SHRADDHA BAHUGUNA
CST SPL-1
30

Date.
Page No.

DESIGN AND ANALYSIS OF ALGORITHMS

TUTORIAL - 1

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

Asymptotic notation is used to describe the running time of an algorithm - how much time an algorithm takes with a given input (n).

Different Asymptotic notations are -

1) Big - O Notation → Describes the worst case running time of a program

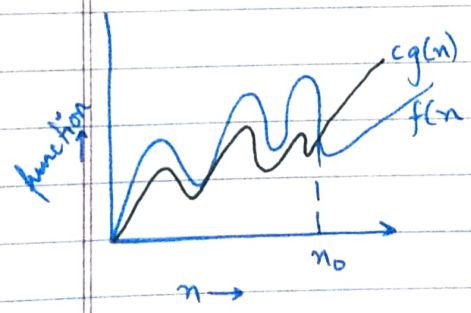
example → O(logn) - Binary Search

2) ~~Big O Notation~~ → $f(n) = O(g(n))$

iff $f(n) \leq cg(n) \quad \forall\ n > n_0$

for some constant, $c > 0$

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



2) Big Omega (Ω) Notation → $f(n) = \Omega(g(n))$

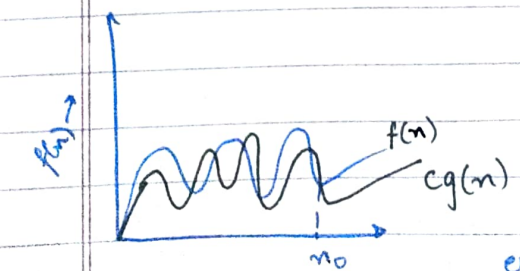
$g(n)$ is tight lower bound of function $f(n)$

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

iff, $f(n) \geq cg(n)$

$\forall\ n \geq n_0$

for some constant, $c > 0$



example → ~~constant~~

$\Omega(1)$ → Binary Search

Date.
Page No.

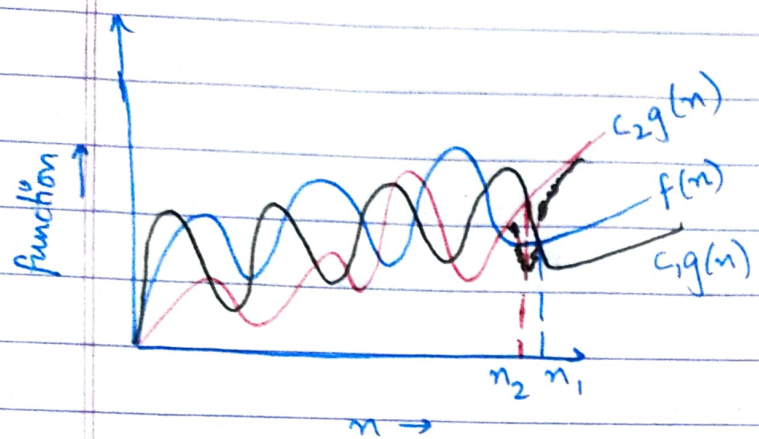
Big theta (Θ) Notation $\longrightarrow$ $f(n) = \Theta(g(n))$

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

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

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

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



Date
Page No.

4 small o Notation

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

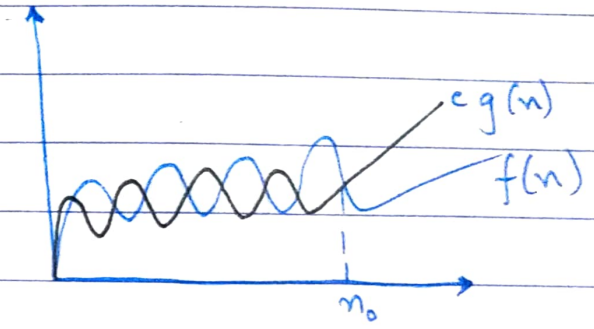
$g(n)$ is upper bound ~~#~~ function of $f(n)$

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

when $f(n) < c.g(n)$

$\forall\ n > n_0$

and $\forall\ c > 0$



5 small omega (ω) Notation

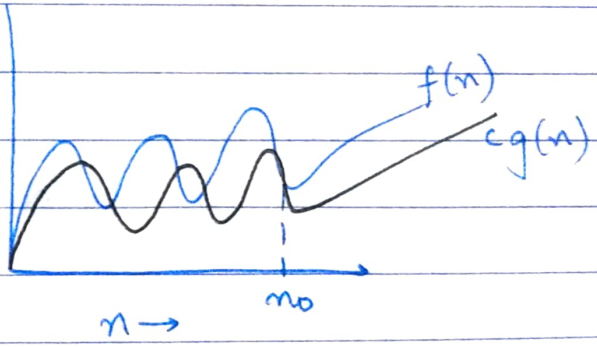
$f(n) = \omega\, g(n)$

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

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

$\forall\ n > n_0$

and $\forall\ c > 0$



2) What should be the time complexity of -

```
for(i=1 to n) {i= i*2;}
```

~~i = 1, 2, 4, ... n~~

```
for (i to n)        // i = 1, 2, 4, 8 ... n.
    i = i*2;        // O(1)
```

$$\sum_{i=1}^{n} (i*=2) \qquad 1+2+4+\dots n$$

GP k_{th} value $\rightarrow t_k = ar^{k-1}$

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

$$= 2^{k-1}$$

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

$$2n = 2^k$$

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

$$\log_2 2 + \log_2 n = k$$

$$\boxed{k = 1 + \log_2 n}$$

$$O(k) \Rightarrow O(1 + \log_2 n)$$

$$= \boxed{O(\log_2 n)}$$

Date
Page No.

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

$T(n) = 3T(n-1)$ — ①

put $n = n-1$ in eqⁿ ①

$T(n-1) = 3T(n-2)$ — ②

put value of eqⁿ ② in ①

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

$T(n) = 9T(n-2)$ — ③

put $n = n-2$ in eqⁿ ①

$T(n-2) = 3T(n-3)$ — ④

put eqⁿ ④ in ③

$T(n) = 9[3T(n-3)]$

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

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

put $n-k=0$

$n = k$

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

$= 3^n$

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

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

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

$T(n) = 2T(n-1) - 1$ —①

put $n = n-1$ in ①

$T(n-1) = 2T(n-2) - 1$ —②

put ② in ①

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

$T(n) = 4T(n-2) - 2 - 1$ —③

put $n = n-2$ in ①

~~$T(n)$~~ $T(n-2) = 2T(n-3) - 1$ —④

put ④ in ③

$4(2T(n-3) - 1) - 2 - 1$ —⑤

$T(n) = 8T(n-3) - 4 - 2 - 1$ —⑤

$2^k T(n-k) - [1 + 2 + 4 + \ldots k][2^{k-1} + 2^{k-2} + \ldots + 1]$

put $n-k=0$

$n=k$

$2^n T(0) - [n(n+1)]$

$2^n - n^2 + n$

$O(2^n - n^2)$

$2^k T(n-k) - [1(2^{k+1} - 1)]$

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

put $n-k=0$

$k=n$

$2^n T(0) - 2^n + 1$

$O(1)$

$\Rightarrow O(2^n)$

5 What should be the time complexity of-

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

i = 1 2 3 4 + ...

s = 1 + 3 + 6 + 10 + 15 + ... n

Sum of $s = 1+3+6+10+\ldots+n$ — (1)

also $s = 1+3+6+\ldots T_{n-1} + T_n$ — (2)

from (1) - (2)

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

$T_k = 1+2+3+\ldots k$

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

for k iterations

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

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

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

$O(k^2) <= n \quad \Rightarrow k = O(\sqrt{n})$

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

6 void function (int n)

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

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

$1+2 \ldots \sqrt{n}$

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

$$O\left(\frac{\sqrt{n}(\sqrt{n}+1)}{2}\right)$$

$$O\left(\frac{n+\sqrt{n}}{2}\right) \Rightarrow \boxed{O(n)}$$

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++;
}
```

i	j	k
$n/2$	$\log_2 n$	$\log n * \log n$
$n/2+1$	$\log n$	$\log n * \log n$
.	.	.
.	.	.
.	.	.
n	$\log n$	$\log n * \log n$

$$\left[\frac{n}{2} + \left(\frac{n}{2}+1\right) + \left(\frac{n}{2}+2\right) + \ldots n\right] \log n \log n$$

$$\left[\left(n-\frac{n}{2}\right)\left(\frac{n}{2}\right) + \left[1+2+\ldots\frac{n}{2}\right)\right] \log n * \log n$$

$$\left[\frac{n^2}{4} + \frac{n}{2}\frac{\left(\frac{n}{2}+1\right)}{2}\right] (\log n)^2$$

$$n^2 + n(n+2) \quad (\log n)^2$$

$$\frac{32}{}$$

~~$O(n^4 (\log n)^2)$~~

$$n^2 + n^2(\log n)^2$$

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

8 Time complexity of -

```
function (int n)
{
    if (n==1)
        return 1;                          // O(1)
    for (i=1 to n)                         // i = 1,2,3,4,.... n ⇒ O(n)
    {
        for (j=1 to n)                     // j = 1,2,3,4,.... n² ⇒ O(n²)
        {
            printf("*");                   // O(1)
        }
    }
    function (n-3);
}
```

$T(n) = T(n/3) + n^2$

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

$c = \log_3 1 = 0$

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

$T(n) = \theta(n^2)$

Date
Page No.

9 Time complexity of

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

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

$$\sum_{j=n}^{1} n + \frac{n}{2} + \frac{n}{3} + \ldots + 1$$

$$n\left[1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{n}\right]$$

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

Date
Page No.

10 For the functions n^k and c^n, what is the asymptotic relationship between these functions? Assume that $k>=1$ and $c>1$ are constants. Find out the value of c and n_0 for which relation holds.

$f(n) = n^k$

$g(n) = c^n$; $k>=1$, $c>1$

$n^k = O(c^n)$

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

as $n^k \leq a c^n$

$\forall n \geq n_0$ and some constant $a>0$

for $n_0 = 1$

$c = 2$

$1^k \leq a_2$

$\boxed{n_0 = 1}$ and $\boxed{c = 2}$