☆ Sorting Techniques :- There are different type of sorting algorithm are as follows,

1) Bubble Sort :- It is a very simple method that sorts the array elements by repeatedly moving the largest element to the highest index position of the array segment (in case of arranging elements in ascending order).

```
Bubble_Sort(a, n)
1) Repeat step 2 for i = 0 to n-1
2)     Repeat for j = 0 to n-i
3)         If a[j] > a[j+1]
           swap a[j] and a[j+1]
       [End of inner loop]
   [End of outer loop]
4) Exit.
```

Eg :- Consider an array a[] that has the following elements :

a[] = {50, 40, 30, 20, 10}

Pass 1 :-

Iteration 1 :- 50 40 30 20 10 — 50 > 40 swap
40 50 30 20 10

Iteration 2 :- 40 50 30 20 10 — 50 > 30 swap

Iteration 3 :- 40 30 50 20 10 — 50 > 20 swap.

Iteration 4 :- 40 30 20 50 10 — 50 > 10 swap.
40 30 20 10 **50**

50 is sorted at the end of pass-1.

I1:-

Pass 2:- 40 30 20 10 50 40>30 swap
Iteration 1:- 30 40 20 10 50 40>20 swap.
Iteration 3:- 30 20 40 10 50 40>10 swap
30 20 10 40 50

At the end of pass 2 last 2 numbers are in sorted list.

Pass 3:-
I 1:- 30 20 10 40 50 30>20 swap
I 2:- 20 30 10 40 50 30>10 swap
20 10 30 40 50

Pass 4:-
I 1:- 20 10 30 40 50 20>10 swap
10 20 30 40 50

Now, all the numbers are sorted.

2> Insertion Sort:-> The array of values to be sorted is divided into two sets. One that stores sorted values & another that contains unsorted values.
-> Sorting algorithm will proceed until there are elements in the unsorted set.
-> During each iteration of the algorithm, the first element in the unsorted set is picked up & insert into the correct position in the sorted set.

```
Insertion_sort(a, n)
1> Repeat step 2 to 5 for k=1 to n-1
2> set temp = a[k]
3> set j=k-1
4> Repeat while temp < a[j]
       Set a[j+1]=a[j]
       Set j=j-1
   [End of Inner Loop]
5> Set a[j+1] = temp
   [End of Loop]
6> Exit.
```

Eg:- Consider an array of int's given below, & sort them using insertion sort.

a[]=	40	10	50	63	19	82	110	54	70
Index =	0	1	2	3	4	5	6	7	8

Pass 1:- 40 10 50 63 19 82 110 54 70
a[0] is the only element in sorted list.

Pass 2:- 10 40 | 50 63 19 82 110 54 70
sorted list | Unsorted list

Pass 3:- 10 40 50 | 63 19 82 110 54 70
sorted list | Unsorted list

Pass 4:- 10 40 50 63 | 19 82 110 54 70
sorted list | Unsorted list

Pass 5 :- 10 19 40 50 63 | 82 110 54 70
Sorted list | Unsorted list.

Pass 6 :- 10 19 40 50 63 82 | 110 54 70
Sorted list | unsorted list

Pass 7 :- 10 19 40 50 63 82 110 | 54 70
Sorted list | unsorted list

Pass 8 :- 10 19 40 50 54 63 82 110 | 70
Sorted list | unsorted list

Pass 9 :- 10 19 40 50 54 63 70 82 110
Sorted list

3) Selection Sort :-> Selection sort performs worse than insertion sort algorithm, it is noted for its simplicity & also has performance advantages over more complicated algorithms in certain situations.

-> Selection sort is generally used for sortion files with very large objects & small keys.

```
Selection Sort(a,n)
1) Repeat step 2 & 3 for k=1 to n-1
2) call Smallest(a, k, n, pos)
3) swap a[k] with a[pos]
   [End of Loop]
4) Exit.

Smallest(a, k, n, pos)
1) [Initialize] Set small=a[k]
2) [Initialize] Set pos=k
3) Repeat for j= k+1 to n-1
       If small > a[j]
           set small=a[j]
           set pos= j
       [End of If]
   [End of Loop]
4) Return pos.
```

Eg:- Sort the array given below using selection sort.

a[] = 39 29 82 46 91 28 73 19

Pass 1:-

Iteration 1:- 39 29 82 46 91 28 73 19 39>29 swap
Iteration 2:- 29 39 82 46 91 28 73 19 29<82
Iteration 3:- 29 39 82 46 91 28 73 19 29<46
Iteration 4:- 29 39 82 46 91 28 73 19 29>28 swap
Iteration 5:- 28 39 82 46 91 29 73 19 28<73
Iteration 6:- 28 39 82 46 91 29 73 19 28>19 swap
Iteration 7:- 19 39 82 46 91 29 73 28
19 is sorted now after pass 1.

Pass 2:-

I 1 :- 19 39 82 46 91 29 73 28 39<82
I 2 :- 19 39 82 46 91 29 73 28 39<46
I 3 :- 19 39 82 46 91 29 73 28 39<91
I 4 :- 19 39 82 46 91 29 73 28 39>29 swap
I 5 :- 19 29 82 46 91 39 73 28 29<73
I 6 :- 19 29 82 46 91 39 73 28 29>28 swap
19 28 82 46 91 39 73 29

Pass 3:-

I 1 :- 19 28 82 46 91 39 73 29 82>46 swap
I 2 :- 19 28 46 82 91 39 73 29 ~~29<82~~ 46<91
I 3 :- 19 28 46 82 91 39 73 29 46>39 swap
I 4 :- 19 28 39 82 91 46 73 29 39<73
I 5 :- 19 28 39 82 91 46 73 29 39>29 swap
19 28 29 82 91 46 73 39

Pass 4:-

I 1:- 19 28 29 82 91 46 73 39 82<91

I2 :- 19 28 29 82 91 46 73 39 83>46 swap

I3 :- 19 28 29 46 91 82 73 39 46<73

I4 :- 19 28 29 46 91 82 73 39 46>39 swap

19 28 29 39 91 82 73 46

Pass 5:-

I 1 :- 19 28 29 39 91 82 73 41 91>82 swap

I2 :- 19 28 29 39 82 91 73 41 82>73 swap

I3 :- 19 28 29 39 73 91 82 41 73>41 swap

19 28 29 39 41 91 82 73

Pass 6:-

I1 :- 19 28 29 39 41 91 82 73 91>82 swap

I2 :- 19 28 29 39 41 82 91 73 82>73 swap

19 28 29 39 41 73 91 82

Pass 7:-

I 1 :- 19 28 29 39 41 73 91 82 91>82 swap

19 28 29 39 41 73 82 91

Now all the no's are sorted in the list.

4) Merge Sort :- This algorithm that uses the divide, conquer & combining algorithm.

-> Divide :- It means partitioning the n-element array to be sorted into two sub-arrays of n/2 elements.

-> Conquer :- It means sorting the two sub-arrays recursively using merge sort.

-> Combine :- It means merging the two sub-arays of size n/2 to produce the sorted array of n elements.

```
Merg_Sort (a, beg, end)
1) If beg < end
       Set mid = (beg + end)/2.
       Call Merge_Sort (a, beg, mid)
       Call Merge_Sort (a, mid+1, end)
       Merge (a, beg, mid, end)
   [End of If]
2) Exit.

Merge (a, beg, mid, end)
1) [Initialize] Set i = beg, j = mid+1, index = 0
2) Repeat (i ≤ mid) and (j ≤ end)
       If a[i] < a[j]
           set temp[index] = a[i]
           Set i = i+1
       Else
```

VANT PRINTERY