TUTORIAL-3

① Write linear search code pseudocode to search an element in a sorted array with minimum comparisions.

sol:-

```
Void search (int arr[], int n, int x)
{
  if (arr(n-1) == x)
    Cout << " found " << endl;
  int t = arr[n-1];
  arr[n-1] = x;
  for (int i=0; i<n; i++)
  {
    if (arr[i] == x)
    {
      arr[n-1] = t;
      if (i<n-1)
        Cout << " Found " << endl;
      else
        Cout << " Not found " << endl;
    }
  }
}
```

② Write pseudocode for iterative and recursive insertion sort. Insertion sort is called Online sorting. why? What about other sorting algorithms that has been discussed in lectures.

sol: iterative

```
void i-sort (int a[], int n)
{
  for (int i=1; i<n; i++)
    int t = a[i];
    int j = i;
```

```
    while (j>0 && a[j-1] > t)
    {
      a[j] = a[j-1];
      j--;
    }
    a[j] = t
  }
}
```

Recursive

```
void i_sort (int a[], int n)
{
  if (n<=1)
    return;
    i_sort(arr, n-1);
  int last = arr[n-1];
    int j = n-2;
    while (j>=0 && a[j] > last)
    {
      a[j+1] = a[j];
      j--;
    }
    a[j+1] = last;
}
```

Insertion sort is called online sorting because it does not need to know anything about what value it will sort the information is requested while the algorithm is running.

Only insertion is online sorting among all sorts.

(3) Complexity of all sorting algorithms that has been discussed in lectures.

sol

Sorting type	Worst Case (T) / (S)	Avg Case	Best Case
1. Bubble sort	$O(n^2)$ / $O(1)$	$O(n^2)$	$O(n)$
2. Insertion sort	$O(n^2)$ / $O(1)$	$O(n^2)$	$O(n)$
3. Selection sort	$O(n^2)$ / $O(1)$	$O(n^2)$	$O(n^2)$
4. Quick sort	$O(n^2)$ / $O(\log n)$	$O(n \log n)$	$O(n \log n)$
5. Merge sort	$O(n \log n)$ / $O(n)$	$O(n \log n)$	$O(n \log n)$
6. Count sort	$O(k)$ / $O(k)$	$O(N+k)$	$O(n)$
7. Randomized Quicksort	$O(n^2)$ / $O(\log n)$	$O(n \log n)$	$O(n \log n)$
8. Heap sort.	$O(n \log n)$ / $O(n)$	$O(n \log n)$	$O(n \log n)$

(4) Divide all the sorting algorithms into inplace / stable / online.

sol

Sorting type	Inplace	stable	Online
1. Bubble	✓	✓	
2. Insertion	✓	×	✓
3. Selection	✓	×	×
4. Quick.	✓	×	
5. Merge.	×	✓	×
6. Count.	×	✓	
7. Randomised Quick	✓	×	
8. Heap	✓	×	

(5) Write recursive/iterative pseudocode for beinary sea
What is the time and space complexity of linear & binary

Ans Recursive

```
int binarySearch (int [] A; int low, int high, int x)
{ if (low > high)
  {
    return -1;
  }
int mid = (low + high) / 2;
  if (x == A[mid] {
     return mid;
  } else if (x < A [mid])
    {
      return binarysearch (A, low, mid - 1, x);
       }
     else {
       return binarySearch (A, mid + 1, high, x);
      }
   }
```

Iterative

```
int binarysearch (int [] A, int x)
{ int low = 0, high = A.length - 1;
   while (low <= high)
    { int mid = (low + high) / 2;
      if (x == A[mid])
      {
        return mid;
```

```
    else if (x < A[mid])
    {
      high = mid - 1;
    }
    else {
      low = mid + 1
    }
  }
  return -1;
}
```

Binary Search

Time Complexity of recursive = O(log n)

iterative = O(log n)

Space Complexity of recursive = O(log n)

iterative - O(1) / O(log n)

Best Avg / worst

Linear Search

Time complexity of recursive - O(n)

iterative - O(n)

Space Complexity of recursive - O(nm)

iterative - O(1)

(6) Write recurrence relation for binary recursive search

Ans Recurrence relation is $T(n) = T(n/2) + 1$.

where $t(n)$ is the required time for binary search in an array of size n.

(8) Which sorting is best for practical uses? Explain.

Ans:- Quicksort is fastest general purpose sort. In most of practical situations, Quicksort is method of choice. If stability is important and space is available, merge sort might be sort.

(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.

Sol:- Inversion count for an array indicates - how far (or close) the array is from being sorted.

arr[] = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5}

inversions - (7,21) (7,31) (7,8) (7,10) (7,20) (21,31) (8,10) (8,20) (10,20) (1,20) (1,6) (1,4) (1,5) (4,5)

No:of inversions (14)

(10) In which cases Quicksort will give the best and worst case time complexity.

Sol:- The Worst case time complexity of Quicksort is $O(n^2)$

The Worst case occurs when the picked pivot is always extrem. This happens when input array is sorted or reverse sorted and either first or last element is picked as pivot.

The best case Time Complexity of Quick sort is $O(n \log n)$.

The best case occurs when we select pivot as a main element.

(11) Write recurrence relation of merge sort & Quicksort in best and worst case? What are the similarities and differences between complexities of two algorithms & why?

Merge Sort — Best Case — Worst Case

Recurrence relation $T(n) = 2T(n/2) + n$.

Quick Sort

Recurrence relation $T(n) = 2T(n/2) + n$ $T(n-1) + n$.

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

Sol A version of stable selection can be written in which instead of swapping values, three values can be inserted.

(13) Bubble sort scans whole array even when array is sorted. Can you modify the bubble sort so that it doesn't scan the whole array once it is sorted.

```
void bubbleSort (int A[], int n)
{ for(int i=0; i<n; i++)
& int swap = 0;
  for (int j = 0; j < n - 1 - i; j++)
```

```
if (A[j] > A[j+1])
{
    int t = A[i];
    A[j] = A[j+1];
    A[j+1] = t;
    swap++;
}
}
if (swap == 0)
    break;
}
}
```

(14) Your Computer has a RAM of 2 GB and you are given an array of 4 GB for sorting. Which algorithm your going to use for this purpose and why? Also explain the concept of external & internal sorting.

(A) The Algorithm that is going to be used to implement this task is external sorting. The 4 GB array is stored in main memory and sorted by Quick sort. As Quick sort is one of the external sorting methods which deals with data to be stored in RAM & is based on divide and Conquer method.

EXTERNAL SORTING:

This is a type of sorting algorithm that deals with large amount of data. It is used when the data to be stored does not fit in main memory and is stored in external memory.

Ex - Quick sort, Merge sort

INTERNAL SORTING

It is a type of sorting algorithm that deals with data which can be adjusted in the main memory.

Ex - Bubble sort, selection sort.