TUTORIAL : 3

1.
```
int LinearSearch (int *arr, int n, int key)
    for (i<=0 to n-1)
        if (arr[i] = key)
            return i
    return -1;
```

2. insertion sort:

iterative:

```
for J=2 to n,
    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
```

3. Complexity for sort algorithms are:

1. Bubble sort
   Best = O(n²)
   Worst = O(n²)
   Avg = O(n²)

2. Selection sort
   Best = O(n²)
   Worst = O(n²)
   Avg = O(n²)

Insertion sort
Best = O(n)
Worst = O(n²)
Avg = O(n²)

merge sort
Best = O(nlogn)
Worst = O(nlogn)
Avg = O(nlogn)

Quick sort

Best = O(nlogn)

Worst = O(n²)

Avg = O(nlogn)

Counting sort

Best = O(n+k)

Worst = O(n+k)

Avg = O(n+k)

7 Heap sort

Best : O(nlogn)

Avg O(nlogn)

Wor O(nlogn)

4 Iterative pseudo code

Binary search

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

Time Complexity

Best Case → O(1)

Worst Case → O(logn)

Avg Cas → O(logn)

5. Inplace Sorting: Bubble, selection, insertion, heap, quick sort.
Online Sort : self ; insertion
stable sort :→ Bubble, insertion, merge, Counting

6 Quick sort is mostly used in practical used it is the fatest general purpose sort In most patical situation quick sort is the mthd of choice if stability is important and space is available merge sort might be best

7. $T(n) = T(n/2) + 1$ put $n = n/2$

$T(n/2) = T(n/4) + 1$

$T(n) = T(n/4) + 1 + 1$

$T(n/4) = T(n/8) + 1$

$T(n) = T(n/8) + 1 + 1 + 1$

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

$\frac{n}{2^k} = 1 \Rightarrow 2^k = n, \Rightarrow k = \log n$

$T(n) = T\left(\frac{n}{2^{\log_2 n}}\right) + \log n$

$T(n/n) + \log n$

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

11. Best Case of Quick sort.

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

$a = 2 \quad b = 2 \quad f(n) = n$

$= n^{\log_b a} \Rightarrow n^{\log_2 2} = n$

$f(n) = n$

$F(n) = a(n^{\log_b a} \log^k n)$

$F(n) = \Theta(n \log^k n) = \Theta(n \log n)$

Worst case of Quick sort

$T(n) = T(n-1) + n \qquad n = n-1,$

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

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

For $n = n-2$

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

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

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

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

$n = k+1,$

$k = n-1$

$T(n) = 1 + 2 + 3 + \cdots + n.$

$T(n) = n(n+1)/2 = O(n^2)$

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

Best case of merge sort:

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

$$a=2,\ b=2,\ f(n)=n.$$

$$n\log_b a = n\log_2 2 = n$$

$$F(n) = a(n\log_b a \cdot \log^k n)$$

$$F(n) = \Theta(n \log^k n)$$

$$F(n) = \Theta(n\log n).$$

Same as worst case:

Similarities is Best case Complexities of merge sort and quicle sort both are same

$F(n) = \Theta(n\log n)$ at worst case Complexity of both the sorting are difficult. merge sort $= \Theta(n\log n)$ & Quick sort $= O(n^2)$