Satyam Rawat
2017006
G

DAA

Tutorial 1

Q1) What do you understand by Asymptotic notations? Define different Asymptotic notation with examples.

Ans Asymptotic notations are the mathematical notations used to describe the running time of an algorithm.

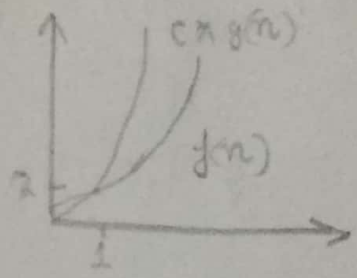
① **Big O Notation (O)** → It represents the upper bound of algorithm.

$f(n) = O(g(n))$ if $f(n) \leq c * g(n)$ $\forall$ $n \geq n_0, c > 0$

For eg $f(n) = n^2 + n$ $c.g(n) = 2n^2$

then $f(0) = 0 = g(0)$

$f(1) = 2 \quad g(1) = 2$



~~② Big Omega (Ω)~~ $f(2) = 4+2 = 6$ $\quad g(2) = 8$

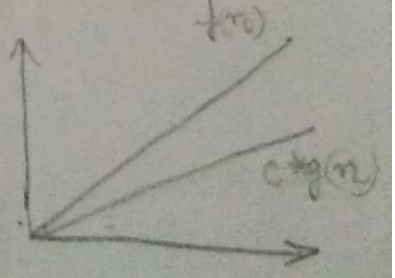
$\Rightarrow n_0 = 1$ $\quad f(3) = 12 \quad g(3) = 18$

$\therefore f(n) = O(g(n)) = O(n^2)$

② **Big Omega (Ω)** → It represents the lower bound of algorithm.

$f(n) = \Omega(g(n))$ if $f(n) \geq c * g(n)$ $\forall$ $n \geq n_0, c > 0$

For eg $f(n) = n$ $\quad g(n) = \frac{1}{2}n$



then $f(0) = 0 = g(0)$ $\Rightarrow n_0 = 0$

$f(1) = 1 \quad g(1) = \frac{1}{2}$

$f(2) = 2 \quad g(2) = 1$

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

3 Big Theta → (Θ)

It represents Upper & lower bound of algorithm.

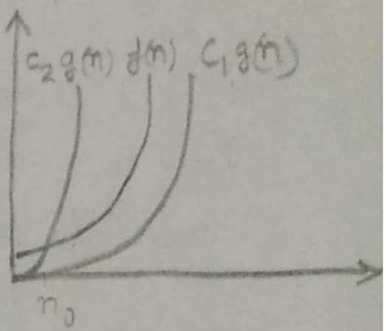
$f(n) = \Theta(g(n))$ if $c_1 g(n) \le f(n) \le c_2 g(n)$

$\forall c_1, c_2 > 0, \; n \ge \max(n_1, n$

For eg $f(n) = n^2+2$

$c_2 \cdot g(n) = 3n^2$

$c_1 \cdot g(n) = n^2$



$n=0$ $\quad f(0)=2 \quad c_1 g(0) = 0 \quad c_2 \cdot g(0)=0$

$n=1$ $\quad f(1)=3 \quad c_1 g(1) = 1 \quad c_2 \cdot g(1)=3 \Rightarrow n_0 = 1$

$n=2$ $\quad f(2)=6 \quad c_1 g(2) = 4 \quad c_2 \cdot g(2)=12$

$\Rightarrow 1 \cdot n^2 \le f(n) \le 3n^2$

Similarly we have Little (o), Little (θ) Theta, Little Omega(Ω)

Q1) What should be time complexity of :

for (i=1 to n) i*=2 ;

soln) for (i=1 to n) $\Rightarrow$ $i = 1, 2, 4, 8, 16$

i* =2

Clearly this is a GP

$\boxed{a_n = a r^{n-1}}$

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

$\log_2 n = k-1$

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

$\therefore = O(\log n)$

Q3 $T(n) = \begin{cases} 3T(n-1) & n > 0 \\ 1 & n \le 0 \end{cases}$

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

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

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

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

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

$\Rightarrow T(n) = 3^n T(0) = 3^n$.

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

Q4.) $T(n) = \begin{cases} 2T(n-1)-1 & n > 0 \\ 1 & n \le 0 \end{cases}$

$T(0) = 0$

$T(1) = 2T(0) - 1 = 1$

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

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

$\Rightarrow T(n) = 1 \Rightarrow O(1)$

Q5.)

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

	i	s
Initially	1	1
	2	1+2
	3	1+2+3
	:	:

This is an AP :

$\cancel{a_n = a + (n-1)d}$

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

$\cancel{k = n}$

$S_m = 1 + (1+2) + \dots T(m-1) + T_m$ — ①

$S_m = \quad 1 + (1+2) \dots + T(m-1) + T(m)$ — ②

① − ②,

$T_m = 1 + 2 + \dots m = m\left(\frac{m+1}{2}\right)$.

$\therefore$ m^{th} term is n,

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

$\Rightarrow 2n = k^2 + k$

$\Rightarrow k^2 + k - 2n = 0$

$\Rightarrow k \simeq \sqrt{n} = O(\sqrt{n})$

26.)
```
void funct (int n)                1
{ int i, c = 0 ;                  1
  for (i=1 ; i*i <= n ; i++)
      c++ ;
}
```

i	i^2
1	1
2	4
3	9

Loop Terminates when

$i^2 > n \Rightarrow k^2 > n$

$\Rightarrow k > \sqrt{n}$

$= O(\sqrt{n})$

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

Loop 1 →

$i = \frac{n}{2}$ to n, $i{+}{+}$

$= \frac{n}{2}$ times

Loop 2 →

$j = 1$ to n, $j *= 2$

$j = 1, 2, 4, 8$

$\Rightarrow \log_2 n$ times

Loop 3 →

$k = 1$ to n, $k *= 2$

$\Rightarrow \log_2 n$ times

Total complexity

$= \frac{n}{2} \times (\log_2 n)^2$

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

Q8) funct (int n)

```
{ if (n==1) return;          - 1
  for (i= 1 to n)
    for(j=1 to n)
      print('*');            - n^2
  funct(n-3);                - T(n-3)
}
```

Soln $T(n) = T(n-3) + n^2$ with $T(1) = 1$

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

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

$T(n) = T(n-3k) + kn^2$

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

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

Q9) funct (int n)

```
{ for (i=1 to n)                    Loop 1
                                    - n
    for(j=1 to n ; j+=i)
                                    Loop 2
      print("*");
}
```

1	$j = 1, 2, 3 \ldots n$	$= n$ times
2	$j = 1, 3, 5 \ldots n$	$= \frac{n+1}{2}$ times $\simeq \frac{n}{2}$
3	$j = 1, 4, 7 \ldots n$	$= \frac{n+2}{3}$ times $\simeq \frac{n}{3}$
⋮		
n	$j = 1$	$= 1$ times $\simeq \frac{n}{n}$

$$\left(n + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n}\right) = n\left(\frac{1}{1}+\frac{1}{2}+\frac{1}{3}\cdot\frac{1}{n}\right) = n\log n$$

$= O(n \log n)$

10) For n^k & c^n what is Asymptotic relationship between them? (Assume $k >= 1$ & $c > 1$ are constant.) Find c & n_0 for which relation holds.

Soln. $f_1(n) = n^k$ $\qquad f_2(n) = c^n$

~~$f_1(1) = 1^k = 1$ $\qquad f_2(1) = c^1 = c \Rightarrow f_2 > f_1$ for $n=1$~~

~~$f_1(2) = 2^k$ $\qquad f_2(2) = c^2 \Rightarrow f_1 > f_2$ if $n > c$, $k > 2$~~

~~$f_1(3) = 3^k$ $\qquad f_2(3) = c^3 \Rightarrow f_1 > f_2$ if $n > c$, $k > 3$~~

~~$\Rightarrow n_0 = 2$ if $c \leq n$ and $k \geq n$~~

$\Rightarrow$ Let $k = 2$ & $c = 2$

$\Rightarrow f_1(n) = n^2$ $\qquad f_2(n) = 2^n$

$n=2$ $\quad f_1(2) = 4$ $\qquad f_2(2) = 4$

$n=3$ $\quad f_1(3) = 9$ $\qquad f_2(3) = 8$

$n=4$ $\quad f_1(4) = 16$ $\qquad f_2(4) = 16$

$n=5$ $\quad f_1(5) = 25$ $\qquad f_2(5) = 32$

$n=6$ $\quad f_1(6) = 36$ $\qquad f_2(6) = 64$

$\Rightarrow$ if $k=2, c=2, n_0 = 4$ for which

$f_1(n) \leq c_1 * f_2(n)$

ie $\boxed{n^k = O(c^n)}$