

③

Name - Samaidin
Roll no. - 24
Section - F
Date
University Roll no -
2016985

Tutorial - 3

1. while (low <= high)

mid = $\frac{low + high}{2}$

if (arr[mid] == key)

return true;

else if (arr[mid] > key)

high = mid - 1

else

low = mid + 1

return false;

2. 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;
}
```

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

Other sorting algorithm:-

- Bubble sort
- Quick sort
- Merge sort
- Selection sort
- heap sort

3.

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.

Inplace Sorting	STABLE SORTING	ONLINE SORT
Bubble sort	Merge sort	Insertion Sort
Selection sort	Bubble sort	
Insertion sort	Insertion sort	
Quick sort	Count sort.	
Heap sort		

3. Iterative:

```

int bsearch (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 b_search(arr, mid+1, s, key);

}

return -1;

}

Time Complexity.

- Linear search $O(n)$
- Binary search $O(\log n)$

Write Recursive relation for binary recursive search.

Ans

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

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

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

$$\begin{aligned} T(n) &= T(n/2) + 1 \\ &= T(n/4) + 1 + 1 \\ &= T(n/8) + 1 + 1 + 1 \end{aligned}$$

!

!

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

$$\begin{aligned} \text{LET } g^k &= n \\ k &= \log n \end{aligned}$$

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

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

$$T(n) = O(\log n) \quad \text{--- Ans.}$$

```

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

9. A pair $(A[i], A[j])$ is called to be inversion if

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

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

10. Worst case $O(n^2)$: The worst case occurs when 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 median element.

11. Merge sort :

$$\begin{array}{l} \text{Best case: } T(n) = 2T(n/2) + O(n) \\ \text{Worst case: } T(n) = 2T(n/2) + O(n) \end{array} \quad \left. \vphantom{\begin{array}{l} \text{Best case: } T(n) = 2T(n/2) + O(n) \\ \text{Worst case: } T(n) = 2T(n/2) + O(n) \end{array}} \right\} O(n \log n)$$

Quick sort

$$\begin{array}{l} \text{Best case: } T(n) = 2T(n/2) + O(n) = O(n \log n) \\ \text{Worst case: } T(n) = T(n-1) + O(n) = O(n^2) \end{array}$$

In Quicksort, array elements is divided into 2 parts repeatedly until it is not possible to divide it further.

In merge sort the elements are split into 2 subarray $(n/2)$ again, if again until only one element is left.

```
12. 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;
}

```

13. A better version of bubble sort, known as modified bubble sort, includes a flag that is set if an exchange is made after an entire pass over. If no exchange is made then it should be called the array is already sorted because no two elements need to be switched.

