TUTORIAL - 3

Q1 Write linear search pseudocode to search an element in a sorted array with min. comparisons.

Ans

```
Void linear Search (int A[], int n, int key){
        int flag =0;
      for( int i=0; i<n; i++)
          if (A [i]== Key ){
              flag =1;
              break;
          } }
      if (flag ==0)
         Cout<<"Not found";
         else
         Cout<<"found";
```

Q2 Write pseudocode for iterative algorithm for loop in Iterative for(i=1 to n-1)

```
{   t= A[i]; j= i-1;
   while ( j>=0  A[i]>t) {
         (A [j+1] = A[i]
            j--;
         }
      A [j+1] = t;
   }
```

Recursive

```
   Void insertionSort (int arr[], int n){
         if (n<=1)
            return j;
```

Q3 $T(n) = 3T(n/2) + n^2$

$n \log_2 3 = n^{1.5}$

$n^{1.5} > n$

Ans 3 insertion sort (arr, n-1)

int last = arr [n-1], j = n-2;

while(j>0 && arr [i] > last)

arr [j+1] = arr [j]

j--;

arr [j+1] = last;

}

Insertion Sort is called outline sorting an input element for iteration and produces a partial solution without require access to offline algorithm.

Q4 Complexity of all sorting algorithm has been discussed.

Algorithm	Best	Average	Worst
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Count sort	$O(n)$	$O(n)$	$O(n)$
Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Heap Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Inplace stable instance

Algorithm	Inplace	Stable	Online
Bubble	✓	✓	✗
Selection	✓	✗	✗
Insertion	✓	✓	✓
Count	✗	✓	✗

Merge	✗	✓	✗
Quick	✓	✗	✗
Heap	✓	✗	✗

Q5 Write pseudo code for binary search the time space.

Ans5 Recursive =

```
int binary (int arr[], int l, int r, int Key)
  { if (r >= l)
     { int mid = l+(r-l)/2;
       if (arr[mid] == Key)
            return mid;
     if (arr [mid] > Key)
          return binary (arr, mid-1, Key)
  else
     return (arr, mid+1, r, key)
    }
  return -1;
   }
```

```
int binary search (int arr, int l, int r, int Key)
   { while (l<=r) {
       int m = l+(r-l)/2;
      if (arr [m] ==Key);
             return m;
   if (arr [m] < Key)
        l = m+1;
   else
      r = m-1;
     }
  return -1;
  }
```

Q7 Find two indices such that A[i]+A[j]=K in minimum.

Ans7. Void Sum (int A[], int K, int n)

```
{
  Sort (A, A+n)
  int i=0; j=n-1;
while (i<j)
 if (A[i]+A[j]==K)
   break;
else if (A[i]+A[j] >K)
     j--;
 print (i, j) ;
```

Q8 Which sorting best for practical uses? Explain.

Ans 8 For practical uses, ~~so~~ it would be best for very large data. Further, time complexity of merge Sort is same in all cases, that is $O(n(\log n))$.