Date _____

Page _____

Design and Analysis of Algorithms

NAME : AKSHIT SINGH

SECTION : F

ROLL NO: 42

COURSE: B. Tech CSE

Date

Page

Tutorial - 3

1. while (low <= high)

```
mid = (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 algorithms:-

- Bubble Sort
- Quick Sort
- Merge Sort
- Selection Sort
- Heap sort

FREEMIND

Date

Page

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 SORT	ONLINE SORT
Bubble Sort	Merge sort	Insertion Sort
Selection Sort	Bubble sort	
Insertion sort	Insertion sort	
Quick sort	Count Sort	
Heap sort		

FREEMIND

Date

Page

3. Iterative =>

```
int bisearch (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 bisearch (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 b_search (arr, l, mid-1, key);
     else
     }    return b_search (arr, mid+1, r, key);
     return -1
  }
```

FREEMIND

Date

Page

Time complexity

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

Q- Write Recurrence relation for binary recursive search.

Ans =>

$$T(n) = T(n/2) + 1 \quad \text{---(i)}$$
$$T(n/2) = T(n/4) + 1 \quad \text{---(ii)}$$
$$T(n/4) = T(n/8) + 1 \quad \text{---(iii)}$$

$$T(n) = T(n/2) + 1$$
$$= T(n/4) + 1 + 1$$
$$= T(n/8) + 1 + 1 + 1$$
$$\vdots$$
$$= T(n/2^n) + 1 \text{ (K-times)}$$

Let $2^K = n$

$K = \log n$

$$T(n) = T(n/n) + \log n$$
$$T(n) = T(1) + \log n$$
$$T(n) = O(\log n) \rightarrow \text{Answer}$$

FREEMIN

Date

Page

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 quicksort is the method of choice as stability is important and space is available, mergesort 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 occur when pivot element is an esctreme (smallest / largest) element. This happen 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 occur when we will select pivot element as mean element.

FREEMIND

Date

Page

11. Merge Sort

Best Case - $T(n) = 2T(n/2) + O(n)$
Worst Case - $T(n) = 2T(n/2) + O(n)$
$\Big\{ O(n \log n)$

Quick Sort

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

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

In merge sort the element 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-j]
          min--;
     } a[i] = key
}
```

Date

Page

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