Tutorial - 3

Name - Shagun Gupta
Section - C
Roll No. - 11
University Roll No. - 2017012

Q1.
Ans

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

Q2.
Ans

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] > n)
                {
                     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 'outline sort' because it does not need to know anything about what values it will ~~sort~~ sort & information is requested while algorithm is running

→ Other sorting algorithm

- Bubble Sort
- Quick Sort
- Merge Sort
- selection Sort.
- Heap Sort.

Q3.

Ans.

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

Q4

Ans

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		

Q5

Ans. 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;
}
```

Recursive

```
    int b_search( int arr[], int l, int r, int key)
    {
        while( l <= r)
        {   int m = (( l+r)/2);
          if(key < arr[m])
              r = m-1;
          else if ( arr [m] == key)
              return m;
          else
        }
           return b_search(arr, mid+1, r, key);
    }                              // time complexity = O(n)
      return -1;
    }
```

Q6.

Ans

$$T(n) = T(n/2) + 1 \quad — ①$$
$$T(n/2) = T(n/4) + 1 \quad — ②$$
$$T(n/4) = T(n/8) + 1 \quad — ③$$
$$T(n) = T(n/2) + 1$$
$$= T(n/4) + 2$$
$$= T(n/8) + 3$$
$$= T(n/2^k) + k$$

let $2^k = n$
$k = \log n$
$$T(n) = T(n/n) + \log n$$
$$T(n) = T(1) + \log n$$
$$T(n) = O(\log n)$$

Q7

Ans

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

Q8

Ans 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

Q9

Ans 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.∴

Q10

Ans 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

Q11 Merge Sort

Ans Best case → $T(n) = 2T(n/2) + O(n)$

Worst case → $T(n) = 2T(n/2) + O(n)$

$\Big\}\ O(n\log n)$

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 elements is divided into 2 parts repeatedly until it is not possible to divide it further.

Date.
Page No.

in merge sort → the elements are split into 2 subarray (n/2) again & again until only 1 element is left

Q12
Ans

```
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;
}
```

Q13. A better version of bubble sort, known as m bubble sort, includes a flag that is set of an 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 g 2 elements need to be switch.

re

```
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;
   }
}
```