Q1 What do you understand by Asymptotic notations. Define different Asymptotic Notation with example?

Ans1 Asymptotic Notation are the mathematical Notation used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value. Asymptotic Notation is a way to compare function that ignores constant fact and small input sizes.

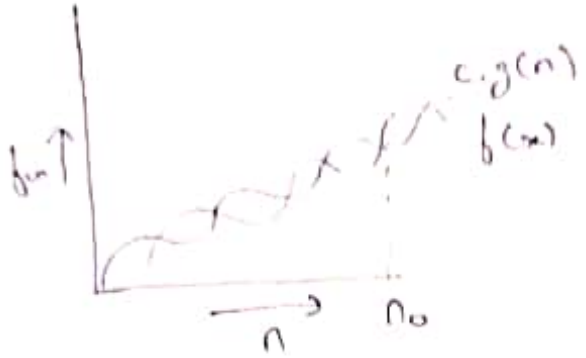
Types of Asymptotic Notations →

① Big Theta (Θ) → Tight bound, complexity represented is like average value or range within which the actual time of execution will be.

② Big Oh (O) - This is used for upper bound of algorithm or worst Case of an algorithm. It tells that a function will never exceed specified time for any value of input n.

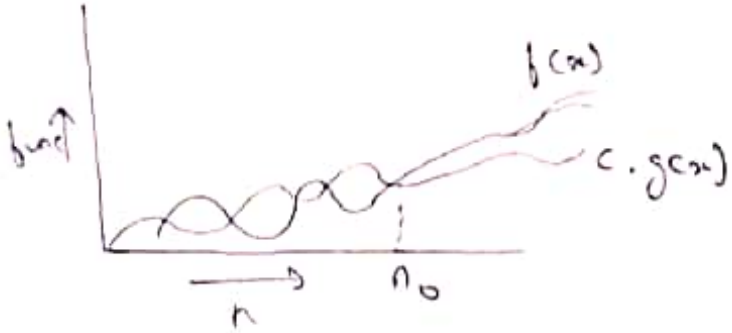
③ Big Omega (Ω) → Used to define lower bound of any algorithm or the best case of a algorithm.

1 Big OH (O) →



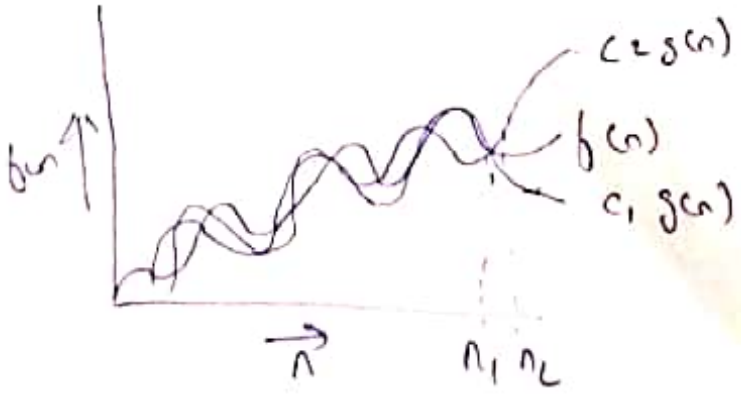
$f(n) = O(g(n))$ $g(n)$ is tight upper bound

2 Big omega (Ω)



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

3 Theta (θ) :- Gives both upper and lower bound



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

Q2 What should be complexity of for (i=1 to n) {i=i*2}

$\sum_{i=1}^{n}$ step ++

1, 2, 4, 8, ... n (k terms)

$\Rightarrow 2^0, 2^1, 2^2, 2^3 \ldots\ldots 2^{(k-1)} = n$

taking log

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

$(k-1) = \log n$

$k = \log n + 1$

$= O(\log n)$

Q3 $T(N) = \{ 3T(n-1)$ if $n>0$ otherwise $1 \}$

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

$T(n) = 3T(n-1) \quad \forall \quad n>0 \quad -(1)$

put $n = (n-1)$

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

putting (2) in (1)st

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

$= 3^2 T(n-2) \quad -(3)$

putting $n = n-2$ in eq (1)

$$T(n-2) = 3T(n-3) \quad -(4)$$

$$T(n) = 3^3 T(n-3) \quad -(5)$$

$$T(n) = 3^k T(n-k) \quad -(6)$$

Base case $\Rightarrow T(0) = 1$

$$n-k = 0$$
$$n = k$$

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

Q4 $T(n) = \{ 2T(n-1)-1 \text{ if } n>0 \text{ otherwise } 1 \}$

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

$$T(n) = 2T(n-1)-1$$
$$= 2(2T(n-2)-1)-1$$
$$\Rightarrow 2^2(T(n-2))-2-1$$
$$\Rightarrow 2^2(2T(n-3)-1)-2-1$$
$$\Rightarrow 2^3 T(n-3) - 2^2 - 2^1 - 2^0$$
$$\Rightarrow 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} \dots - 2^2 - 2^1 - 2^0$$
$$\Rightarrow 2^n - (2^n - 1)$$
$$\Rightarrow 2^n - 2^n + 1 = 1$$

$T(n) = 1$

Q5 What should be time complexity - int i=1 - - - - printf("#");

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

1, 3, 6, 10 - - - n terms

$s = 1+3+6+10+ \cdots k$

$0 = 1+2+3+4 \cdots k$

$k\left(\frac{k-1}{2}\right)$

$n \approx k^2$

$k = \sqrt{n}$

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

Q6 Time complexity of

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

$1, (2)^2, (3)^2, (4)^2 \cdots n$

$1, 2, 3, 4 \cdots \sqrt{n}$

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

Q7 Time Complexity of -

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

$$\sum_{i=n/2}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 1$$

$$\sum_{i=n/2}^{n} \sum_{\substack{j=1 \\ \text{step } j*2}}^{n} \log(n)$$

$$\sum_{i=n/2}^{n} (\log n)^2$$

$$\Rightarrow \left(\frac{n}{2} + 1\right)(\log n)^2 \Rightarrow T(n) = O(n(\log n)^2)$$

Q8 Time complexity - - - - $(n-3)$.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 \Rightarrow \sum_{i=1}^{n} n = n^2$$

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

putting $n = n-3$ in eq (1)

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

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

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

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

$T(1) = 0$

$n - 3k = 1$

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

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

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

Q9 Time complexity -- ---- } }

$$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 \quad \text{Step} = i$$

$$\sum_{i=1}^{n} \left(\frac{n-1}{i} + 1\right) \Rightarrow (n-1) \sum_{i=1}^{n} \frac{1}{i} + \sum_{i=1}^{n} 1$$

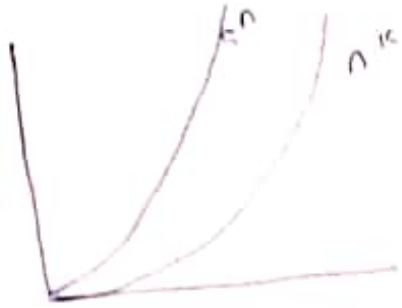
$$\Rightarrow (n-1) \sum_{i=1}^{n} \frac{1}{i} + n$$

$$(n-1) \log n + n$$

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

Q10 For function n^k and c^n what is asymptotic relationship between them. Find out value of c and n_0 for which relationship holds.

Ans 10



$n^k = o(a^n)$

$n^k \leq a^n \cdot c \quad \forall\ c>0$ and $n \geq n_0$

Let $n = n_0$

$n_0^k \leq c \cdot a^{n_0}$

~~Let $n = n_0$~~

$n_0^3 \leq c \cdot 3^{n_0}$ $\quad k = c = 3$ (say)

$\Rightarrow c \geq 1$ & $n_0 \geq 1$

Q11 What is time complexity -

```
void fun(int n) {
int j=1, i=0;
 while (i<n) {
  i = i+j;
  j++; } }
```

j	i
1	0
2	1
3	3
4	6
5	10
6	15
7	21

$S = 0, 1, 3, 6, 10, 15 \cdots$ (1)

$S = 0, 1, 3, 6, 10 \cdots$ (2)

$0 = 0, 1, 2, 3, 4, 5 \cdots k-n$

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

$n \simeq k^2$

$k = \sqrt{n}$

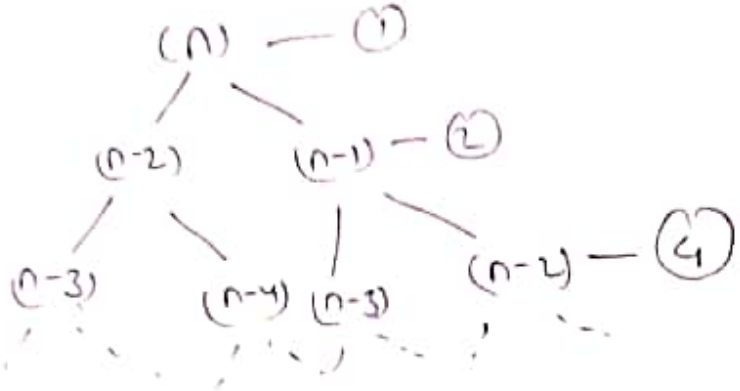
$T(n) = O(\sqrt{n})$

(9)

Q12 write recurrence relation for recursive function that prints fibonacci seq. Find time and space complexity?

0, 1, 1, 2, 3 - - - - - - - - - - n

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



$T = 1 + 2 + 4 + 8 - - - - - - - 2^n$

$a = 1$

$r = 2$

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

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

$T(n) = O(2^n)$

Space complexity = $O(n)$

because max stack frame is same as the longest node.

Q13 write time with complexity - $n(\log n)$, n^{13}, $\log(\log n)$?

(i)

```
for (int i=0; i<=n; i++)
{
    for (int j = 1; j<=n; j*=2)
        print ("*");
```

(1a)

(i) int a=1, b=1;
while (a<=n) {
a += b;
b += 1;
}

$O(\log \log n)$

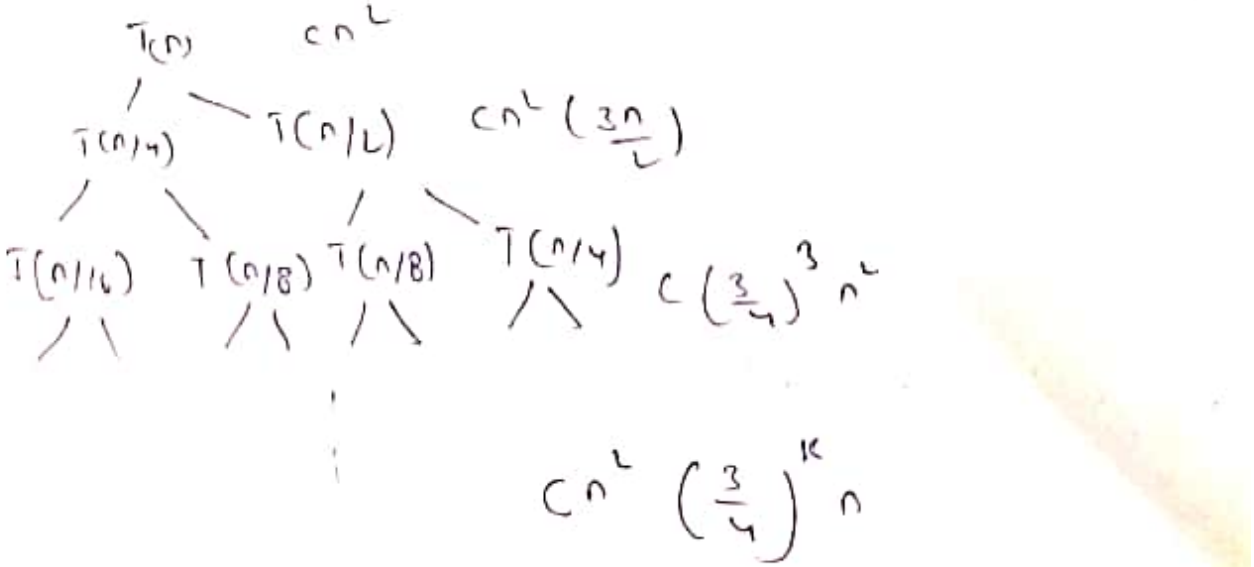
(ii) int func(int n)
{
if (n<=2)
return 1;
else
return (func(floor(sqrt(n)) + n);
}

$\log(\log n)$

(iii) for(int i=0; i<n; i++)
for(int j=0; j<n; j++)
for(int k=0; k<n; k++)
print("*");

n^3

Q1 4 Solve following recurrence relation $T(n) = T(n/4) + T(n/2) + cn^2$



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

$n = 2^k$

$k = \log n$

$$T(n) = cn^2 \left[1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2 + \cdots \left(\frac{3}{4}\right)^{\log n} \right]$$

$$cn^2 (1)$$
$$= n^2$$
$$T(n) = O(n^2)$$

Q15
```
int fun(int n) {
    for(int i=1; i<=n; i++)
    { for(int j=1; j<n; j+=i)
```

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n-1} (1)$$

$$\Rightarrow \sum_{i=1}^{n} \frac{n-1}{i}$$

$$\Rightarrow (n-1) \sum_{i=1}^{n} \frac{1}{i} \Rightarrow (n-1)\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots n\right)$$

$$= (n-1) \log n$$
$$T(n) = \Theta(n \log n)$$

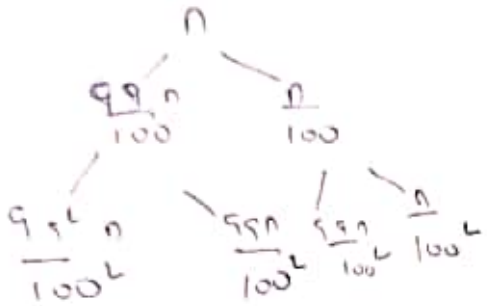
Q16
```
for(int i=2; i<=n; i = pow(i,k))
    { // O(1)
```

$i = 2, 2^k, 2^{k^2}, 2^{k^3}, \cdots 2^{k^k}$

$$\log(n) = k^x \log 2$$
$$\log_k(\log(n)) = k \log k$$
$$n = \Theta(\log(\log(n)))$$

Q17 Write a recurrence ---- ---- analysis?

$$T(n) = T\left(\frac{99}{100}n\right) + T\left(\frac{1}{100}\right)$$



∴ longest branch is $\frac{99}{100}n$

$$T(n) \Rightarrow \log_{\frac{100}{99}} n \approx \log n$$

$$\left(\frac{99}{100}\right)^k n = 1$$

$$n = \left(\frac{100}{99}\right)^k \qquad k = \log_{\frac{100}{99}} n$$

$$k = \log_{\frac{100}{99}} n$$

$$T(n) = n\left(\frac{\log_{\frac{100}{99}} n}{100}\right)$$

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

Q18 Increasing order of growth →

(a) $1 < \log\log n < \log n < \sqrt{n} < n < n\log n < n^2 < 2^n < 2^{2n} < 4^n < n!$

(b) $1 < \log^*(n) < \sqrt{\log(n)} < \log(n) < 2n < 4n < 2(2^n) < \log(n!) < 2\log(n) < n < n\log n = \log(n!) < N!$

(c) $96 < \log 8^n < n\log_6 6 = n\log_2 n < 5n < 8n^2 < 7n^3 < 8^{2n}$

Q19 Write linear search pseudocode to search an element in a sorted array with minimum comparision.

```
for (i=0 to k-1)
 { if (arr[i] = key)
    { return i;
    }
 }
return -1;
```

Q20 Write - - - - - - - - Lecture 8?

Iterative Insertion Sort

```
insertion sort (arr, n)
 loop from i=1 to i=n-1
 Pick elent arr[i] and insert it into sorted sequence
  of arr[0...i-1]
```

Recursive Insertion Sort

```
insertion sort (arr, n)
 { if n <=1
    return
   recursively sort n-1 elent
    insertion sort (arr, n-1)
   Pick last element arr[i] and insert
   it into sorted arr[0...i-1]
    }
```

Insertion sort considers one input element per iteration and produces a partial solution without considering future elements. It is also called online sorting algorithm

Q 21 Complexity of all sorting algorithm?

	Algorithm	Best case	Average Case	Worst case
1	Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
2	Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
3	Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
4	Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
5	Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
6	Heap Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Q 22 Divide all sorting algorithms inplace / stable / online sorting.

Algorithm	Inplace	Stable	Online Sorting
Bubble Sort	✓	✓	✗
Selection Sort	✓	✗	✗
Insertion Sort	✓	✓	✓
Merge Sort	✗	✓	✗
Quick Sort	✗	✗	✗
Heap Sort	✓	✗	✗

Q 23 Write recursive . . . . . Binary Search.

```
int binarysearch ( int ar[], int l , int n)
  {
   while (l <= r)
   {
    int m = (l+r) / 2;
    if(ar [m] = x)
       return m;
     else if (ar [m] < x)
       l = m+1;
     else
       r = m -1;
    }
     return -1;
  }
```

Recursive Binary Search

```
int Binary Search (int ar [], int l, int r, int x)
 { if (l > r)
     return -1
   int m = (l+r)/2 ;
   if (arr [m]=n)
         return m;
    else if (ar [m]<n)
    return Binary Search (ar, m+1, r, x );
     else
     return Binary Search(ar, l, m-1, x)
    }
```

Iterative Binary Search

Time complexity => Best = O(1) Avg = O(log n), worst = O(log n)

Space => O(1)

Recursive binary =>

Time complexity => Best = O(1) Averge = O(log n) worst = O(log n)

Space complexity => Best = O(1), Averge = O(log n) worst = O(log n)

Q 24 → $T(n) = T(n/2)+1 = T(n) = O(\log n)$