Bubble sort (or) sinking sort

— The idea of bubble sort is to repeatedly move the smallest element to the lowest index position in the list.

— To find the smallest element, the bubble sort algorithm begins by comparing the first element of the list with its next element, and upto the end of the list and interchange the two elements if they are not in proper order.

— In either case after such a pass, the smaller element will be in the lowest index position of the list.

- The focus then moves to the next smaller element and the process is repeated.

- Swapping occurs only among successive elements in the list and hence only one element will be placed in its sorted order after each pass.

Routine :-

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

Example

	0	1	2	3	4	5	
	56	91	35	72	48	68	
①	35	91	56	72	48	68	i=0
②	35	56	91	72	48	68	
③	35	48	91	72	56	68	i=1
	35	48	72	91	56	68	
	35	48	56	91	72	68	i=2

0 1 2 3 4 5
35 48 56 91 72 68 i=3

35 48 56 72 91 68

35 48 56 68 91 72 i=4

35 48 56 68 91 72

$\boxed{35\quad 48\quad 56\quad 68\quad 72\quad 91}$

Example 2

0 1 2 3 4 5
5 1 6 2 4 3 i=0
swap

1 5 6 2 4 3

1 5 6 2 4 3

1 5 6 2 4 3 No swaping

1 5 6 2 4 3

i=1 0 1 2 3 4 5
1 5 6 2 4 3 (No swap)

1 5 6 2 4 3 swap

1 2 6 5 4 3 (No swap)

1 2 6 5 4 3 (No swap)

i = 2

0 1 2 3 4 5

1 2 6 5 4 3 (swap)

1 2 5 6 4 3 (swap)

1 2 4 6 5 3 (swap)

1 2 3 6 5 4

i = 3

0 1 2 3 4 5

1 2 3 6 5 4 (swap)

1 2 3 5 6 4 (swap)

1 2 3 4 6 5

i = 4

0 1 2 3 4 5

1 2 3 4 6 5 swap)

1 2 3 4 5 6 sorted.

Time complexity of Bubble sort

- The complexity of sorting algorithm is depends upon the no of comparisons that are made

- The total comparisons in bubble sort is $n(n-1)/2 \approx n^2 - n$.

Best case - $O(n^2)$

Worst case - $O(n^2)$

Average case - $O(n^2)$