Tutorial - 03

Ques.1. Write linear search pseudo code to search an element in a sorted array with minimum comparisons.

```
int linearsearch (a, n, key)
{
    if (abs(a[0] - key) > abs(a[n-1] - key))
        for (i = n-1 to 0; i--)
            if (a[i] == key)
                return i;
    else
        for (i = 0 to n-1; i++)
            if (a[i] == key)
                return i;
```

Ques 2. Pseudo code for iterative and recursive insertion sort. insertion sort called online sorting. why? what about other sorting algo.

→
```
insertion sort (int a[], int n)   // iterative
{
    for (i = 1 to n; i++) {
        x = a[i];
        j = i - 1;
        while (j > -1 && a[j] > x)
        {
            a[j+1] = a[j];
            j--;
        }
        a[j+1] = x;
    }
}
```

→
```
insertion sort (int a[], int n)   // Recursive
{   if (n <= 1)
        return;
    insertionsort (a, n-1);
    int x = a[n-1];
    j = n-2;
    while (j >= 0 && a[j] > x){
        a[j+1] = a[j];
        j--;
    }
    a[j+1] = x;
}
```

Insertion sort is called online sorting because it contains only one input per iteration & produces a partial solution without considering future elements whereas other sorting algorithms process the the whole problem data together from the beginning & is required to output an answer which solve the problem at hand.

Ques. complexity of all sorting algorithms

sorting	Best	worst	Average
i) Bubble sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
ii) selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
iii) insertion sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
iv) Quick sort	$O(n\log n)$	$O(n^2)$	$O(n\log n)$
v) Merge sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$
vi) count sort	$O(n+m)$	$O(n+m)$	$O(n+m)$ {m → range}
vii) Heap sort	$O(n\log n)$	$O(\log n)$	$O(n\log n)$

Ques. 4.

sorting Technique	Inplace	stable	online
i) Bubble sort	✓	✓	×
ii) selection sort	✓	×	×
iii) insertion sort	✓	✓	✓
iv) Quick sort	✓	×	×
v) Merge sort	×	✓	×
vi) count sort	×	✓	×
vii) Heap sort	✓	×	×

Ques. 5.

Recursive / Iterative Pseudo code for Binary search. Time and space complexity of linear & Binary search.

```
int Rbinary_search(a,l,r,x)          // Recursive
{
  while(l<=r){
    mid=(l+r)/2;
    if(x>a[mid])
      return Rbinary_search(a,mid+1,r,x);
    else if(x<a[mid])
      return Rbinary_search(a,l,mid-1,r);
    else
      return mid;
  }
}
```

```
→ int binary_search(a,n,x)           // Iterative
{
    l=0, r=n-1;
    while(l<=r){
      mid=(l+r)/2;
      if(x<a[mid])
        r=mid-1;
      else if(x>a[mid])
          l=mid+1;
      else
        return mid;
    }
```

	Time complexity	space complexity
1) Linear search	O(n)	O(1)
2) Binary search	O(log n)	O(1)

Ques.6.
Recurrence Relation for binary recursive search

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

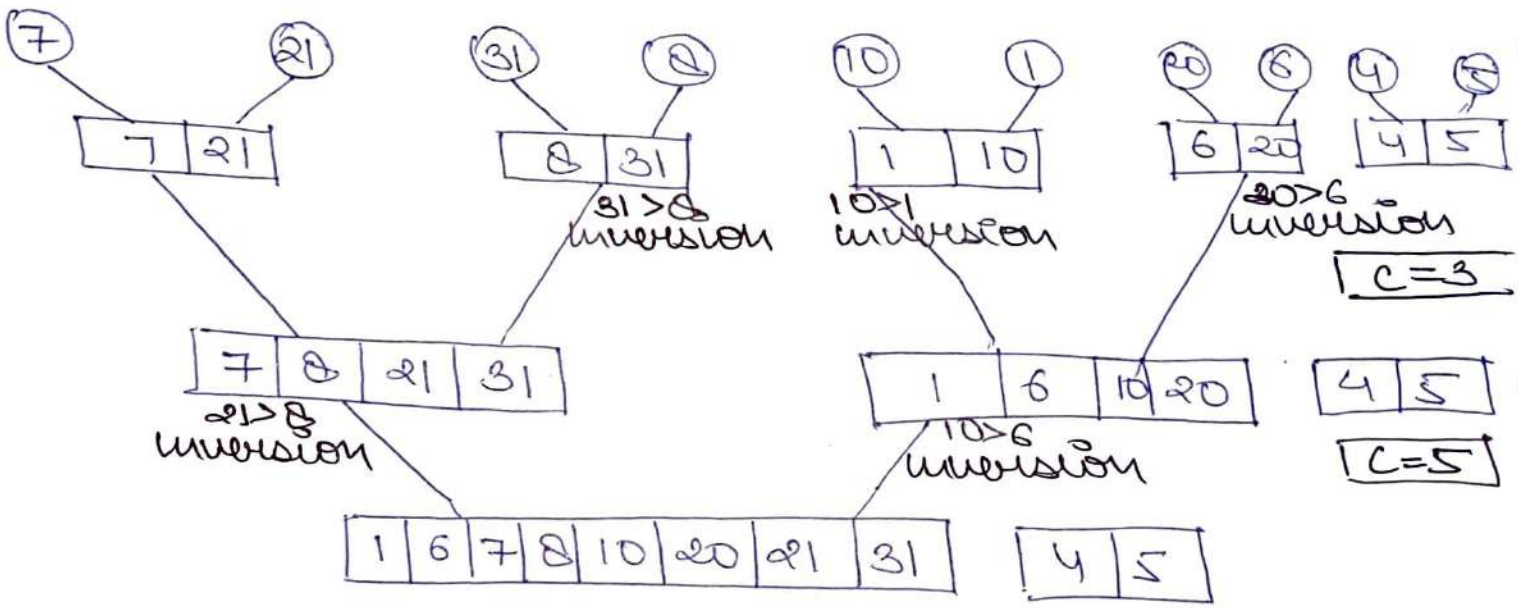
Ques.7. find 2 indexes such that A[i]+A[j]=k in minimum time complexity

```
find Index(int a[], int n, int k){
    i=0, j=1;
    while(i<n && j<n){
      if(i!=j && a[j]-a[i]==k || a[i]-a[j]==k))
        printf("%d, %d", i, j);
      else if(a[j]-a[i]<k)
          j++;
        else  i++;
    }
}
```

Ques 8. which sorting is best for practical uses?
Quick sort is one of the most efficient sorting algorithms which makes it one of the most used as well, it is faster as compared to other sorting algorithms. Also, its time complexity is $O(n \log n)$ but in case of a larger array Merge sort is preferred.

Ques.9. what do you mean by no. of 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.

Ans Inversions is an array basically define how far or close an array is from being sorted. If array is already sorted, inversion count → 0; if array is in reverse order, inversion count → maximum.



7>1, 7>6, 8>1, 8>6, 2>10, 21>20, 31>1, 31>6, 31>10, 31>20, 21>1, 21>6

C = 17

1	4	5	6	7	8	10	20	21	31

6>4, 6>5, 7>4, 8>4, 8>5, 10>4, 10>5, 20>4, 20>5, 21>4, 21>5, 31>4, 31>1

count = 14
= 14 + 17

Total count = 31

Ques.10. In which case Quick sort will give best & worst case time complexity?

BEST CASE – If partitioning element is in the middle.

Time complexity = O(n log n)

WORST CASE – If pivot is at extreme position & array is already sorted in increasing / decreasing order.

Time complexity = O(n²)

Ques.11. ~~which~~ write recurrence relation of ~~any~~ merge & quick sort in best & worst case? similarities & difference b/w complexity of 2 algos & why?

Quick sort – Best : T(n) = 2T(n/2) + n

worst : T(n) = T(n-1) + n

Merge sort –

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

In merge sort, the array is divided into 2 equal halves in times.

∴ T.C. = O(n log n)

In Quick sort, the array is divided into any ratio depending on the position of pivot element.

∴ Time complexity varies from O(n²) to O(n log n)

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

In selection sort, normally we swap the minimum value with the first value, which makes it unstable. To make it stable, instead of swapping, insert the ~~last~~ value at pos-0 to n.

Ques.13. Bubble sort scans whole array when array is sorted. can you modify the bubble sort so that it doesn't scan whole array.

```
void bubble sort (int a[], int n)
{
    for (i=0 to n) {
        swaps=0;
        for (j=0 to n-1-i)
            if (a[j] > a[j+1])
                swap (a[j], a[j+1]);
                swap++;
        if (swaps==0)
            breaks;
    }
}
```

Ques.14. your computer has RAM of 2 GB; given array of 4 GB for sorting which algorithm you would use? External & internal sorting?

In such cases, external sorting algorithms such as k-way. Merge sort is used that can handle large data amounts which can't fit into main memory. A part of array resides in RAM during the execution whereas in internal sorting, process takes place entirely within the main memory; mostly used when data to be sorted is small

eg :- Bubble sort, quick sort, etc.