

Tutorial-3

Ques.1) Write linear Search pseudocode to search an element in a Sorted array with minimum Comparisons?

```
for (int i=0 to n)
{
    if (arr[i] == Value)
        // element found
        return Value;
}
return -1;
```

Ques.2) Write Pseudo Code for iterative and recursive insertion sort. Insertion sort is called online sorting. Why? what about other sorting algorithms that has been discussed in lecture?

Iterative:

```
void insertion_sort (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 insertion_sort (int arr[], int n)
{
    if (n <= 1) return;
    insertion_sort (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;
}
```

Shahid

Insertion Sort is called online sort because it does not need to know anything about values it will sort and information is requested while algorithm is running.

Other Sorting Algorithms:-

-) Bubble Sort
-) Selection Sort
-) Quick Sort
-) Heap Sort
-) Merge Sort

Ques 3) Complexity of all sorting algorithm that has been discussed in lectures.

Sorting Algo	Best	Worst	Aug.
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	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Ques 4) Divide all sorting algorithm into inplace / stable / Online

INPLACE SORTING

Bubble Sort
Selection Sort
Insertion Sort
Quick Sort
Heap Sort

STABLE SORTING

Merge Sort
Bubble Sort
Insertion Sort
Count Sort

ONLINE SORTING

Insertion Sort.

Ques 5) What recursive/iterative Pseudo Code for binary search. What is the Time and Space Complexity of Linear & Binary Search.

Iterative:

```

int binarySearch(int arr[], int l, int r, int Key)
{
    while (l <= r)
    {
        int m = l + (r - l) / 2;
    }
}

```

Shubh

```

if (arr[m] == key)
    return m;
else if (key < arr[m])
    r = m - 1;
else
    l = m + 1;
}
return -1;
}

```

Time Complexity: -

Linear Search - $O(n)$

Binary Search - $O(\log n)$

Recursive:

```

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;
        else if (key < arr[m])
            return binarySearch(arr, l, mid - 1, key);
        else
            return binarySearch(arr, mid + 1, r, key);
    }
    return -1;
}

```

Ques-6) Write recurrence relation for binary search (recursive)

$$T(n) = T(n/2) + 1 \quad \text{--- (1)}$$

$$T(n/2) = T(n/4) + 1 \quad \text{--- (2)}$$

$$T(n/4) = T(n/8) + 1 \quad \text{--- (3)}$$

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

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

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

1

$$T(n/2^k) + 1 \text{ (k times)}$$

$$\text{Let } 2^k = n$$

$$k = \log n$$

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

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

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

Shubham

Ques-7) Find two index such that $A[i] + A[j] = K$ in minimum time complexity

```
for (i=0; i<n; i++)  
{  
    for (j=0; j<n; j++)  
    {  
        if (a[i] + a[j] == K)  
            printf("%d %d", i, j);  
    }  
}
```

Ques-8) which sorting is best for practical uses?

Quick Sort is the fastest general purpose sort in most practical situations, quick sort is the method of choice. As stability is important and if space is available, merge sort might be best.

Ques-9) what do you mean by inversion in an array? Count the number of inversion in Array arr[] = {7, 21, 31, 8, 10, 1, 20, 4, 5} using merge sort.

A pair $(A[i], A[j])$ is said to be inverted if

- $A[i] > A[j]$
- $i < j$

Total no. of inversion in given array is 31 using merge sort

Ques-10) In which case, quick sort will give least & worst case T.C.

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 & either first or last element is selected as pivot.

Best Case $O(n \log n)$:- The best case occurs when we select pivot element as a mean element.

Ques-11) write recurrence relation of merge/quick sort in best & worst case

Merge Sort:
Best Case - $T(n) = 2T(n/2) + O(n)$
Worst Case - $T(n) = 2T(n/2) + O(n)$ } $O(n \log n)$

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

Ques-12) Selection Sort is not stable by default but can you write a version of stable Selection Sort code?

```
for(i=0; i < n-1; 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-1];
        min--;
    }
    a[i]=key;
}
```

Ques-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.

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

Shreshth