Sorting Algorithms

1. Bubble Sort:

- Bubble sort proceeds by scanning the list from left to right and whenever a pair of adjacent keys found out of order, those elements are swapped. This process continues till all the elements of the array are in sorted order.

Algorithm:

```
void bubbleSort( int a[ ], int n )
        {
         int i, j, temp;
         for(i = 0; i< n-1; i++)
                {
                for(j = 0; j < n-i-1; j++)
                        {
                         if(a[ j ] > a[ j+1] )
                                {
                                temp = a[ j ];
                                a[ j ] = a[ j+1 ];
                                a[ j + 1] = temp;
                                }
                        }
                }
        }
```

Example: a:

44	55	33	88	77	22	11	66

44	**55**	33	88	77	22	11	66

44	33	55	**88**	77	22	11	66

44	33	55	77	**88**	22	11	66

44	33	55	77	22	**88**	11	66

44	33	55	77	22	11	**88**	66

44	33	55	77	22	11	66	**88**

At the end of first pass, the largest element is bubbled up to the last array position.

Continuing the same process at the end of 2nd pass the output will be:

33	44	55	22	11	66	**77**	**88**

Continuing the same process at the end of 3rd pass the output will be:

33	44	22	11	55	**66**	**77**	**88**

..
...
...

Finally form the sorted array:

11	**22**	**33**	**44**	**55**	**66**	**77**	**88**

2. Selection Sort

The selection sort algorithm selects the largest element from the unsorted array in each iteration and place it in its appropriate position till the whole array is sorted. This is also called as *push-down sort*.

Note: Instead of maximum, the minimum element can also be used.

Algorithm:

```
void selectionSort(int a[], int n)
        {
        int i, j, max, pos;
        for (i=n-1; i>0; i--)
                {
        max = a[0];
                pos=0;
        for (j =1; j <=i; j++)
                        {
                if (a[j] >max)
                                {
                                max=a[j];
                                pos=j
                                }
                }
                a[pos]=a[i];
                a[i]=max;
                }
        }
```

Example:

a:

30	20	35	14	90	25	32

30	20	35	14	**90**	25	32

max=90 swap with last position element i.e.32

30	20	**35**	14	32	25	90

max=35 swap with last position not sorted yet. i.e.25

30	20	25	14	**32**	35	90

max=32, same position so no exchange

30	20	25	14	32	35	90

max=30, swap with 14

14	20	**25**	30	32	35	90

max=25, same position so no exchange

14	**20**	25	30	32	35	90

max=20, same position so no exchange

14	20	25	30	32	35	90

max=14, same position so no exchange

14	20	25	30	32	35	90

Sorted array

3. Insertion Sort

It sorts the values by inserting them into their appropriate positions in an already sorted sub array.

Algorithm:

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

Example

a:

30	20	35	14	90	25	32

30	**20**	35	14	90	25	32

20	30	**35**	14	90	25	32

20	30	35	**14**	90	25	32

14	20	30	35	**90**	25	32

14	20	30	35	90	**25**	32

14	20	25	30	35	90	**32**

14	20	25	30	32	35	90

Sorted array