Tutorial No - 3

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 from d.
}
```

Ques 2. Write a Pseudo code for iteration and recursive insertion sort. Insertion sort is called online sorting. why? what about other sorting algorithms that has been discussed?

Ans 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;
    }
}
```

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 Algorithm :

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

Ques 3 Complexity of all sorting algorithm that has been discussed in lectures.

Ans

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

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

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		

Ques Write recursive/iterative Pseudo code for binary search. What is the Time and Space complexity of linear and Binary search.

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])
        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 -

a). Linear search - $O(n)$

b) Binary search - $O(\log n)$

Ques - 6. Write recursive relation for binary recursive search.

$T(n) = T(n/2) + 1$ —— 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 $2^k = n$.

$K = \log n$

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

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

$T(n) = O \log n$ — Ans

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);
      }
  }
```

Ques 8 Which sorting is best for practical uses? Explain

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

Ques 9 What do you mean by inversions in an array? Count the number of inversions in Array arr[] = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5} using merge sort.

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 ~~of~~ 31 using merge ~~so~~ sort.

Ques 10. In which case quick sort will give best and worst case time complexity.

Ans 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

Ques 11. Write Recurrence Relation of Merge/Quick sort in best & worst case. What are the similarities and difference between complexities of two algorithm and why?

Ans 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 untill it is not possible to divide it further.

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

Ques 12. Selection sort is not stable by default but can you write a version of stable selection sort?

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;
}
```

Ques 13. Bubble sort scans array even when array is sorted. Can you, modify, the bubble sort so that it does not scan the whole array once it is sorted.

Ans

A better version of bubble sort, known as m bubble sort includes a flag that is set of exchange is made after an entire pass over. If no exchange is made then it should be called the array 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-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;
   }
}
```