Assignment-3

Name - Vibha Rani
Section - CE [Computer engineering]
Roll no - 45
University Roll no - 2017395

Tutorial-3

Ques1 Write a linear search pseudo code to search a element in a sorted array with minimum comparisons.

```
int LinearSearch(a, n, key)
{
    for (int i=0; i<n; i++)
    {
        if(a[i] == key)
            return i;
    }
}
```

Ques2 Pseudo code for iterative and Recursive insertion sort. Insertion sort called online sorting. why? what about other sorting algos.

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

→ insertionSort (int a[], int n) // Recursive code

```
{
    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 whole problem data together from the beginning & is required to output an answer which solve the problem at hand.

Ques3) 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	$O(n^2)$	$O(n^2)$	$O(n^2)$

Sorting	Best	worst	Average
(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	$O(n+m)$	$O(n+m)$	$O(n+m)$
(vii) Heap sort	$O(n \log n)$	$O(\log n)$	$O(n \log n)$

Ques 4 > Sorting Techniques that are stable, inplace and online sorting.

Sorting Technique	Inplace	Stable	Online
1. Bubble sort	✓	✓	×
2. Selection Sort	✓	×	×
3. Insertion Sort	✓	✓	✓
4. Quick sort	✓	×	×
5. Merge Sort	×	✓	×
6. Count sort	×	✓	×
7. Heap Sort	✓	×	×

Ques 5 Recursive / iterative Pseudo code for Binary search. Time and space complexity of linear and Binary search.

→ int BinarySearch(a, l, r, x) // Recursive Pseudo-code

```
{
    while ( l <= r)
    {
        mid = (l+r)/2 ;
        If (n > a[mid])
            return BinarySearch(a, mid+1, r, n);
        else if (n < a[mid])
            return BinarySearch(a, l, mid-1, r);
        else
            return mid;
    }
}
```

→ int binarySearch(a, n, key) // Iterative pseudo-code

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

Searching Technique	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

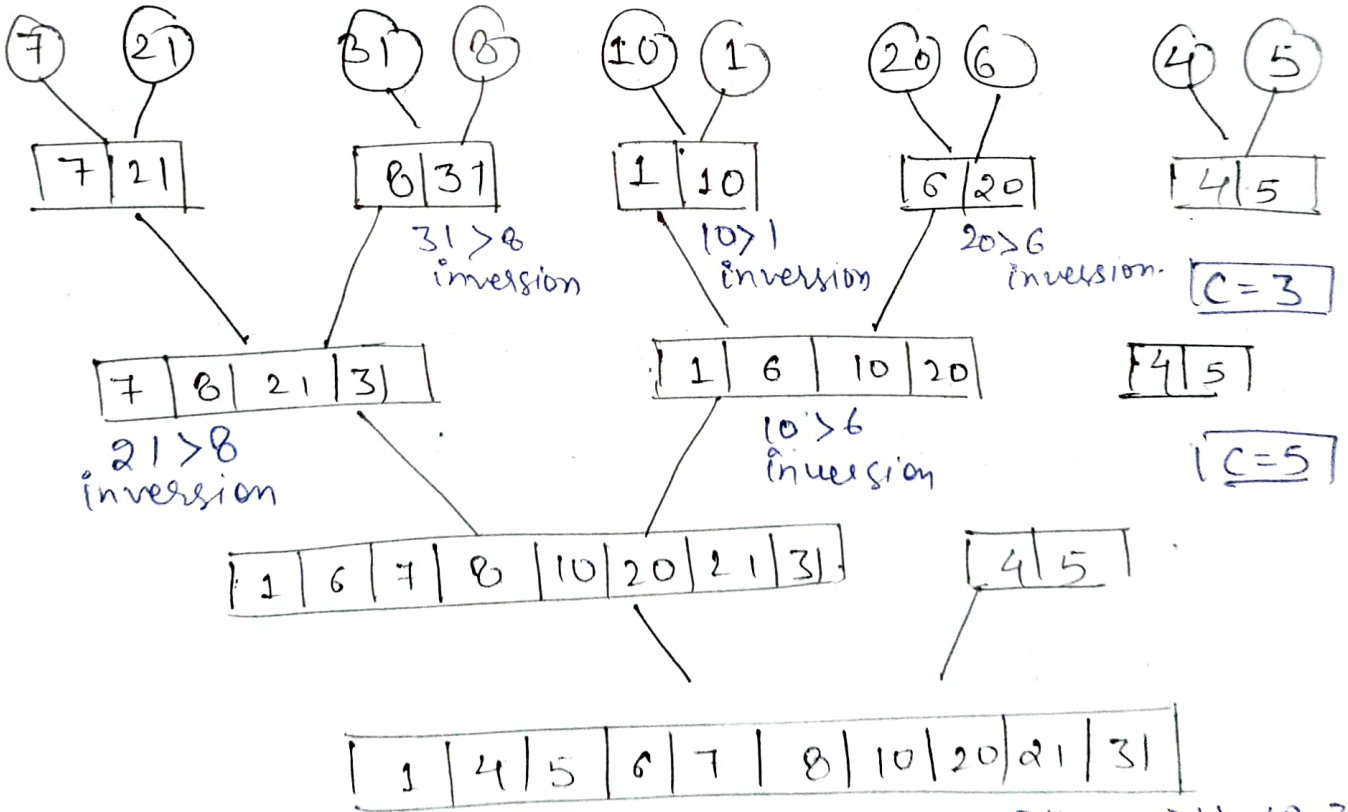
```
findIndex(int a[], int n, int K)
{
int i=0, j=1;
  while(i<n && j<n)
  {
    if(i!=j && a[j]-a[i]==K || a[i]-a[j]==K)
    {
      printf("%d, %d", i, j);
    }
    else if(a[j]-a[i]<K)
    {
      j++;
    }
    else
      i++;
  }
}
```

Ques8) 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 fast as compared to other sorting algorithms. Also its time complexity is O(nlogn) But in case of a larger array, Merge sort is preferred.

Ques9) 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.

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



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

C = 17

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

Total count of inversions = 14+17
= 31

Ques 10> In which case Quick sort will give best and 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> write Recurrence relation of Merge & Quicksort in best & worst case? similarities & difference b/w Complexity of 2 Algos & why?

Quicksort :- 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 in merge sort, the array is divided in 2 equal halves
∴ Tc = 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(n) 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 least value at pos - 0 ton.

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 BubbleSort (int a[], int n)
{
    for (i=0 ton)
    {
        swaps = 0;
        for (j=0; to n-1-i)
        {
            if (a[j] > a[j+1])
            { swap(a[j], a[j+1]);
              swaps++;
            }
            if (swaps == 0)
              break;
        }
}
```

Ques 14) Your Computer has RAM of 2 GB; Given array of 4 GB for sorting which sorting 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; mainly used when data to be sorted is small.

eg: Bubble sort, Quick sort. etc