TUTORIAL-3

Name - Nancy Jaiswal
Section - F
Roll No. - 64

1. Write linear search pseudocode to search an element in a sorted array with minimum comparisons.

```
for (i = 0 to n)
  {
    if (arr[i] == value)
      // element found at i
  }
```

2. Iterative:

```
void insertion_sort (int arr[], int n)
{
  for (int i = 1; i < n; i++)
  {
    j = i - 1;
    x = arr[i];
    while (j > -1 && arr[j] > x)
    {
      arr[j+1] = arr[j];
      j--;
    }
    arr[j+1] = x;
  }
}
```

Recursion:

```
void insertion_sort(int arr[], int n)
{
  if (n<=1)
    return;
  insertion_sort(arr, n-1);
  int last = arr[n-1];
  int j = n-2;
  while (j>=0 && arr[j]> last)
  {
    arr[j+1] = arr[j];
    j--;
  }
  arr[j+1] = last;
}
```

Insertion sort is called 'Online Sort' because it does not need to know anything about what values it will sort & information is requested while algorithm is running.

Other sorting algorithms:

- Bubble Sort
- Quick Sort
- Merge Sort
- Selection Sort
- Heap Sort

3.

Sorting Algorithm	Best	Worst	Average
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Heap Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quick Sort	$O(n \log n)$	$O(n^2)$	$O(n \log n)$
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

4.

Inplace Sorting	Stable Sorting	Online Sorting
Bubble Sort	Merge Sort	Insertion Sort
Selection Sort	Bubble Sort	
Insertion Sort	Insertion Sort	
Quick Sort	Count Sort	
Heap Sort		

5. Iterative Code:

```
int binary_search(int arr[], int l, int r, int key)
{
    while (l <= r) {
        int m = ((l+r)/2);
        if (arr[m] == key)
            return m;
```

```
        else if (key< arr[m])
            r = m-1;
        else
            l = m+1;
    }
    return -1;
}
```

Recursive Code:

```
int binary_search (int arr[], int l, int r, int key)
  {
    while (l<=r) {
    int m = (l+r)/2;
    if (key == arr[m])
      return m;
    else if (key< arr[m])
      return binary_search (arr, l, m-1, key);
    else
      return binary_search (arr, m+1, r, key);
    }
    return -1;
}
```

Time Complexity:

Linear Search - O(n)

Binary Search - O(log n)

6.

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

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

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

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

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

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

$\vdots$

$T(n/2^k) + 1$ (k times)

Let $q^k = n$

$k = \log n$

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

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

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

7.

```
for (i=0; i<n; i++)
  {
    for (int j=0; j<n; j++)
      {
        if (a[i] + a[j] == k)
          printf("%d %d", i, j);
      }
  }
```

8. Quick sort is fastest general-purpose sorting. In most practical situation quick sort is the method of choice as stability is important and space is available, merge sort might be best.

9. A pair $(A[i], A[j])$ is said to be inversion if
$A[i] > A[j]$
$i < j$
Total no. of inversions in given array are 31 using merge sort.

10. Worst Case $O(n^2)$: The worst case occurs when the pivot element is an extreme (smallest / largest) element. This happens when input array is sorted or reverse sorted and either first or last element is selected as pivot.

Best Case $O(n \log n)$: The best case occurs when we will select pivot element as a mean element.

11. Merge Sort:
Best Case - $T(n) = 2T(n/2) + cn$
Worst Case - $T(n) = 2T(n/2) + O(n)$ } $O(n \log n)$

Quick Sort:

Best Case - $T(n) = 2T(n/2) + O(n) \rightarrow O(n \log n)$

Worst Case - $T(n) = T(n-1) + O(n) \rightarrow O(n^2)$

In quick sort, array of element is divided into 2 parts repeatedly until it is not possible to divide it further.

In merge sort, the elements are split into 2 subarray $(n/2)$ again & again until only one element is left.

12.

```
for (int i=0 ; i< n-1 ; i++)
  {
    int min = i;
   for (int j = i+1 ; j<n ; j++)
    {
      if (a[min] > a[j])
        min = j;
    }
   int key = a[min];
   while (min > i)
    {
      a[min] = a[min-j];
      min--;
    }
   a[i] = key;
}
```

13.

A better version of bubble sort, known as a bubble sort, includes a flag that is set if a exchange is made after an entire pass over. If no exchange is made then it should be called the array is already order because no two elements need to be switched.

```
void bubble (int arr[], int n)
{
  for (int i=0; i<n; i++)
  {
    int swaps=0;
  for (int j=0; j<n-1; j++)
  {
    if (arr[j] > arr[j+1])
    {int t= arr[j];
     arr[j] = arr[j+1];
     arr[j+1] = t;
     swaps++;
    }
  }
  if (swaps == 0)
  break;
}
}
```