Tutorial -1

Name - Kulmeet Singh
Section - F
Roll No. - 13
University Roll No. - 2016827

Q. What do you understand by Asymptotic notation, define different asymptotic notation, define ~~Diff~~ with example.

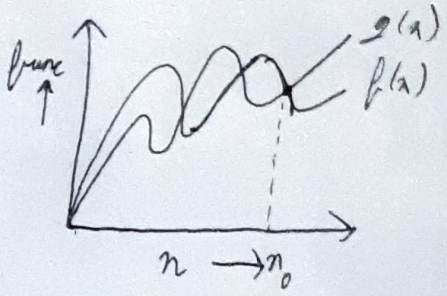
i) **Big O(n) →**

$f(n) \Rightarrow O(g(n))$

if $f(n) \leq g(n) \times c \quad \forall \; n \geq n_0$

for some constant, $c > 0$

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



e.g. $f(n) \Rightarrow n^2 + n$

$g(n) \Rightarrow n^3$

$n^2 + n \leq c * n^3$

$n^2 + n = O(n^3)$

ii) **Big Omega (Ω) →**

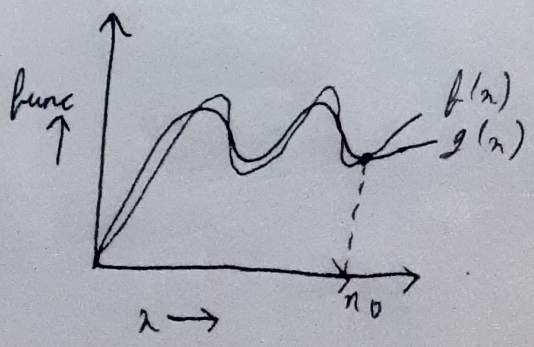
When $f(n) = \Omega(g(n))$

means $g(n)$ is "tight" lowerbound of $f(n)$ i.e. $f(n)$ can go beyond $g(n)$

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

if and only if $f(n) \geq c.g(n)$

$\forall \; n_2 > n_0$ and c = constant > 0



e.g. $f(n) = n^3 + 4n^2$

$g(n) = n^2$

i.e. $f(n) \geq c * g(n)$

$n^3 + 4n^2 = \Omega(n^2)$

iii) Big Theta (Θ) –

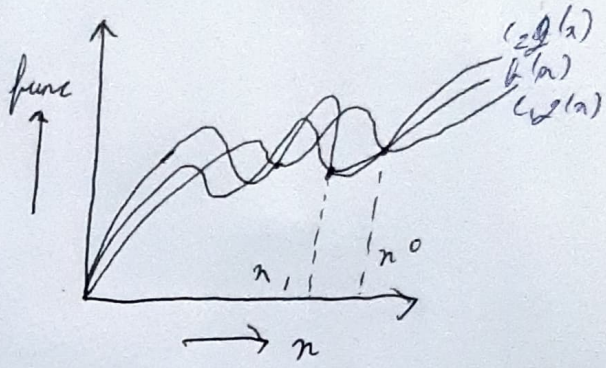
When $f(n) = \Theta(g(n))$ gives the tight upperbound and lowerbound both. i.e. $f(n) = \Theta(g(n))$

if and only if

$c_1 * g(n_1) \leq f(n) \leq c_2 * g(n_2)$

for all $n \geq \max(n_1, n_2)$, same constant

$c_1 > 0$ & $c_2 < 0$



i.e. $f(n)$ can never go beyond $c_2 g(n)$ and will never come down of $c_1 g(n)$

Eg – $3n+2 = \Theta(n)$ as $3n+2 \geq 3n$ &

$3n+2 \leq 4n$ for n_1, $c_1 = 3$, $c_2 = 4$ & $n_0 = 2$

iv) Small o (o) →

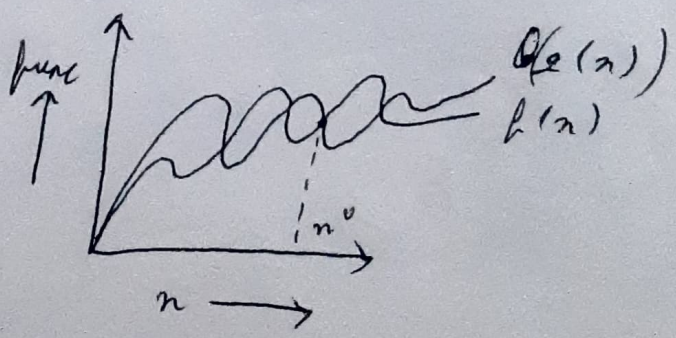
When $f(n) = o(g(n))$ gives the upper bound

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

if and only if

$f(n) < c * g(n)$

$\forall\ n > n_0$ & $n > 0$



Eg – $f(n) = n^2$; $g(n) = n^3$

$f(n) < c * g(n)$

$n^2 = o(n^3)$

ω) Small omega (ω) →

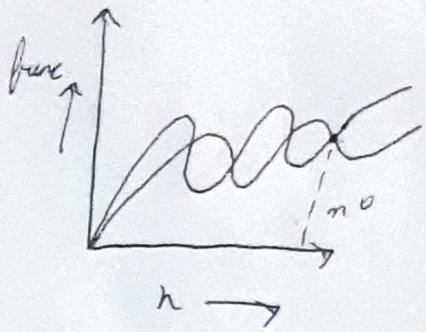
It gives the 'lower bound' i.e.,

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

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

if and only if $f(n) > c * g(n)$

$\forall\ n > n_0$ & same constant, $c > 0$.



Q2. What should be time complexity of:

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

Sol:- for $i \Rightarrow 1, 2, 4, 8, 8, \ldots\ldots$ n times

i.e. Series is a GP

So, $a = 1$ $r = 2$

K^{th} value of GP:

$t_k = a\, r^{k-1}$

$t_k = 1\,(2)^{k-1}$

$2^n = 2^k$

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

$\log_2 2 + \log_2 n = k$

$\log_2(n+1) = k$ (Neglecting '1')

So, Time complexity $\boxed{T(n) \Rightarrow O(\log n)}$

Q3, $T(n) = \begin{cases} 3T(n-1) & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$

Sol:- $T(n) \Rightarrow 3T(n-1)$ — (1)

$T(n) = 1$

put $n = n-1$ in (1)

$T(n-1) = 3T(n-2)$ — (2)

put (2) in (1)

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

$T(n) = 9T(n-2)$ — (3)

put $n = n-2$ in eq. (1)

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

put in (3)

$T(n) = 27T(n-3)$ — (4)

Generalising series,

$T(k) = 3^k T(n-k)$ — (5)

for k^{th} terms, Let $n-k = 1$ (Base case)

$k = n-1$

put in (5)

$T(n) = 3^{n-1} T(1)$

$T(n) = 3^{n-1}$ (neglecting 3.)

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

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

$$T(n) = 2T(n-1) - 1 \quad \text{— (1)}$$

put $n = n-1$

$$T(n-1) = 2T(n-2) - 1 \quad \text{— (2)}$$

put in (1)

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

$$= 4T(n-2) - 2 - 1 \quad \text{— (3)}$$

put $n = n-2$ in (1)

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

Put in (1)

$$T(n) = 8T(n-3) - 4 - 2 - 1 \quad \text{— (4)}$$

Generalizing series,

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

→ k th term

Let $n - k = 1$

$k = n - 1$

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

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

i.e. Series in GP

$a = 1/2$, $r = 1/2$

So, $T(n) = 2^{n-1} \left(1 - \left(1/2 \left(\frac{1 - (1/2)^{n-1}}{1 - 1/2} \right) \right) \right)$

$$= 2^{n-1} \left(1 - 1 + (1/2)^{n-1} \right)$$

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

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

Q5. What should be the time complexity of

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

Sol:- i = 1 2 3 4 5 6 ...

$s = 1+3+6+10+15+\ldots$

Sum of $s = 1+3+6+10+\ldots+n \rightarrow (1)$

Also $s = 1+3+6+10+\ldots+T_{n-1}+T_n \rightarrow (2)$

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

$T_k = 1+2+3+4+\ldots+k$

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

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

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

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

$O(k^2) <= n$

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

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

Q6. Time complexity of

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

Sol:- As $i^2 = n$

$i = \sqrt{n}$

$i = 1, 2, 3, 4, \ldots \sqrt{n}$

$\sum_{i=1}^{} 1+2+3+4+\ldots \sqrt{n}$

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

$$T(n) = \frac{n + \sqrt{n}}{2}$$

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

Q7. Time complexity of

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

Sol:- Since, for $k = k^2$

$k = 1, 2, 4, 8, \ldots, n$

∴ Series is in GP

So, $a = 1, r = 2$

$$\frac{a(r^n-1)}{a(r-1)} = \frac{1(2^k-1)}{1} \Rightarrow n = 2^k - 1$$

$$n+1 = 2^k$$

$$\log_2(n) = k$$

i	j	k
1	log(n)	log(n) * log(n)
2	log(n)	log(n) * log(n)
⋮	⋮	⋮
n	log(n)	log(n) * log(n)

T.C => O(n * log n * log n)

=> O(n log²(n))

Q8. Time complexity of

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

Sol:- for (i = 1 to n)

We set j = n times every turn

∴ i * j = n²

kth, Now, $T(n) = n^2 + T(n-3)$;

$T(n-3) = (n-3)^2 + T(n-6)$;

$T(n-6) = (n-6)^2 + T(n-9)$;

and $T(1) = 1$;

Now, substitute each value in T(n)

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \ldots + 1$

Let $n - 3k = 1$

$k = (n-1)/3$ total times $= k+1$

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \ldots + 1$

$T(n) \geq k n^2$

$T(n) \geq (k-1)/3 * n^2$

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

Q9. Time complexity of →

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

Sol:- for $i=1$ $j = 1+2+\ldots (n \geq j+i)$

$i=2$ $j = 1+3+5+\ldots (n \geq j+i)$

$i=3$ $j = 1+4+7+\ldots (n \geq j+i)$

n^{th} term of AP is

$T(n) = a + d * n$

$1 + (n) = 1 + d * n$

$(n-1)/d = n$

for $i=1$ $(n-1)/1$ times

$i=2$ $(n-1)/2$ times

$i = n-1$

we get, $T(n) = i_1 j_1 + i_2 j_2 + \ldots i_{n-1} j_{n-1}$

$= \frac{n-1}{2} + \frac{n-2}{2} + \frac{n-3}{2} + \ldots 1$

$= n + \frac{n}{2} + \frac{n}{3} + \ldots \frac{n}{n-1} - n \times 1$

$= n\left(1 + 1/2 + 1/3 + \ldots 1/n-1\right) - n \times 1$

$< n \log n - n + 1$

Since $\int 1/x = \log x$

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

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

Sol:- For given n^k and c^n

Relationship b/w n^k & c^n is

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

$$n^k \leq a(c^n)$$

$\forall\ n \geq n_0$ & constant, $a > 0$

for $n_0 = 1$; $c = 2$

$\Rightarrow 1^k < a^2$

$\Rightarrow n_0 = 1$ & $c = 2$