DESIGN and ANALYSIS of ALGORITHM

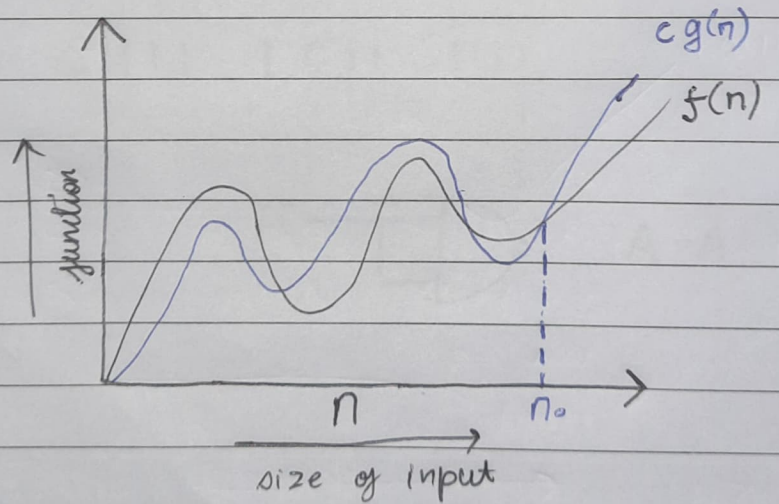
Tutorial - 1

Q) 1 What are asymptotic notations. Define different asymptotic notation with examples.

Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

TYPES

1. Big O



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

iff

$f(n) \leq c\, g(n) \quad \forall\, n \geq n_0$

for some constant, $c > 0$

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

2. Big Ω (Omega)

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

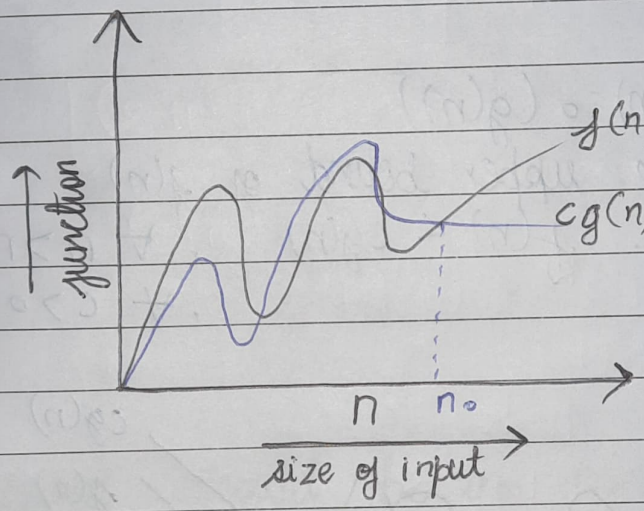
$g(n)$ is the tight lower bound of function $f(n)$

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

iff

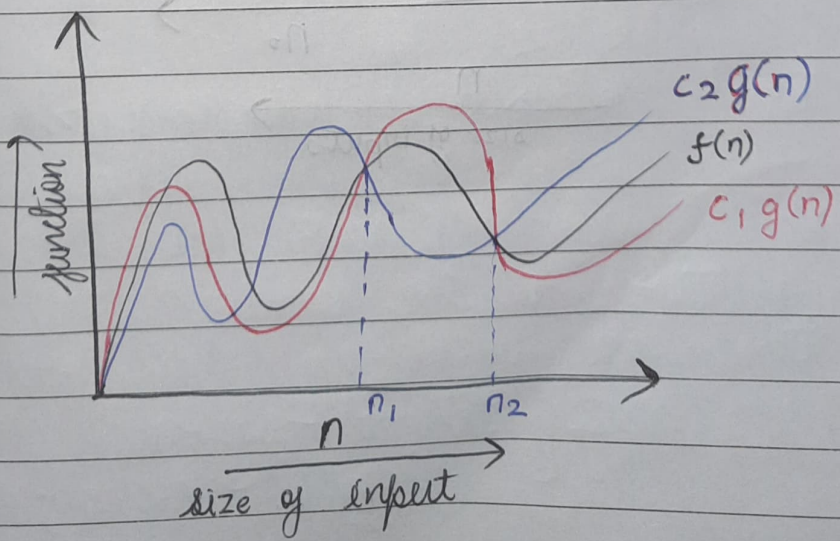
$f(n) \geq c\, g(n)$, $\forall\ n \geq n_0$

$c > 0$



3. Theta (θ)

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



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

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

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

iff

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

$$, \forall \; n \geq \max(n_1, n_2)$$

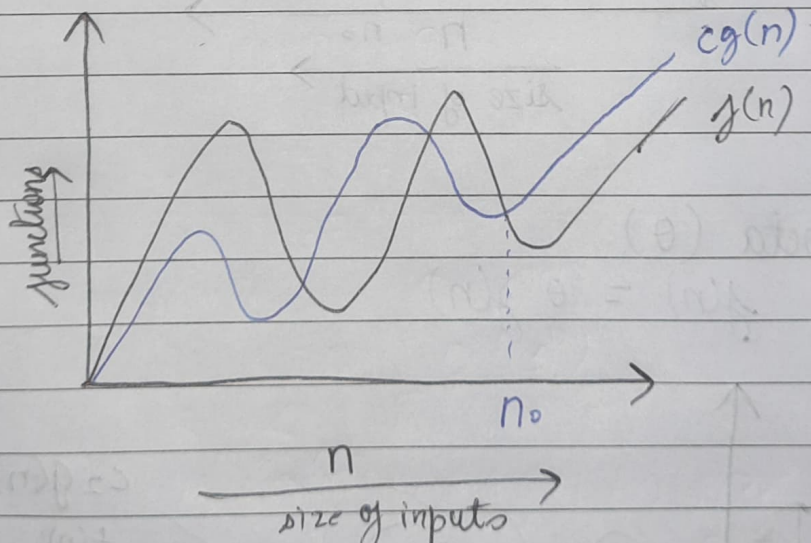
4. Small O

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

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

$$f(n) < cg(n) \quad , \forall \; n > n_0$$

$$, \forall \; c > 0$$



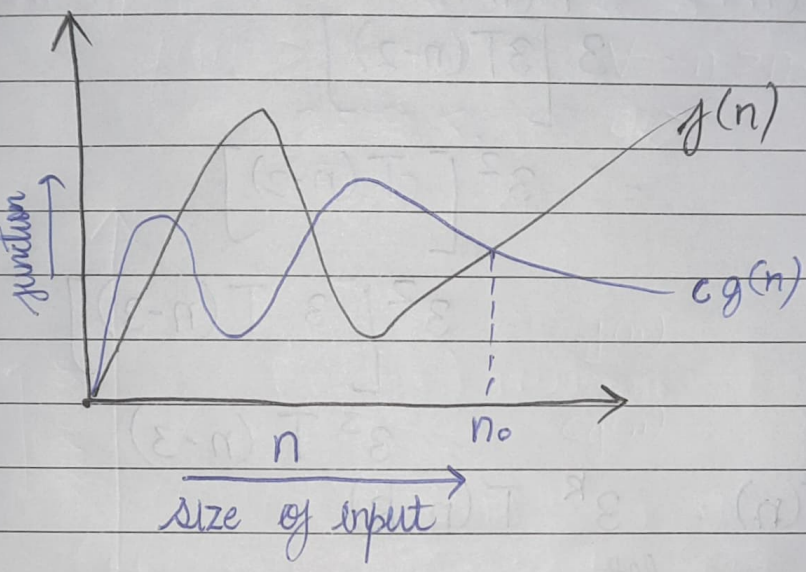
5. Small Omega (ω)

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

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

$f(n) > cg(n)$, $\forall$ $n > n_0$

$c > 0$



Q2) What should be the complexity of

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

series for value of i

1, 2, 4, 8 .... i ..

$2^0, 2^1, 2^2, 2^3 \ldots\ldots 2^k$

k^{th} term of the series

$n = 2^k$

$\log_2 n = k \log_2 2$

$k = \log_2 n$

$\therefore$ complexity is $O(\log_2 n)$

Q3. $T(n) = \{ 3T(n-1) \}$ if $n > 0$,
else 1

$T(0) = 1$

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

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

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

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

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

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

let

$n - k = 0$

$n = k$

$T(n) = 3^n$

complexity = $O(3^n)$

Q4. $T(n) = 2T(n-1) - 1$, $n > 0$

1 , otherwise

$T(0) = 1$

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

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

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

$= 2^2 [2T(n-3) - 1] - 2 - 1$

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

for k terms

$T(n) = 2^k [T(n-k)] - 2^{k-1} - \dots - 1$

let

$T(n-k) = 1$

$n - k = 0$

$n = k$

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

$= 2^n - \left[\frac{1(2^n - 1)}{2-1}\right]$

$= 2^n - 2^n + 1$

$= 1$

$O(1)$

Q5)

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

series for 5 ↴

1, 2, 4, 7, 11, 16, 22 . . . .

1, 2, 3, 4, 5, 6, . . ← series for i

$S = S + i$

$S = S + (1 + 2 + 3 + 4 \ldots)$

$\therefore \quad n > \frac{k(k+1)}{2}$

if 'k' is the total no. of terms

$2n > k^2 + k$

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

Q 6

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

series for i

$1, 2^2, 3^2, 4^2, 5^2, \ldots k^2$

$k^2 <= n$

$k <= \sqrt{n}$

$\therefore$ complexity is $O(\sqrt{n})$

Q7

```
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 starts from $\frac{n}{2}$ to n
∴ the 1st loop runs for $(n/2)$ times

j loop runs for $o(\log_2 n)$ times
as

$1, 2, 2^2 \ldots 2^k$

$2^k <= n$

$k <= \log_2 n$

$= O(\log_2 n)$

similarly k loop runs $O(\log_2 n)$ times

i	j	k
$n/2$	$\frac{n}{2} n \log_2 n$	$(\log_2 n)^2$
$n/2+1$	$\frac{n}{2} n \log_2 n$	$(\log_2 n)^2$
$n/2+2$	⋮	⋮
⋮	⋮	⋮
total terms $= \frac{n}{2}$	total terms $\frac{n}{2} \log_2 n$	$\frac{n}{2} (\log_2 n)^2$ terms

$$O(n \lfloor \log_2 n \rfloor^2)$$

Q 8

```
function(int n)          — T(n)
{
  if (n == 1)            — O(1)
    return;
  for (i = 1 to n)       — O(n)   } O(n^2)
  {
    for (j = 1 to n)     — O(n)   }
      printf("*");
  }
  function(n-3);         — T(n-3)
}
```

recurrence relation

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

$$T(n) = T(n-3) + n^2$$
$$T(n-6) + n^2 + n^2$$
$$T(n-9) + n^2 + n^2 + n^2$$

generalizing for K terms

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

let $T(n-3k) = 1$

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

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

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

Q9

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

series for j

1, 2, 3, 4 ... n

k = n;

i	j
1	n
2	n
⋮	⋮
n	n

n terms — n terms

$\therefore O(n^2)$

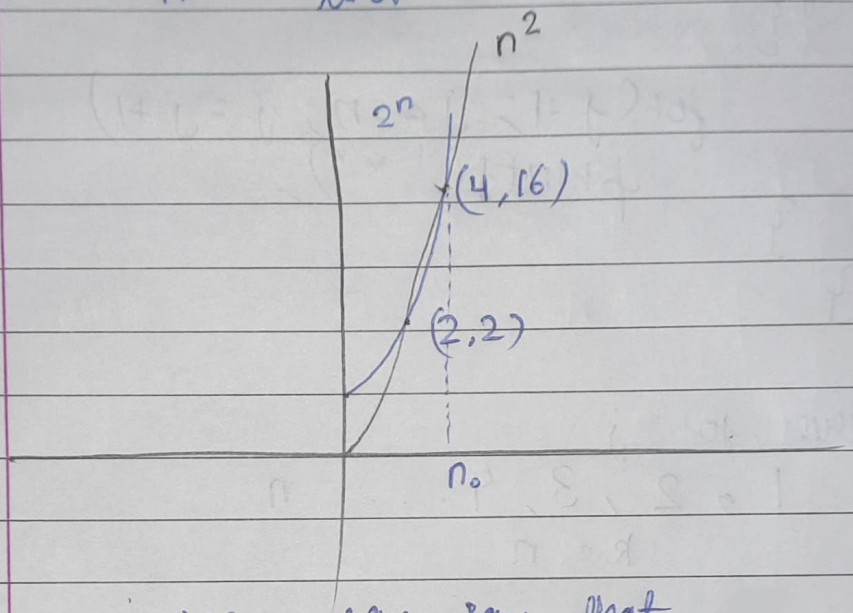
Q10 For the function, $n \hat{} k$ and $c \hat{} n$, what is the asymptotic relationship between these functions

Assume that $k >= 1$ and $c > 1$ are constants

Find out the value of c and n_0 for which relation holds.

let R = c = 2

n^2 and 2^n



∴ we can say that

$n^2 \leq 2^n$ for $n \geq n_0$

$n^2 \leq 1(2^n)$

∴ $n^2 = O(2^n)$

OR

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

∴ $c = 2$ and $n_0 = 4$