TUTORIAL-3

Name :- Srijan
Section :- F
RollNo :- 30
UniversityRollNo :- 2047067

Ques :-

Solution :-

```
int linearsearch(int arr[], int n, int key) {
    for(int i=0; i<n; i++) {
        if (arr[i] == key)
            return i;
    }
    return -1;
}
```

Ques :-

Solution :- Iterative insertion sort :-

```
void insertionsort(int arr[], int n) {
    int i, j, t=0;
    for (int i=1; i<n; i++){
        t = arr[i];
        j = i-1;
        while (j>=0 && arr[j]){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = t;
    }
}
```

Recursive insertion sort :-

```
void insertionsort(int arr[], int n) {
    if (n<=1)
        return;
    insertionsort(arr, n-1);
    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 called online sorting because it does not need to know anything about what values it will sort and the information is maintained while the algorithm is running.

Ques

Solution

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	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Ques:-

Solution:-

Sorting	Inplace	Stable	Online
Selection sort	✓		
Insertion sort	✓	✓	✓
Merge sort		✓	
Quick sort	✓		
Heap sort	✓		
Bubble sort	✓	✓	

Ques:-

Solution:- Iterative binary search :-

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

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

Best case = O(1)
Avg case = O(log n)
worst case = O(log n)

Recursive binary search

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

Best case = O(1)
Avg. case = O(log n)
worst case = O(log n)

Linear search

Best case = O(1)
Avg. case = O(n)
worst case = O(n)

Ans

Solution:- Recurrance relation for binary recursive search

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

7Ques:→

Solution:→

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

8Ques:-

Solution:→ Quick sort is fastest general-purpose sort. In most practical situations quicksort is the method of choice. If stability is important and space is available, mergesort might be best.

9Ques:-

Solution:→ A pair [A[i], A[j]] is said to be inversion if

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

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

10Ques:-

Solution:→ The worst case time complexity of quicksort is $O(n^2)$.

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

The best case of quick sort is when we will select pivot as a mean element.

11Ques:→

Solution:→ Recurrence relation of:

a) merge sort ⇒ $T(n) = 2T\left(\frac{n}{2}\right) + n$

b) Quick sort ⇒ $T(n) = 2T(n/2) + n$

→ Merge sort is more efficient & works faster than quick sort in case of larger array size or datasets.
→ Worst case complexity for quick sort is $O(n^2)$ whereas $O(n \log n)$ for merge sort

---

Ques 12 :→

Solution :→

```
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 -j];
          min--;
    }
    a[i] =key;
}
```

Ques 13 →

Solution :→ A better version of bubble sort, known as modified bubble sort, includes a flag that is set if a exchange is made after an entire pass over. If no exchange is made then it should be assumed the array is already order because no two elements need 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-j ; j++)
        {
```

```
if (arr[j] > arr[j+1])
{
    int x = arr[j];
    arr[j] = arr[j+1];
    arr[j+1] = t;
    swap++;
    }
}
if (swap == 0)
break;
}
}
```