Tutorial - 3

Name: HARSH RASTOGI Class Roll No - 6
Section: F University Roll No - 2016761

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

Ans:
```
for (i=0 to n)
{
    if (arr[i] == value)
        // element from d

}
```

2. Write pseudo code for iterative & recursive insertion sort. Insertion sort is called Online sorting. Why? What about other sorting algorithms that has been discussed?

Ans:

Iterative

```
void insertionsort (int arr[], int n)
{
  for (int i=1; i<n; i++)
  {
      j = i-1;
      x = arr [i];
```

```
while( j>-1 && arr[j] > x)
{
    arr[j+1] = arr[j];
        j--;

  }
 arr[j+1] = x;
 }
}
```

Recursive:-

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

Insertion sort is called 'Online Sort' because it does not used to know anything about what values it will sort and information is requested while algorithm is running.

Other Sorting algorithms:-

1.) Bubble sort

2.) Quick Sort

3.) Merge Sort

4.) Selection Sort

5.) Heap Sort

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

Ans:-

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

4. Divide all sorting algorithms into inplace / stable / online sorting.

Ans:- Inplace sorting

1) Bubble sort
2) Selection sort
3) Insertion sort
4) Quick sort
5) Heap sort

Stable sorting

1) Merge sort
2) Bubble sort
3) Insertion sort
4) Count sort

Online sorting

Insertion sort

5. Write recursive / iterative pseudocode for binary search. What is the time complexity of linear & binary search.

Ans:- Iterative

```
int b_search(int arr[], int l, int r, int key)
{
  while(l<=r) {
  int m=((l+r)/2);
  if(arr[m]==key)
     return m;
  else if(key<arr[m])
```

```
r=m-1;
else
    l=m+1;
}
return -1;
}
```

Recursive

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

Time complexity

1) Linear search - O(n)
2) Binary search - O(logn)

6. Write recurrence relation for binary recurrence search.

Ans:- $T(n) = T(n/2) + 1$ — ①

$T(n/2) = T(n/4) + 1$ — ②

$T(n/4) = T(n/8) + 1$ — ③

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

$T(n) = T(n/4) + 1 + 1$

$T(n) = T(n/8) + 1 + 1 + 1$

⋮

$T(n/2^n) + 1$ (k times)

Let $2^k = n$

$k = \log n$

$T(n) = T(n/n) + \log n$

$T(n) = T(1) + \log(n)$

$T(n) = O(\log n)$

Ans.

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

Ans:-
```
for(i=0; i<n; i++)
{
    for(int j=0; j<n; j++).
    {
        if(a[i] + a[j] == k)
        printf("%d %d", i, j);
    }
}
```

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

Ans:-

Quick sort is fastest general-purpose sort. In most practical situations, quicksort is the method of choice as stability is important and space is ~~als~~ available, mergesort might be best.

9. What do you mean by inversions in an array? Count the number of inversions in Array arr[ ] = {7, 21, 31, 8, 19, 1, 20, 6, 4, 5} using merge sort?

Ans:- A Pair (A[i], A[j]) is said to be inversion if

* A[i] > A[j]
* i < j
* Total no. of inversions in given array are 31 using merge sort.

10. In which cases Quick Sort will give least & worst case time complexity

Ans:- Worst Case $O(n^2)$ →

The worst case occurs when the pivot element is an extreme (smallest / largest) element. This happens when input array is sorted or reverse sorted and either first or last element is selected as pivot.

Best Case $O(n \log n)$

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

11. Write recurrence relation of Merge/Quick sort in best and worst case. What are similarities & difference between complexities of 2 algorithm & why?

Ans Merge Sort

Best Case → $T(n) = 2T(n/2) + O(n)$
Worst Case → $T(n) = 2T(n/2) + O(n)$

$\left.\begin{matrix} \\ \\ \end{matrix}\right\} O(n \log n)$

Quick Sort

Best Case → $T(n) = 2T(n/2) + O(n) \rightarrow O(n \log n)$
Worst Case → $T(n) = T(n-1) + O(n) \rightarrow O(n^2)$

In quicksort, arrays of elements is divided into 2 parts repeatedly until it is not possible to divide it further.
In mergesort, the element are split into 2 subarray (n/2) again & again until only one element is left.

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

Ans:

```
for (int i=0; i<n; i++)
{
  int min=i;
  for (int j=i+1; j<n; j++)
  {
    if(a[min] > a[j])
    {
      min=j;
    }
  }
  int key = a[min];
  while (min>i)
  {
    a[min]=a[min-j];
    min--;
  }
  a[i]= key;
}
```

13. Bubble sort scans array even when array is sorted. Can you modify the bubble sort so that it does not scan the whole array once it is sorted.

Ans:

A better version of bubble sort, known as m bubble sort, includes a flag that is set of a exchange is made than it should be called the array is already order because no 2 elements used to be switched.

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