Ayush Agarwal
Sec-I
Roll No-30

Tutorial 3

Ques- Write linear search Pseudo code to search an element in a sorted Array with min no of comparisions.

```
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)
    print ("Not found")
  else print ("Found"); }.
```

Ques Pseudo code for iterative & recursive insertion sort. Insertion sort is also called online sort why? what about other sorting that was been discussed.

Iterative :-

```
for i=1 to n-1
    t=A[i], j=i-1;
  while (j>=0 && A[j]>t)
    if A[j+1] = A[j])
      { j--;
        A[j+1]=t;
      }
```

Recursion

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

Insertion sort is the online Algorithm bec insertion sort considers one input element per iteration & produces a partial soln without considering future elements.

Q3. Complexity of all sorting algo that has been discussed.

Algo	worst case O	Best case O	Avg case.
Bubble	$O(n^2)$	$O(n)$	$O(n^2)$
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion	$O(n^2)$	$O(n)$	$O(n^2)$
count	$O(n+k)$	$O(n+k)$	$O(n+k)$
quick	$O(n^2)$	$O(n \log n)$	$O(n \log n)$
merge	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
heap	$O(n \log n)$.	$O(n \log n)$	$O(n \log n)$

Q4 Divide all sorting algo into inplace / stable / online.

Algo	Inplace	stable	Inline
Bubble	✓	✓	x
Selection	✓	x	x
Insertion	✓	✓	✓
count	x	✓	x
merge	x	✓	x
quick	✓	x	x
heap.	✓	x	x

Ques write recursive / iterative pseudo code for BS
what is time / space complexity linear / Binary
search (recursive & iterative)

Recursive linear

```
int binarySearch (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;
  }
```

Recursive

```
int b-search (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 b-search (arr, l, mid-1, key)
       "          "     (arr, mid+1, r, key)
  }
  return -1
  }
```

Ques6 Write recursive relation for binary search (recurs

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

Ques7 Find two indices such that $A[j] + A[j] = k$ in minimum time complexity.

```
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--;
    else
      i++;
  }
}
print (i, j);
}
```

Here sort function has $O(n \log n)$ complexity & for while loop it is $O(n)$.
Overall complexity $= O(n \log n)$.

Ques8 Which sorting is best for practical uses? Explain
→ In practical use, we mostly prefer merge sort because of its stability & it can best for very large data. Furthermore the time complexity of merge sort is same in all cases i.e. $O(n \log n)$.

Ques9 What do you mean by number of inversions in an array? Count the number of inversions in an array $[] = \{7, 21, 31, 8, 10, 1, 20, 8, 4, 5\}$ using merge sort.
Inversion count for an array indicates how far or close the array is from being sorted. If the array is already sorted, inversion count is zero. but if array is sorted in reverse order then inversion count is maximum

Pseudo code for Inversion Count

```
int getInvCount (int arr[], int n)
{
    int c=0;
    for (i=0; i<n-1; i++)
    {
        for(int j=i+1; j<n; j++)
        {
            if(arr[i]>arr[j])
                c++;
        }
    } return c;
}
```

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

Total Inversion ⇒ 31

4) Write Recurrence Relation of merge / Quick sort in best / worst case - write the order of both the complexities of two algo & why?

Algo	Recurrence Relation	
	Best Case	worst Case
Quick sort	$T(n) = 2T(n/2)+n$	$T(n) = T(n-1)+n$
Merge sort	$T(n) = 2T(n/2)+n$	$T(n) = 2T(n/2)+n$

Both Algorithms are based on divide & conquer Algo. Both the Algo have the same time complexity in best or average case bec both the algo divides array into sub parts, sort them & finally merge are the sorted parts.

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

Selection sort can be made stable if instead of swapping the minimum element is placed in its position without swapping is by placing the number in its position by pushing every element one step forward. In simple words use insertion sort which means inserting element on its correct place.

(3) 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.

We can modify bubble sort by placing a flag variable If array is already sorted we can know the process by checking the flag variable if it is value changes or not.

Pseudo code for Bubble sort (modified)

```
void bubble (int A[], int n)
  {for (int i=0 ; i<n ; i++)
     { int swaps=0 ;
   for (int j=0 ; j< n-i-1; j++)
     { if(A[j] > A[j+1])
     {swap(A[j] , A[j+1]);
        swap++;
     }
   if (swaps ==0)
        break;
  }
```

Q14 Your comp has RAM of 2GB and you are given an array of 4GB of sorting. which algo you are going to use for this purpose & why? Also explain the concept of Internal / External sorting.

⇒ For the array of 4GB, we use external sorting bec array size is Greater than RAM of our comp.

External sorting → These are sorting algo that can handle large data amounts which cannot fit in the main memory. They only a part of array resides in RAM during execution.
eg → K way merge sort.

Internal sorting → These are sorting algo where the whole array needs to be in RAM during execution.
eg → Bubble sort.