Tutorial-3

Name - Anshika Garg
Section - F
Roll No - 46

Ques-1 Write linear search Pseudo Code to search an element in a sorted array with minimum Comparisons-

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

Ques 2 Write Pseudo Code for Iterative & recursive Insertion sort. Insertion sort is called online sorting. why? what about other sorting algorithms that has been discussed?

Iteration-

```
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;
    }
}
```

Recursive -

```
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 and information is requested while algorithm is running

Other Sorting Algorithms -

1) Bubble Sort
2) Quick Sort
3) Merge Sort
4) selection Sort
5) Heap Sort.

Q.3 Complexity of all sorting algorithms that has been discovered in lectures.

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

Ques.4 Divide all sorting algorithm into inplace/ stable/ online sorting.

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		

Ans. 5

Iterative →

```
int b_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;
}
```

// Time complexity = O(n)

Recursive →

```
int b_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 b_search(arr, l, mid-1, key);
   else
        return b_search(arr, mid+1, r, key);
  }
  return -1;
}
```

// Time Complexity = O(logn)

Q.6 Write recurrence relation for binary recursive search.

$$T(n) = T(n/2) + 1 \rightarrow ①$$
$$T(n/2) = T(n/4) + 1 \rightarrow ②$$
$$T(n/4) = T(n/8) + 1 \rightarrow ③$$

$$T(n) = T(n/2) + 1$$
$$= T(n/4) + 2$$
$$= T(n/8) + 3$$
$$= T\left(\frac{n}{2^k}\right) + k$$

let $2^k = n$

$k = \log n$

$$T(n) = T(n/n) + \log n$$
$$T(n) = T(1) + \log n$$
$$T(n) = O(\log n) \rightarrow \text{Answer}$$

Ques.7 Find two indexes such that A[i] + A[j] = k in minimum time complexity.

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

Q-8 Which Sorting is Best for Practical Use? Explain.

Quick-Sort is fastest general-purpose sort. In most practical situations quicksort is the method of choice as stability is important & space is available, merge sort might be best.

Que.9 What do you mean by inversions in an array? Count the no. of inversions in Array arr[ ] = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5} using merge sort.

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.

Ques.10 In which case Quick sort will give best & worst case T.C.?

W.C. ($O(n^2)$) - 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.

B.C. ($O(n \log n)$) - The Best case occurs when we will select pivot element as a mean element.

11 Merge Sort →

Best Case − $T(n) = 2T(n/2) + O(n)$ ⎫ $O(n\log n)$
Worst Case − $T(n) = 2T(n/2) + O(n)$ ⎭

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 elements 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 1 element is left.

Q-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 - 1];
        min --;
    }
    a[i] = key;
}
```

Q-13 A better version of bubble sort, known as m bubble sort, includes a flag that is set of 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 2 elements need to be switched.

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