Design And Analysis of Algorithm

Tutorial -1

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

Ans 1. Asymptotic notations are those notations that describing the limiting behaviour of a func^n.

There are 3 different types of notations:-

→ Big oh (O).

→ Big (Ω)

→ Big (θ)

Big (o)

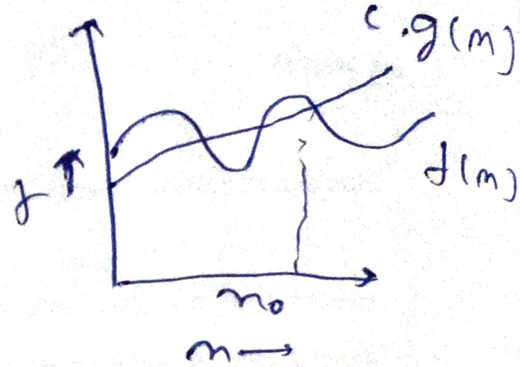
Big(O) notation gives an upper bond for a func^n f(n) to within a constand factor.

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

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

$f(n) \le c.g(n)$

$\forall\ n \ge n_0$



Ex:- for (i=1 ; i≤n ; i++)

{

sum+= i;

}.

Big Omega Notation

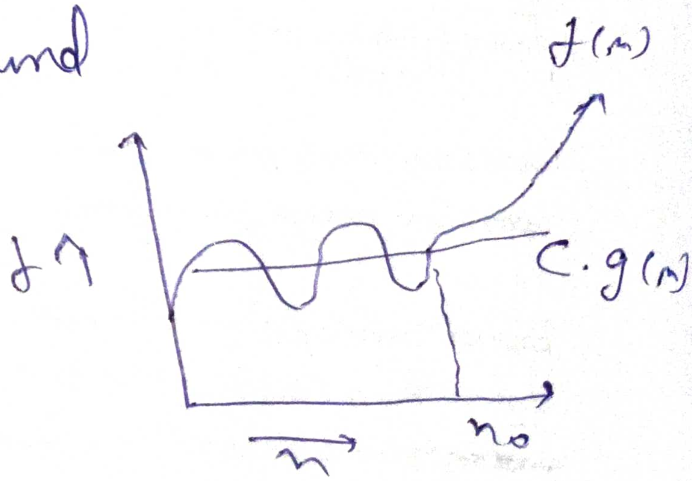
$f(n) = \Omega(g(n))$.

$g(n)$ is "tight" lower bound

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

iff $f(n) \ge c.g(n)$

$\forall\ n \ge n_0$.



Big Theta (Θ)

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

$g(n)$ is both "tight" upper & lower bond of $f(n)$

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

iff $c_1 g(n) \le f(n) \le c_2 g(n)$

$\forall\ n \ge \max(n_1, n_2)$.

Small (o)

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

g(n) is upper bond of the f(n)

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

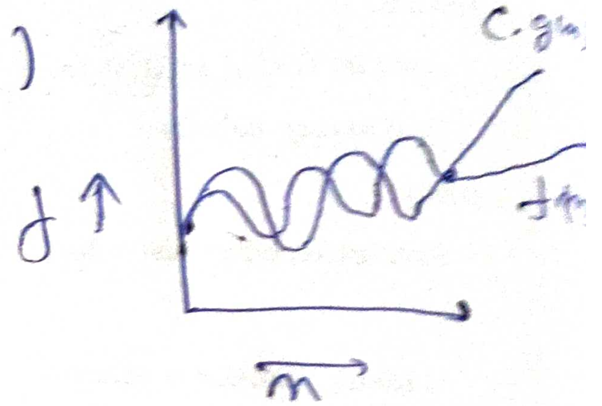
g(n) is "upper" bound of ~~f(n)~~ f(n)

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

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

$\forall \; n \geq n_0$

$\forall \; c > 0.$



Small omega (ω)

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

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

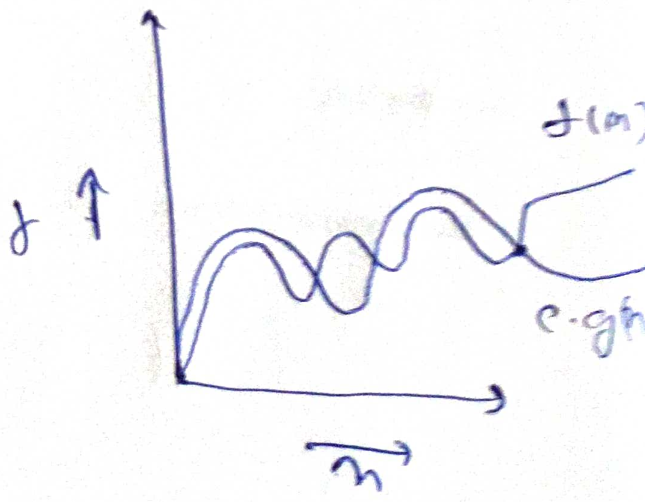
$$f(n) = \omega.(g(n))$$

when

$f(n) > . c.g(n)$

$\forall \; n > n_0.$

if $\forall \; c > 0.$



Q2. what should be time complexity.

for (i = 1 to m) { i = i * 2 }.

for (i = 1 to m)   // $i = 1, 2, 4, 8 \ldots m$.

{ i = i * 2 }.   // O(1).

Ans $\sum_{i=1} 1+2+4+8+\ldots m$.

k^{th} term of GP = $T(n) = a r^{k-1}$.

$m = 1 * 2^{k-1}$.

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

$m = \frac{2^k}{2}$.

$2m = 2^k$

$\log_2 (2m) = k$

$k = 1 + \log_2(m)$

$O(\log_2(m))$.

Q3 $T(m) = 3T(m-1)$

put $m = m-1$ in 1

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

put $m = m-2$ in 1

$T(m-2) = 3T(m-3)$ — 3

substitute 3 in 2

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

$T(m) = 3^{k} T(m-k)$

put $m - k = 0$

$K = m - 0$

$T(m) = 3^{m-0}(T(0)) = \frac{3^{n}}{3^{0}}$.

$O(3^{n})$.

Q4

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

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

put $m = n-1$ in 1

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

$T(m) = 4T(m-2) - 2 - 1$ — 2

put $m = m-2$ in 1

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

Substitute ②

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

$T(n) = 8T(n-3) - 4 - 2 - 1$

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

$n - k = 0$

$n = k$

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

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

$= O(1)$.

Q5

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

$S = 1, 3, 6, 10, 15 \dots$

$T_n = Ak^2 + Bk + C$

Put $k = 1$

$$A + B + C = 1 \quad -(1)$$

put $k = 2$

$$4A + 2B + C = 3 \quad -(2)$$

put $k = 3$

$$9A + 3B + C = 6 \quad -(3)$$

$$A = \frac{1}{2} \quad B = \frac{1}{2} \quad C = 0$$

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

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

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

Q6 void function(int n)

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

for(i = 1 ; i * i <= n ; i++)

$O(1)$

Values of i = $1, 4, 9, 16 \ldots (\sqrt{n})^2$.

$Ak^2 + Bk + C = k$

$k = 1$

$A + B + C = 1$ — (1)

$k = 2$

$4A + 2B + C = 4$ — (2)

$k = 3$

$9A + 3B + C = 9$ — (3)

Solving (1) (2) (3)

$A = 1, B = 0$ & $C = 0$

$m = Ak^2$

$m = k^2$

$k = \sqrt{m}$

$(O\sqrt{n})$.

Q7 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 n	log n × log n
n/2 + 1	log n	+ log n × log n
⋮	⋮	⋮
n	log n	log n × log n

$O(n \times (\log_2 n)^2)$

Q8

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

$(n-3), (n-6), (n-9), \ldots (1)$

$a = n-3, \quad d = n-6-n+3 = -3$

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

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

$3k = n - 1$

$k = \frac{n-1}{3} = O(n \times n^2)$

$O(n^3)$

Q9 void function (int n)

for (i = 1 to n)

{ for j = 1; j < n; j += i)

{ printf ("*");

}

}

for (i = 1, j = n times

i = 2, j = n/2 times

i = 3, j = n/3 times

⋮

i = k, j = n/k times

i = n, j = n/n times

$n * (1 + 1/2 + 1/3 + \dots 1/n)$

$H_n = 1 + 1/2 + \dots \frac{1}{n}$

$= \sum_{k=1} 1/k$

$= \log(n) + O(1)$.

$O(n \log(n))$

Q10 $f(n) = n^k \quad g(n) = c^n$.

where $k = 1$ & $c > 1$

Let $k = 1$ ~~for~~ & $c = 2$.

$f(1) = (1)^1$

$g(1) = (2)^1$

$f(1) < g(1)$

$f(2) = (2)^1 \quad g(2) = (2)^2 = 4$

$f(2) < g(2)$

Satisfy O notation $f(n) \leq cg($

$f(n_0) = c \cdot g(n_0)$

$n^k = c_0 \cdot c^{n_0}$

$k = 1, c = 2$

$n_0^1 = c_0 \cdot 2^{n_0}$

$\left(\frac{n_0}{c_0}\right) = (2)^{n_0}$

~~contradiction~~

Comparing

$n_0 = 1$

$\frac{n_0}{c_0} = 2$

$\frac{1}{2} = c_0$

$f(n) \leq 0.5g(n)$

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