Tutorial-3

Q: Write linear search --- comparisons:

```
for(i=0 to n)
  {  if(arr[i]==value)
  }
```

Q: Write pseudo code for iteration & recursion. --- Has been discussed?

iteration

```
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 info is requested while algo is running.

Q: Complexity of all sorting algo n

→

Sorting Algo	Best	Worst	Average.
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
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	$O(n\log n)$	$O(n\log n)$	$O(n^2)$
Merge Sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$

Q: Divide all sorting algo into inplace / stable / online sorting;

INPlace Sorting	STABLE SORTING	ONLINE SORTING
Bubble Sort	Merge Sort	Insertion Sort
Selection Sort	Bubble Sort	
insertion Sort	Insertion Sort	
Quick Sort	Count Sort	
Heap Sort		

narzo Shot by aniket

Q5:

Iterative:

```
int b_search (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;
}                    [T.C = O(n)]
```

Recursive →

```
int b_search (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,m-1,
                      key);
```

Q: In which case quick sort will give best & worst case T.C?

W.C ($O(n^2)$) – when the 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.

B.C ($O(n \log n)$) – The Best case occurs when we will select pivot element as a mean element.

Best case time complexity of quicksort is $O(N \log(N))$ & that will be when part is selected as mean element.

Q:
```
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;
    }
```

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

[T.C = O(logn)]

Q: Write recurrence relation for binary recursive search.

$T(n) = T(n/2) + 1$ — (1)

$T(n/2) = T(n/4) + 1$ — (2)

$T(n/4) = T(n/8) + 1$ — (3)

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

$= T(n/4) + 2$

$= T(n/8) + 3$

$= T(n/2^k) + k$

Let $g^k = n$

$k = \log n$

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

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

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

```
int key = a[min];
while (min > j)
{
    a[min] = a[min-j];
    min--;
}
a[j] = key;
}
```

Q:13

⇒ A better version of bubble sort, known as modified bubble sort, includes a flag that is set after an exchange is made after an entire pass over. If no exchange is made then it should be called the array is already ordered because no 2 elements need to be switched.

```
void bubble (int a[], int n)
{ for (int i=0; i<n; i++)
  { swaps = 0;
    for (int j=0; j<n-i-j; j++)
    { if (a[j] > a[j+1])
      { int t = a[j];
```

marzo Shot by aniket

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

narzo Shot by aniket