Name :- Abhay Rawat

Course :- B. Tech. (CSE) Sec :- B

Sem :- 5 Roll No :- 01

University Roll No :- 1961002

Design & Analysis of Algorithms

Assignment - 1

Ques What do you understand by Asymptotic notations. Define different Asymptotic notations with example.

Ans **Asymptotic notations**

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.

There are mainly three asymptotic notations:-

I Big-O notations

II Omega (Ω) notation

III Theta (Θ) notation

I Big-O notation

⊖ Big-O notation represents the upper bound of the running time of an algorithm.

$O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0 \}$

where $c > 0$ and $n \geq n_0$

⊖ This notation is known as the upper bound of the algorithm, or a worst case of an algorithm.

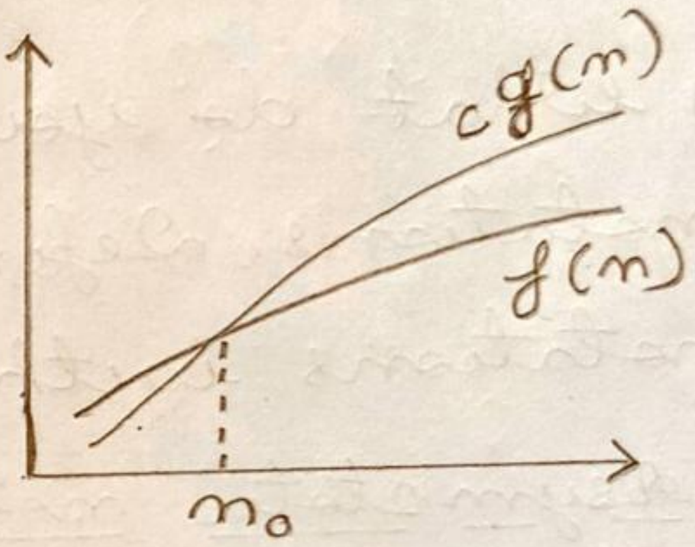
Example

$f(n) = 3 \log n + 100$

$g(n) = \log n$

$3 \log n + 100 <= c * \log(n)$

$c = 150$ & $n > 2$ (undefined at $n = 1$)



II Omega (Ω) Notation

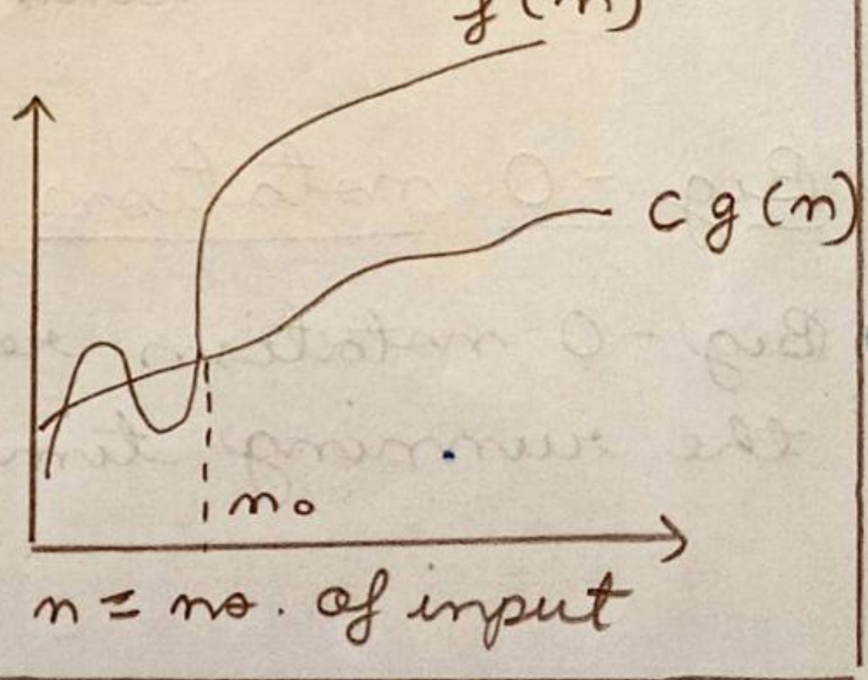
⊖ Omega notation represents the lower bound of the running time of an algorithm.

$\Omega(g(n)) = \{ f(n)$: there exist positive constant c and n_0 such that.

$0 \leq c\, g(n) \leq f(n)$

for all n, $n \geq n_0$.

⊖ this notation is known as the lower bound of the algorithm, or a best case of an algorithm.



Example

$f(n) = 3n + 2$

$c.g(n) \leq f(n)$ $\quad c = \text{constant} \quad g(n) = n$

$cn \leq 3n + 2$

$cn - 3n \leq 2$

$n(c-3) \leq 2 \Rightarrow n \leq \frac{2}{c-3}$

if we assume $c = 4$, then $n_0 = 2$.

$\boxed{c = 4,\ n_0 = 2}$

III Theta (Θ) Notation.

- Theta notation encloses the function from above & below. Since it represent the upper & the lower bound of the running time of an algorithm.
- $\Theta(g(n)) = \{ f(n)$: there exist positive constant $c1, c2$ and n_0 such that

$$0 \leq c1 * g(n) \leq f(n) \leq c2 * g(n)$$

for all $n > n_0$.

- This is known as tight Bounds of an algorithm or a average case of algorithm.
- Example

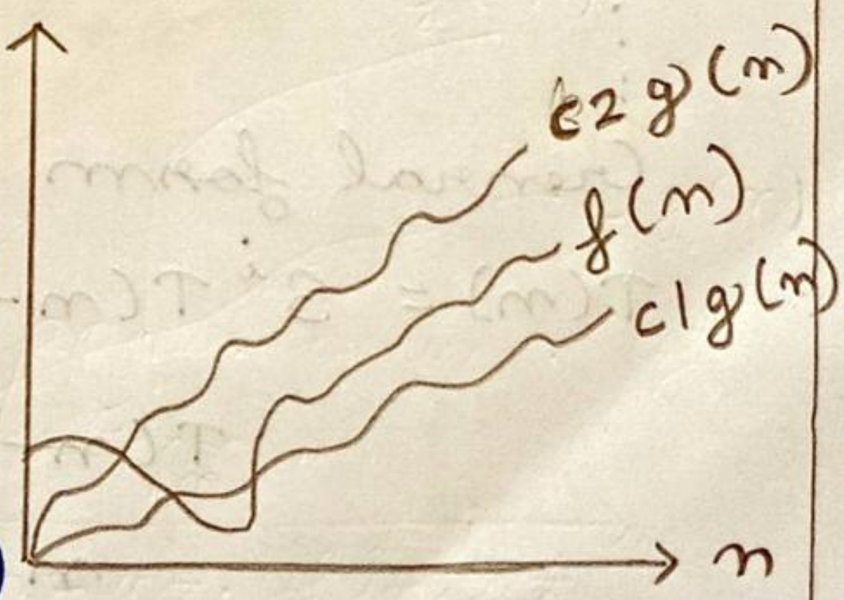
$f(n) = 5n^3 + 16n^2 + 3n + 8$

$5n^3 \leq 5n^3 + 16n^2 + 3n + 8 \leq (5+16+3+8)n^3$

$5n^3 \leq f(n) \leq 32n^3$

$\boxed{c_1 = 5,\ c_2 = 32,\ n_0 = 1}$

$f(n) \leftrightarrow \Theta(n^3)$



Ques 2 What should be time complexity of

for (i = 1 to n) { i = i * 2 }

Ans $i = 1, 2, 4, 8, 16, \ldots$ kth term $\ldots n$

$$G_n = a r^{n-1}$$

$$G_n = 1 (2)^{k-1}$$

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

$$\log_2 n = (k-1) \log_2 2$$

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

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

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

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

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

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

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

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

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

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

$T(n) = 3^3 \times 3 T(n-4)$

$T(n) = 3^4 \times T(n-4)$

⋮

General form

$T(n) = 3^i T(n-i)$ — I $\qquad T(0) = 1$

$T(n-i) = T(0)$

$n - i = 0 \Rightarrow \boxed{n = i}$

Put $n = i$ in Eq I

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

$T(n) = 3^n T(0)$ $\quad (T(0) = 1$ Given$)$

$T(n) = 3^n$

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

Ques4 $T(n) = \begin{cases} 2T(n-1) - 1 & \text{if } n > 0, \\ 1 & \text{otherwise} \end{cases}$

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

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

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

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

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

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

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

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

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

~~$T(n) = 2^4 T$~~

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

$\vdots$

General form.

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

$T(n-i) = T(0)$

$n - i = 0$

$\boxed{n = i}$

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

$$T(0) = 1$$

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

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

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

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

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

$$\boxed{T(n) = O(2^n)}$$

Ques 5 & 11 What should be time complexity of :-

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

Ans

No. of step (k)	s	i
0	0	1
1	1	2
2	3	3
3	6	4
4	10	5
5	15	6
6	21	7
⋮	⋮	⋮
k step	n	

$T(n) = O(k)$

$i = 0, 1, 3, 6, 10, \dots\dots, n$.

$S_n = 1 + 3 + 6 + 10 + 15 + \ldots\ldots + n$

$S_n = 1 + 3 + 6 + 10 + \ldots\ldots + (n-1) + n$

$(-) \quad (-) \quad (-) \quad (-) \quad . \quad . \quad . \quad . \quad . \quad . \quad .$

$0 = 1 + 2 + 3 + 4 + 5 + \ldots\ldots - n$

$n = 1 + 2 + 3 + 4 \ldots\ldots$ k step

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

$2n = k[2 + k - 1]$

$2n = k^2 + k \Rightarrow 2n = (k + 1/2)^2 - (1/2)^2$

$2n + (1/2)^2 = (k + 1/2)^2$

$k + 1/2 = \sqrt{2n + (1/2)^2}$

$k = \sqrt{2n + (1/2)^2} - 1/2$

$T(k) = T(k)$

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

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

Ques 7 Time complexity of

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

Ans $O(n \log n \log n)$

$O(n (\log n)^2)$

Ques8 Time complexity of

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

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

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

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

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

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

⋮

General Term

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

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

$n = i+1 \Rightarrow \boxed{n-1 = i}$

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

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

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

$$T(n) = \frac{n(n+1)(2n+1)}{6}$$

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

Ques Time complexity of -

```
void function (int n)
{
  for(i=0; to n){
    for(j=0; j<=n; j+=i)
      print("#").
  }
}
```

Ans $O(n\sqrt{n})$

Ques For the function n^k and a^n, what is the asymptotic relation b/w these function? Assume that $k>=1$ & $a>1$ are constant. Find out the value of c & n_0 for what relation hold

Ans If $c>1$, then the exponential c^n for outgrows any term, so that ans is

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

Ques 12 Write recurrence relation for the recursive function that prints Fibonacci series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity of this program & why?

Ans

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

$$T(n-2) \approx T(n-1)$$

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

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

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

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

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

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

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

$$\vdots$$

General Term

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

$$n - i = 0$$

$$\boxed{n = i}$$

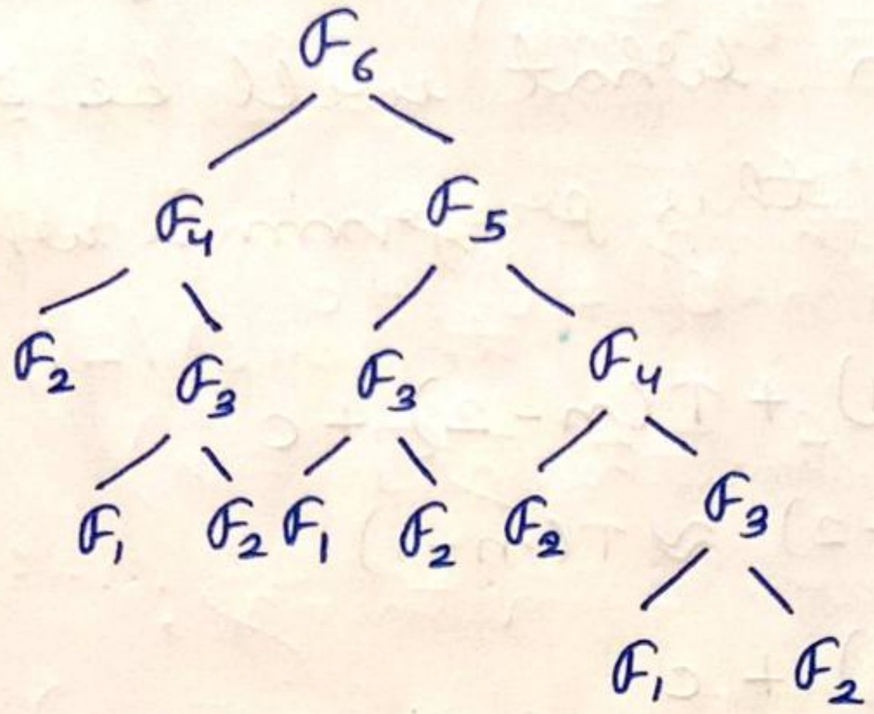
$$T(n) = 2^n T(0) + (2^0 + 2^1 + 2^2 + \dots + 2^{n-1})c$$

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

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

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

fib(6)



The maximum depth is proportional to the N, hence the space complexity of Fibonacci recursive is O(n).

Ques 13 Write programs which have complexity

① n log n.

Ans
```
void fun( )
{
   int i, j;
   for(i=1; i<=n; i++)
   {
     for(j=0; j<=n; j=j*2)
        printf("#");
   printf("\n");
   }
}
```

(ii) n^3

Ans

```
void fun(int n)
{
    int i, j, k;
    for(i=0; i<=n; i++)
    {
        for(j=0; j<=n; j++)
        {
            for(k=0; k<=n; k++)
                printf("#");
        }
    }
}
```

(iii) $\log(\log n)$

Ans

```
void SieveOfEratosthenes(int n)
{
    bool prime[n+1];
    memset(prime, true, sizeof(prime));
    for(int p=2; p*p<=n; p++)
    {
        if(prime[p]==true)
        {
            for(int i=p*p; i<=n; i+=p)
                prime[i]=false;
        }
    }
}
```

```
for ( int p=2 ; p <=n; p++)
    if ( prime[p])
        cout << p << endl;
```

3

Ques14 Solve the following requrrence relation

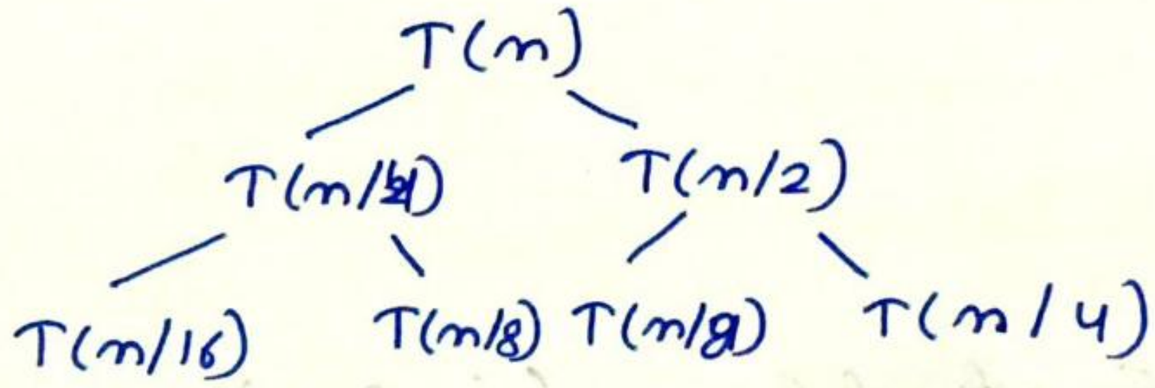
$$T(n) = T(n/4) + T(n/2) + cn^2$$

Ans $T(n) = T(n/4) + T(n/2) + cn^2 \qquad T(1) = c$

$n = n/2$

$T(n/2) = T(n/8) + T(n/4) + c(n^2/4)$

$T(n) = T(n/4) + 2T(n/16) + c(n^2/16 + n^2/4 + n^2)$



$$T(n) = c\left[n^2 + \frac{5n^2}{16} + \frac{25n^2}{256} + \cdots\right]$$

$$T(n) = n^2 c\left[1 + \frac{5}{16} + \frac{25}{16^2} + \cdots\right]$$

$$\boxed{T(n) = O(n^2)}$$

Ques15 What is the time complexity of following function fun()?

```
int fun(int n)
{
  for(int i=1; i<=n; i++)  ---- n
  {
    for(int j=1; j<n; j+=i)
    {
      //Some O(1) task
    }
  }
}
```

Ans For i = 1, the inner loop is executed n times
For i = 2, the inner loop is executed n/2 times
For i = 3, the inner loop is executed n/3 times
For i = 4, the inner loop is executed n/4 times
⋮
For i = n, the inner loop is executed n/n times.

Total time = $n + n/2 + n/3 + \ldots + n/n$

$= n(1 + 1/2 + 1/3 + \ldots + 1/n)$

$= n \log n$

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

Ques16 What should be the time complexity of

```
for(int i=2; i<=n; i=pow(i,k))
{
```

//some O(1) expressions or statements

}

where, k is a constant

Ans O(log(log n))

Ques 18 Arrange the following in increasing order of rate of growth.

(a) n, $n!$, $\log n$, $\log \log n$, root(n), $\log(n!)$, $n \log n$, 2^n, 2^{2n}, 4^n, n^2, 100

Ans 100, $\log \log n$, $\log n$, root n, n, $n \log n$, n^2, 2^n, 2^{2n}, 4^n, $n!$

(b) $2(2^n)$, $4n$, $2n$, 1, $\log(n)$, $\log(\log(n))$, $\sqrt{\log(n)}$, $\log 2n$, $2\log(n)$, n, $\log(n!)$, $n!$, n^2, $n \log(n)$

Ans 1, $\log(\log(n))$, $\sqrt{\log n}$, $\log n$, $\log(2n)$, $\log(n!)$, $2\log(n)$, n, $2n$, $4n$, $n \log(n)$, n^2, $2(2^n)$, $n!$

(c) 8^{2n}, $\log_2(n)$, $n \log_6(n)$, $n \log_2(n)$, $\log(n!)$, $n!$, $\log_8(n)$, 96, $8n^2$, $7n^3$, $5n$.

Ans 96, $\log_8 n$, $\log_2 n$, $\log(n!)$, $5n$, $n \log_6 n$, $n \log_2 n$ $8n^2$, $7n^3$, 8^{2n}, $n!$

Ques 19 Write linear search pseudocode to search an element in a sorted array with minimum comparision.

Ans
```
LINEAR_SEARCH(A, key)
    comp ← 0, f ← 0
    for i = 1 to A.length
        comp ← comp + 1
        if A[i] == key
            print "Element founded"
```

```
        f = 1
    if f == 0
        print "Comperision not Element not found"
print comp.
```

Ques 2° Write pseuds code for iteration & recursive insertion sort. Insertion sorting is called online sorting. Why? What about other sorting algorithms that has been discussed in lecture?

Ans Iterative Method of insertion sort.

```
INSERTION_SORT(A)
for j = 2 to A.length
    key = A[j]
    i = j - 1
    while i > 0 and A[i] > key
        A[i+1] = A[i]
        i = i - 1
    A[i+1] = key
```

Recursive Method of insertion sort

```
INSERTION_SORT(A, n)
if n ≤ 1
    return
INSERTION_SORT(A, n-1)
key = A[n-1];
j = n-2;
while j ≥ 0 and A[j] > key
```

$$A[j+1] = A[j]$$
$$j = j-1$$
$$A[j+1] = key)$$

Insertion sort considers one input element per iteration & produces a partial solution without considering future elements. that why it is called online sorting).

Other sorting algorithm that have been discussed in lectures are:-

- Bubble sorting
- Selection sorting
- Quick sort
- Merge sort
- Heap sort
- counting sort

Ques 2 | Complexity of all sorting algorithms that has been discussed in lectures.

Ans

	Best case	Average case	Worst case
Bubble Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$
Selection Sort	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$
Insertion Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$
Merge Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$
Heap Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$
Quick Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N^2)$
counting sort	$\Omega(N+K)$	$\Theta(N+K)$	$O(N+K)$

Ques 22 Divide all the sorting algorithms into in place/stable/online sorting.

Ans

	In Place	Stable	Online
Bubble Sort	Yes	Yes	Yes
Insertion Sort	Yes	Yes	Yes
Selection Sort	Yes	No	Yes
Merge Sort	No	Yes	Yes
Quick sort	Yes	No	Yes
Heap Sort	Yes	No	Yes
Count Sort	No	Yes	Yes

Ques 23 Write recursive/iterative pseda code for binary search. What is the Time & Space complexity of Linear & Binary (Recursive & Interative)

Ans Linear Search

```
LINEAR_SEARCH (A, key)
    found ← 0
    for i = 1 to N
    if A[i] == key
        found ← 1
        print "Element found"
        break

if found == 0
    print "Element Not found."
```

Time complexity - O(n)

Space complexity - O(1)

Binary Search (~~Linear~~ Iterative)

```
BINARY-SEARCH (A, beg, end, key)
while beg ≤ end
        mid = beg + (end - beg)/2
        if mid == key
                return mid
        if ~~mid~~ A[mid] < key
                beg = mid + 1
        if A[mid] > key
                end = mid - 1
return -1
```

Time complexity - $O(\log_2 n)$

Space complexity - O(1)

Binary Search (Recursion)

```
BINARY_SEARCH(A, beg, end, key)
    if end ≥ beg
            mid = (beg + end)/2
    if A[mid] == item
            return mid + 1
    else if A[mid] < item
            return BINARY_SEARCH(A, mid+1,
                                 end, key)
```

```
else
    return BINARY_SEARCH(A, beg, mid-1,
                                end)

return -1
```

Time complexity - O(log n)

Space complexity - O(1)

Ques 24 Write recurrence relation for binary recursive search.

Ans $T(n) = T(n/2) + C$