Aditi Banerji
38 - F

Tutorial- 3

Q1. Write linear search Pseudo code to search an element in a sorted array with minimum comparisoins.

```
for (int i= 0; i<n; i++)
{
    if (a[i] == Keyvalue)
    {
        cout << "element found";
    }
}
```

Q2. Write Pseudo code for iterative and Reccursive insertion sort. Insertion sort is called online sorting. Why? What about other sorting Algorithms that has been discussed in lectures?

* iterative :-

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

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

Aditi

* Recursive :-

```
void insertion-sort (int a[], int n)
{
    if (n<=1)
    {
        return;
    }
    insertion-sort (a, n-1);
    int last = a[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 Sort because it doesnot need to know anything about what values it will sort and the information is requested WHILE the algorithm is Running.

* Other sorting Algorithm :-

1) Bubble Sort
2) Quick Sort
3) Merge Sort
4) Selection Sort
5) Heap Sort.

Aditi

Q3. Complexity of all the Sorting Algorithm that has been discussed in lectures. ③

Sorting	Best	Worst	Average
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion	$O(n)$	$O(n^2)$	$O(n^2)$
Quick	$O(n\log n)$	$O(n^2)$	$O(n\log n)$
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 the sorting Algorithms into inplace / stable / online sorting.

Inplace Sorting	stable sorting	online Sorting
Bubble Sort	Merge Sort	Insertion
Selection Sort	Bubble	
Insertion Sort	Insertion	
Quick Sort	Count	
Heap Sort		

Q5. Write Recursive / iterative Pseudo code for Binary Search. what is the time and space Complexity of Linear and Binary Search (Recursive & iterative).

* iterative :

```
int Binary_search (int a[], int l, int r, int key)
{
    while ( l<=r )
    {
        int m = l+(r-l)/2;
```

Aditi

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

* Recursive :-

```
    int Binary_Search (int a[], int l, int r, int key)
{
    {   if (l>r)
            return 0;
        }
    int mid = l+(r-l)/2;
    {   if (a[mid] == key)
            return mid;
    }
    else if ( a[mid] <key)
    {
        return Binary_Search (a, mid+1, r, key);
    }
    else
    {
        return Binary_Search (a, l, mid-1, key);
    }
}
```

Aditi

* time complexity :-

1) Linear Search →

iterative : $O(n)$

Recursive : $O(n)$

2) Binary Search →

iterative : $O(\log n)$

Recursive : $O(\log n)$

* Space Complexity :-

1) Linear Search → $O(1)$

2) Binary Search → $O(1)$

Q6. Write Reoccurence Relation for Binary Reccursive Search.

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

put $n/2$ $\quad T(n/2) = T(n/4) + 1 \quad$ — (ii)

put $n/4$ $\quad T(n/4) = T(n/8) + 1 \quad$ — (iii)

from (i) $\quad T(n) = T(n/2) + 1 \quad$ from (ii)

$T(n) = T(n/4) + 1 + 1 \quad$ from (iii)

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

⋮

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

Let $2^k = n$

$k = \log n$

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

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

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

Aditi

Q7. Find two Indexes such that $a[i]+a[j] = k$ in minimum time Complexity. (6)

```
for ( int i=0; i<n ; i++ )
{
    for ( int j=0; j<n; j++ )
    {
        if ( a[i]+a[j]==k )
    }
    cout << i << j ;
```

Q8. which sorting is best for practical uses? Explain.

Quick Sort is the fastest general purpose sort. In most practical situations, Quick Sort is the method of choice as stability is important and space is available, Merge Sort might be best.

Q9. what do you mean by number of inversions in an array? Count the number of inversions in Array $a[] = \{7, 21, 31, 8, 10, 1, 20, 6, 4, 5\}$ using Merge Sort.

→ A pair (a[i], a[j]) is said to be inversion if

$a[i] > a[j]$ and if $i < j$

→ Total Number of inversions in given array are 31 using Merge Sort.

Aditi

Q10. In which cases Quick Sort will give the best and the worst case Time Complexity?

* <u>Worst Case ($O(n^2)$)</u> :- The worst case occurs when the picked pivot is always an extreme (smallest or largest) element. This happens when input array is sorted or Reverse sorted Array and either first or last element is picked as pivot.

* <u>Best Case ($O(n\log n)$)</u> :- The Best occurs when we will select pivot element as a mean element.

Q11. Write Recurrence Relation of Merge and Quick Sort in best and Worst Case? What are the similarities and differences between Complexities of two Algorithm and why?

* Merge sort :- 1) Best Case : $T(n) = 2T(n/2) + O(n)$

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

ie $O(n\log n)$

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

2) Worst Case : $T(n) = T(n-1) + O(n) \rightarrow O(n^2)$

→ In Quick Sort the array of elements is divided into parts repeatedly until it is not possible to divide it further. It is not necessary to divide half.

→ In Merge Sort the elements are split into two sub-array ($n/2$) again and again until only one element is left.

-Aditi

Q12. Selection Sort is not stable by default but can you write a version of stable selection. (8)

```
for (int 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;
}
```

Q13. Bubble Sort Scans array even when Array is Sorted. Can you modify the Bubble Sort so that it doesnot Scan the whole Array once it is Sorted.

A Better version of Bubble Sort, known as Bubble sort, includes a flag that is set if exchange is made after an entire pass over the array. If no exchange is made, then it should be the array is already order because no two element need to be switched. In that Case Sort is

```
void Bubble (int a[], int n)
{
    int swaps =0;
    for (int j=0; j<n-i-j; j++)
    {
```

Aditi

```
        if ( a[j] > a[j+1] )
        {
            int t = a[j];
            a[j] = a[j+1];
            a[j+1] = t;

            swaps ++;
        }
    }

    if ( swaps == 0 )
        break;

}
}
```

Aditi