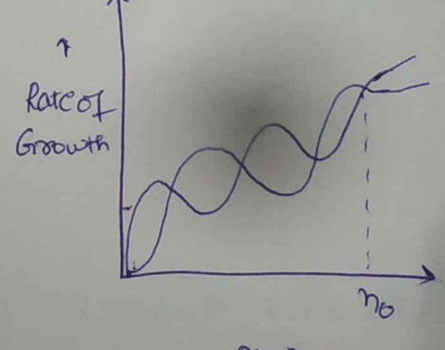
$T(n) = O(n)$



[ii] Small oh Notation (o)

$f(n) = o(g(n))$ if $f(n) < c(g(n)) \; \forall \; n > n_0$ & $c > 0$

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



[iii] Big Omega (Ω)

$f(n) = \Omega(g(n))$, if $f(n) \ge c(g(n)) \ge 0 \; \forall \; n \ge n_0$ & some constant $c > 0$

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

Example: $f(n) = 6n^2+n+1$, $g(n) = n^2$

$0 \le c\,g(n) \le f(n)$

$0 < c \cdot n^2 \le 6n^2+n+1$

$c \le 6 + \frac{1}{n} + \frac{1}{n^2}$ on putting $n=\infty$, $\frac{1}{n} = \frac{1}{\infty} = 0$

$c \le 6$

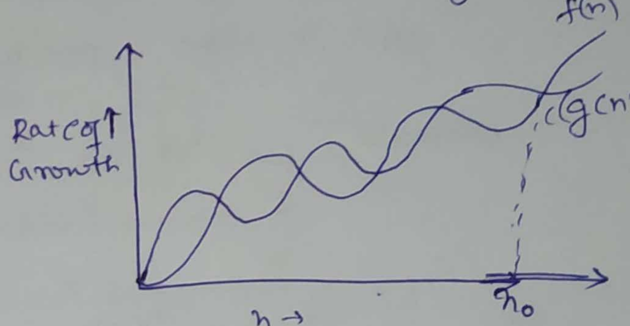
$6n^2 \le 6n^2+n+1 \Rightarrow (n=1)$

$6 \le 6+1+1 \Rightarrow 6 \le 8$ True $\because c > 0$ and $n \ge n_0 (n=1)$ $(n_0 = 1)$

$f(n) = \Omega(n^2)$

(iv) Small omega (ω)

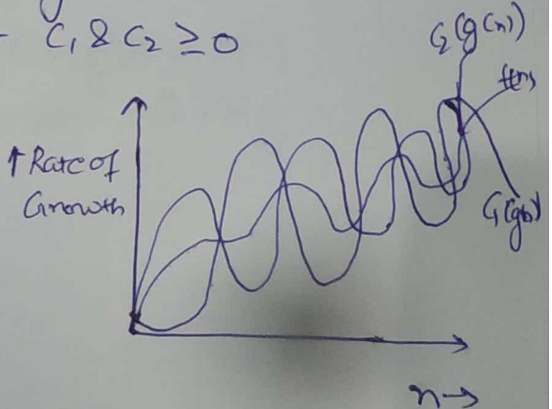
$f(n) = \omega(g(n))$, if $f(n) > c(g(n))$ $\forall n > n_0$ & $\forall c > 0$. $g(n)$ is the lower bound of $f(n)$



[V] Theta (Θ) notation

$f(n) = \Theta(g(n))$, if $c_1(g(n)) \le f(n) \le c_2(g(n))$

$\forall n \ge \max(n_1, n_2)$ and some constant c_1 & $c_2 \ge 0$



Ques 2. What should be time complexity of { for (i=1 to n) { i = i*2; } }

i would have 1, 2, 4, 8, 16 .... n

let say there are k terms

It is a G.P with $a=1$ $r=2$

Now, k^{th} term $= t_k = ar^{k-1}$

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

$n = 2^{k-1}$

Taking $\log_2$ on both sides

$$\log_2 n = \log_2 (2^{k-1})$$

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

$$\log n = k-1$$

$$\Rightarrow k = 1 + \log_2 n$$

$$T(n) = O(k) = O(1 + \log n) \Rightarrow \underline{\underline{O(\log n)}}.$$

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

$$T(n) = 3T(n-1) \quad -① $$

By Backward subtitution

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

$$T(n-1) = 3T(n-2) \quad - ②$$

Put ② in ①

$$T(n) = 3(3T(n-2)) \Rightarrow T(n) = 9T(n-2) \quad - ③$$

$$T(n) = 27\,T(n-3) \qquad T(n-2) = 3T(n-3)$$

continue for k times

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

assume $n-k=0 \Rightarrow n=k$

$$T(n) = 3^k\, T(0) \qquad \boxed{\because T(0) = 1}$$

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

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

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

$$T(n) = 2T(n-1) - 1 \quad -①$$

By using Back substitution method

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

$T(n-1) = 2T(n-2) - 1$ — ②

Putting ② in ①

$T(n) = 2^2\ T(n-2) - 2 - 1$ — ②

$T(n) = 2^3\ T(n-3) - 4 - 2 - 1$

⋮ Continue k times

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

Assume $n-k=0 \Rightarrow n=k$

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

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

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

GP k terms

$a = 2^{n-1}$, $r = 2^{-1} = 1/2$

Sum of G.P

$\dfrac{a(\cancel{1}-r^{n-1})}{1-r} = \dfrac{2^{n-1}(1-(1/2)^{n-1})}{1-1/2} = \dfrac{2^{n-1}\left(1-\frac{(1/2)^n}{1/2}\right)}{1/2}$

$= \dfrac{2^n(2^n-2)}{2^n} \Rightarrow 2^n - 2$

$2^n - [2^n - 2] \Rightarrow 2 \Rightarrow T(n) = O(2)$

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

$T(n) - O(n-3) + 7C$

Ques 5 What should be the complexity of

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

i	s
1	1
2	3
3	6
4	10
5	15
.	.
.	.
n	n

k times.

$s = 1, 3, 6, 10, 15, \ldots n$

let say k term

k^{th} term $\Rightarrow$ $t_k = t_{k-1} + t_k$

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

$$k^2 = 2n$$

$$k = \sqrt{n}$$

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

Ques 6. Time complexity of

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

$i*i$

$1 \times 1 = 1^2$

$2 \times 2 = 2^2$

$3 \times 3 = 3^2$

$\vdots$

$k \times k = k^2 = n$

$1^2, 2^2, 3^2, \ldots n$

let say k terms

$t_k = k^2$

$n = k^2 \Rightarrow k = \sqrt{n}$

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

P7. Time complexity of

```
void func (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 ++;
}
```

$i = \frac{n}{2}, \frac{n}{2}+1, \frac{n}{2}+2 \ldots\ldots n$

$= \frac{n}{2}, \frac{n+2}{2}, \frac{n+4}{2} \ldots n$

General form $= \frac{n+0*2}{2} + \frac{n+1*2}{2} + \frac{n+2*2}{2}, \ldots\ldots n$

$= \frac{n+k*2}{2} \qquad (k=0,1,2,\ldots n)$

Total terms $= k+1$

$t_{k+1} = n$

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

$n-2 = 2k$

$k = \frac{n}{2} - 1$

i	j	k
$n/2$	logn time	$(\log n)^2$
$\frac{n+2}{2}$	logn times	$(\log n)^2$
⋮		
n	logn times	$(\log n)^2$

$\left(\frac{n}{2}-1\right)$ time

$\Rightarrow \left(\frac{n}{2}-1\right)(\log n)^2$

$= n/2 \log^2 n - \log^2 n$

$$\boxed{T(n) = O(n\log^2 n)}$$

Q4 Solve the Recurrence relation $T(n) = T(n/4) + T(n/2) + cn^2$

Tutorial

Q8 Time complexity of

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

Function call would be $n, n-3, n-6, n-9, \ldots,$

let say k times

A.P = $a = n$, $d = -3$

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

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

$1 = n - 3k + 3$

$3k = n+2$

$k = \frac{n+2}{3}$

function have recursive call $\frac{n+2}{3}$ times

Time complexity for two inner loop $= n^2$

$\frac{(n+2)}{3} n^2 \Rightarrow n^3$

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

Q9 Times complexity of → void function (int n)

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

for i (outer loop)

when $i=1$ → $j = 1, 2, 3, 4 \ldots n \Rightarrow n$

when $i=2$ → $j = 1, 3, 5, 7 \ldots n \Rightarrow n/2$

when $i=3$ → $j = 1, 4, 7 \ldots n \Rightarrow n/3$

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

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

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

Q10 For the function, n^k and c^n, what is the assymptotic relationship b/w these functions.
Assume that $k >= 1$ and $c > 1$ are constants. Find out the value of c and n_0 for which relation holds.

As given n^k and c^n
relation b/w n^k and c^n is $n^k = O(c^n)$

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

for $n_0 = 1$
$c = 2$

$1^k \leq a 2^1$

$\therefore n_0 = 1$ as $c = 2$