1. Sorting – Arranging in ascending / descending or dn.

Bubble sort → adjacent elements swapping

[4 5 3 2 1] → Sort. (i, i+1)

pass 1: 4 5 3 2 1

j=0 arr[0]<arr[1] 4 5 3 2 1

j=1 arr[1]>arr[2] 4 3 5 2 1

j=2 arr(2)>arr(3) 4 3 2 5 1

j=3 arr(3)>arr(4) 4 3 2 1 5

unsorted | sorted

pass 2: 4 3 2 1 5

arr(0)>arr(1) 3 4 2 1 5 — j=0

arr(1)>arr(2) 3 2 4 1 5 — j=1

arr(2)>arr(3) 3 2 1 4 5 — j=2

unsorted | sorted

arr(3) arr(4)

i	pass	j+pass
3	1	4
2	2	4
1	3	4
0	4	4

pass 3: 3 2 1 4 5

j=0 — arr(0)>arr(1) 2 3 1 4 5

j=1 — arr(1)>arr(2) 2 1 3 4 5

unsorted | sorted

pass 4: 2 1 3 4 5

arr(0)>arr(1) 1 2 3 4 5 — j=0

sorted

j + pass = n-1

j = n-1-pass

Code – How many loop ?? 2 loops

① pass → 1 to n-1

② j = adjacent element swapping → 0 to (n-1-pass)

```
for(int pass=1; pass<=n-1;pass++){          n
    for(int j= 0;j<=n-1-pass;j++){          n
        if(arr[j]>arr[j+1]){
            //swapping
            int temp= arr[j];               } n×n
            arr[j]= arr[j+1];
            arr[j+1]= temp;
        }
    }
}
```

n×

$= O(n^2)$

SC → O(1)

Selection sort – Select minimum index.

[4 5 3 2 1]

M

Step 0 – initial_min_ind = 0, min_ind = 4

[1 | 5 3 2 4]

sorted | unsorted

n = 5

Step 1 – initial_min_ind = 1, min_ind = 3

[1 2 | 3 5 4]

sorted | unsorted

Step 2 – initial_min_ind = 2, min_ind = 2

[1 2 3 | 5 4]

sorted | unsorted

Step 3 – initial_min_ind = 3, min_ind = 4

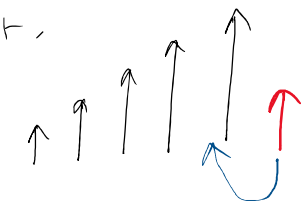
1 1 1 2 1 3 1 4 1 5

Sorted.

Loop?? 2 → i → 0 to n − 2

j = i + 1 to n − 1

Insertion Sort.



4 5 3 2 1

4 5 3 2 1 i=1.

assembly line

4 5

Sorted

4 5 3 → 4 3 5 → 3 4 5

Sorted

Sorted

3 4 5 2 → 3 4 2 5 → 3 2 4 5 → 2 3 4 5

Sorted

2 3 4 5 1 → 2 3 4 1 5 → 2 3 1 4 5 → 2 1 3 4 5

↓

1 2 3 4 5

Sorted.