Tutorial - 3

Name → Janvi Sah
Section → G
Roll No. → 62

Ans 1.

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

Ans 2.

```
void insertionsort(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;
  }
}
```

Scanned with CamScanner

Recursive

```
void insertionsort(int arr[], int n)
{
  if (n<=1)
    return;
  insertionsort(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 sorting because it does not need to know anything about what values it will sort and information is requested while algorithm is running.

Other Sorting Algorithm

Bubble sort
Quick sort
Merge sort
Selection sort
Heap sort
Count sort

Scanned with CamScanner

Ans 3.

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

Ans 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		

Ans 5. Iterative :→

```
int linearsearch (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
```

Scanned with CamScanner

```
            l = m+1;
        }
        return -1;
    }
```

Recursive

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

Time complexity →

Linear search → O(n)

Binary search → O(log n)

Scanned with CamScanner

Ans 6.

$T(n) = T(n/2) + 1$ — (1)

$T(n/2) = T(n/4) + 1$ — (2)

$T(n/4) = T(n/8) + 1$ — (3)

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

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

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

$\vdots$

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

let $g^k = n$

$k = \log n$

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

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

$\boxed{T(n) = O(\log n)}$ → Answer

Ans 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);
      }
  }
```

Scanned with CamScanner

Ans 8.

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

Ans 9

A pair (A[i], A[j]) is said to be inversion if

$A[i] > A[j]$

$i < j$

Total no. of inversion in given array are 31 using merge sort.

Ans 10.

Worst case $O(n^2)$ → The worst case occurs when the pivot element is on extreme (smallest / largest) element.

This happen 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.

Scanned with CamScanner

Ans11

Merge Sort.

Best case → $T(n) = 2T(n/2) + O(n)$

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 and again until only one element is left.

Ans13

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.

Scanned with CamScanner