Tutorial - 03

Ques-1. Write linear search pseudocode to search an element in an array with minimum comparisions.

Pseudocode for Improved Linear Search

```
Linear_Search ( int array [] , int Value )
 {
  for ( int left = 0 ; left <= right ; )
    {
      if ( array [left] == Value )
      {
          position = left ;
      printf ("Element found at %d position", position+1);
          break;
      }
    if(array [right] == Value)
    {
      position = right ;
      printf ("Element found at %d", position+1);
      break;
    }
    left ++;
    right --;
  }
  if ( position == -1)
  {
    printf ("Element not found");
  }
}
```

Ques--2. Write pseudocode for iterative and recursive insertion sort. Explain why it is called insertion sort. What about other Sorting algorithm?

Pseudocode for iterative insertion sort.

```
Insertion_sort (int array [])
{
 for i=1 to n
   key ← A[i];
    j = i-1;
 while((j>=0) && A[j] > key)
 {
    A[j+1] ← A[j];
      j← j-1;
 }
A[j+1] ← key;
 }
}
```

Pseudocode for Recursive Insertion Sort

```
Recursive_insertion_sort (int array [], int i, int size)
{
     Value ← array [i];
       int j ← i;
}
while (j>0 && array [j-1] > Value)
{
   array [j] ← array [j-1];
    j-1;
}
```

```
    array [j] = value;
    if ((j+1) <= size)
  }
Recursive_Insertion_sort (array, i+1, size);
  }
}
```

An Online algorithm is one that can process its input piece by piece in a serial fashion i.e., in the order that the input is fed to the algorithm without having the entire input from the start. Since insertion sort can sort the array as it recieves it, therefore it is called as Online Sorting Algorithm.

Sorting Algorithm

Merge Sort : Offline Sort

Selection Sort : Offline Sort

Quick Sort : Stable Sort

Counting Sort : Stable Sort

Radix Sort : Stable Sort

Heap Sort : In-place Sort.

Ques-3. Complexities of all Sorting Algorithms discussed so far :

Sorting Algorithms	Time Complexity		
	Worst	Average	Best
Bubble Sort	$O(N^2)$	$O(N^2)$	$O(N)$
Selection Sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
Insertion Sort	$O(N^2)$	$O(N^2)$	$O(N)$
Quick Sort	$O(N^2)$	$O(N \log N)$	$O(N \log N)$
Merge Sort	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
Heap Sort	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
Counting Sort	$O(n+k)$	$O(n+k)$	$O(n+k)$

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

Algorithms	Type
Bubble Sort	In-place
Insertion Sort	Online
Selection Sort	In-place
Merge Sort	Stable
Quick Sort	In-place
Counting Sort	Stable
Heap Sort	In-place.

Ques-5. Write recursive / iterative code for binary search. What is time complexity and space complexity of linear and binary search?

- ## Pseudocode for Iterative Binary Search

```
int iterative_binary_search (int A[], int Value)
 {
   int i = 0, j = A.size() - 1;
   while (i <= j)
    {
      mid = (low + high) / 2;
     if (Value == A[mid])
       return mid;
    else if (Value < A[mid])
        i = mid + 1;
    else
        j = mid - 1;
    }
 return -1;
}
```

- ## Pseudocode for Recursive Binary Search

```
int recursive_binary_search (int arr[], int l, int r, int
                                                  Value)
  {
    if (r >= l)
     {
        mid ← l + (r-1) / 2;
    if (arr[mid] == Value)
      return mid;
    if (arr[mid] > Value)
      return recursive_binary_search (arr, l, mid - 1, Value);
    return binary_search (arr, mid + 1, r, Value);
      recursive_
  }
 return -1;
```

Shruti

	Iterative			Recursive	
	W	Avg	B	W	Avg. B
Linear Search : (i) Time Complexity	O(N)	O(N)	O(N)	O(N)	
(ii) Space Complexity	O(1)			O(nm).	
Binary Search : (i) Time Complexity	O(logN)	O(logN)	O(1)	O(log N)	
(ii) Space Complexity	O(log N)			O(1)	

Ques-6. Write recurrence relation for Binary Recursive Search.

Recurrence relation for Binary Recursive Search :

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

Put $n = \frac{n}{2}$ in eq^n (1)

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + 1$$

So, $T(n) = T\left(\frac{n}{4}\right) + 1 + 1$

Put $n = \frac{n}{4}$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + 1$$

So, $T(n) = T\left(\frac{n}{8}\right) + 1 + 1 + 1$

$\therefore T(n) = T\left(\frac{n}{2^k}\right) + k$ ——— (11)

Also, $\frac{n}{2^k} = 1$

$$n = 2^k$$

$$\log n = k \log 2$$

$$\boxed{k = \log n}$$

Put value of k in eq^n (11)

$$T(n) = T\left(\frac{n}{2} \log n\right) + \log n$$

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

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

Ques - Find two indexes such that $A[i] + A[j] = k$ in minimum time complexity.

Pseudo-code for the same is :-

```
int find_sum(int A[], int n, int k)
{
  Sort(A, n);
  for(i = 0 to n-1)
  {
    x = binary_search(A, 0, n-1, k-A[i])
    if(x)
      return 1;
  }
  return -1;
}
```

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

Quick sort is usually best or preferred for practical uses. This sorting algorithm is based on divide and conquer algorithm. It picks on elements on pivot and partitions the given list around the picked pivot. After partitioning on the basis of pivot element it is applied recursively to two sublists.

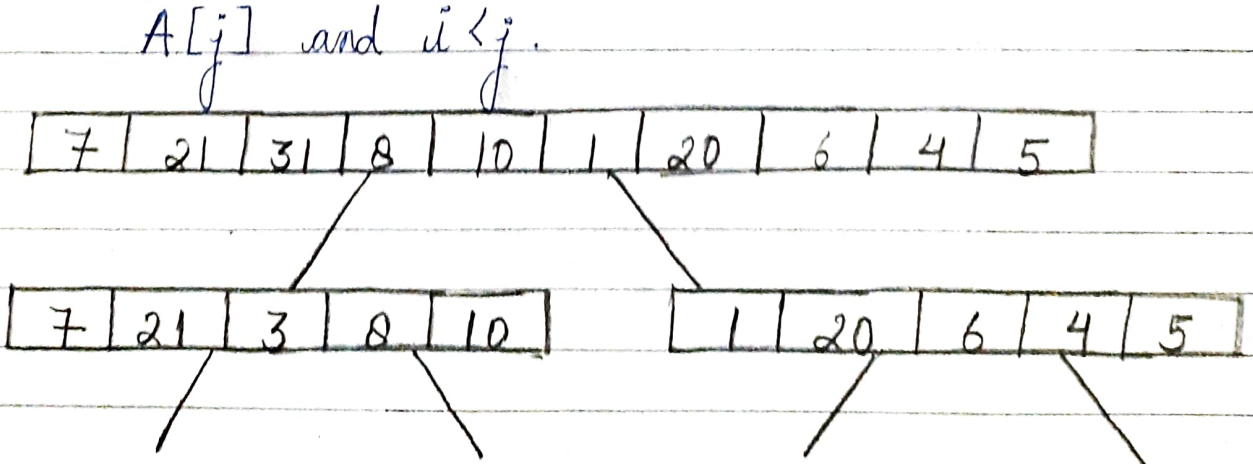
Features of Quick Sort :-

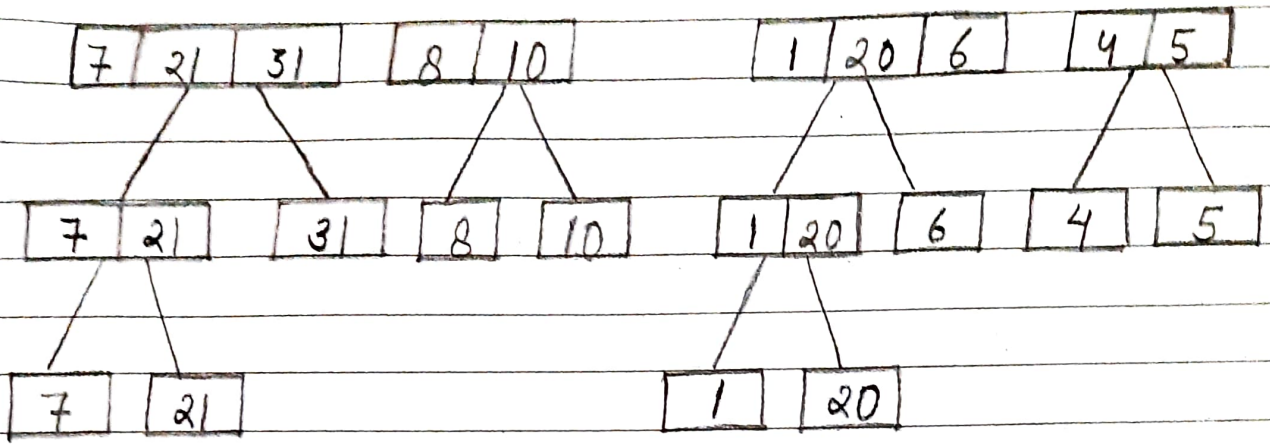
(i) It is the fastest ($O(n \log n)$) but not always as the worst case is $O(N^2)$
(ii) It is more effective for dataset that fits in the memory.
(iii) Quick Sort is an in-place sorting algorithm.

Ques-9. What do you mean by inversions in an array? Count the number of inversions in array = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5} using merge sort.

Inversions indicate how far the array is from being sorted.

So, elements A[i] and A[j] forms an inversion if A[i] > A[j] and i < j.





Total inversion count = 31.

Ques-10. In which case quick sort will give the best and worst case time complexity?

When the pivot element is always an extreme element (smallest or largest) then the worst case of quick sort occurs [$O(n^2)$]. This happens when input array is sorted or reverse sorted and either first or last element is picked as pivot element.

The best case time complexity is when the picked pivot element divides the array into two equal halves by coming exactly in the middle position.

Ques-11. Write recurrence relation of Merge and Quick Sort in best and worst case. What are the similarities and differences between complexities of two algorithms and why?

Recurrence relation of merge sort:

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

(i) Recurrence relation of quick sort (Best Case):

$$T(n) = 2T\left(\frac{n}{2}\right) + n - 1$$

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

(ii) Worst Case : $T(n) = T(n-1) + N - 1$

$$= \frac{N(N-1)}{2}$$

Similarities :-

Merge sort and quick sort are divide and conquer based algorithms. In quick sort, array is divided into two parts until it is not possible to divide it further. In merge sort, array is split into two sub-arrays until only one array is left.

Differences :-

1. In Merge Sort, array is parted into just two halves whereas in quick sort there is no compulsion of dividing array into two equal parts.

2. Merge Sort can work well with any type of dataset irrespective of its size whereas quick sort cannot work well with large datasets.

3. Quick Sort exhibits good cache locality which makes it faster than merge sort.

Ques-13. Bubble Sort scans the whole array even when the array is sorted. Can you modify?

```
void bubble_sort(int A[], int n)
{
    for(int i=0; i<n; i++)
    {
        bool flag = 0;
    for(int j=0; j<n-i-1; j++)
    {
        if(A[j] > A[j+1])
        {
            flag = =1;
        int temp = A[j+1];
        A[j+1] = A[j]
          A[j] = temp;
        }
    }
    if(flag != 0)
      return;
}
```

Ques-14. Your computer has a RAM of 2 GB and you are given an array of 4 GB for sorting. Which also you are going to use and why? Also explain concept of External and Internal Sorting.

Merge Sort is usually preferred if the size of the data to be sorted is larger than the size of RAM.

- External Sorting : It is a term for a class of sorting algorithm that can handle massive amount of data. It is required when the data being sorted does not fit into the main memory and instead resides in the external memory. Example : Merge Sort.
- Internal Sorting : If the data sorting takes place entirely within the RAM of a computer then it is called as Internal Sorting.
Example : Bubble, Insertion and Quick Sort.