Space Complexity : auxiliary space required.

↙ extra space apart from given.

SC → O(1)

new array n size → O(n)

In-place → no extra space O(1)

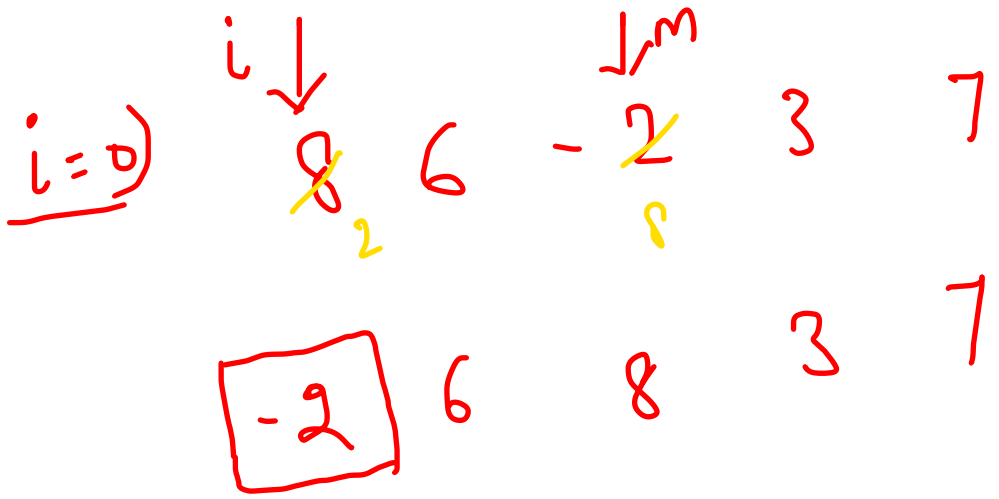
SC of bubble sort → O(1)

Selection Sort: pick the smallest element & place it in the start. i will start from 0

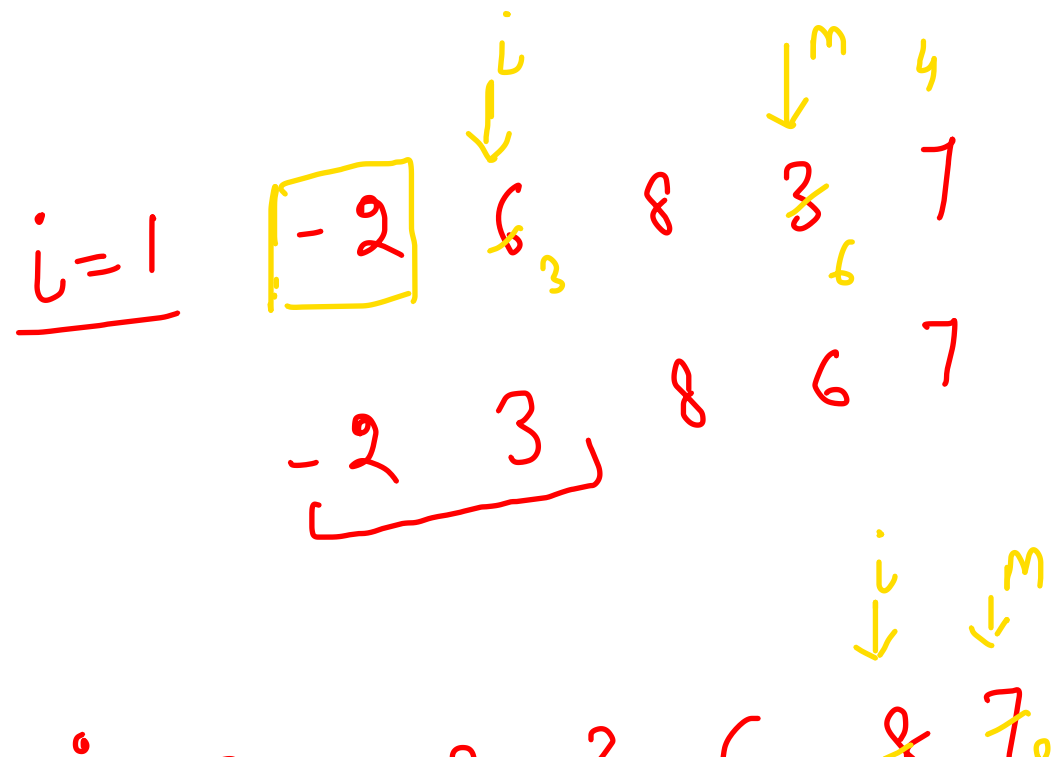
arr =

0	1	2	3	4
8	6	-2	3	7

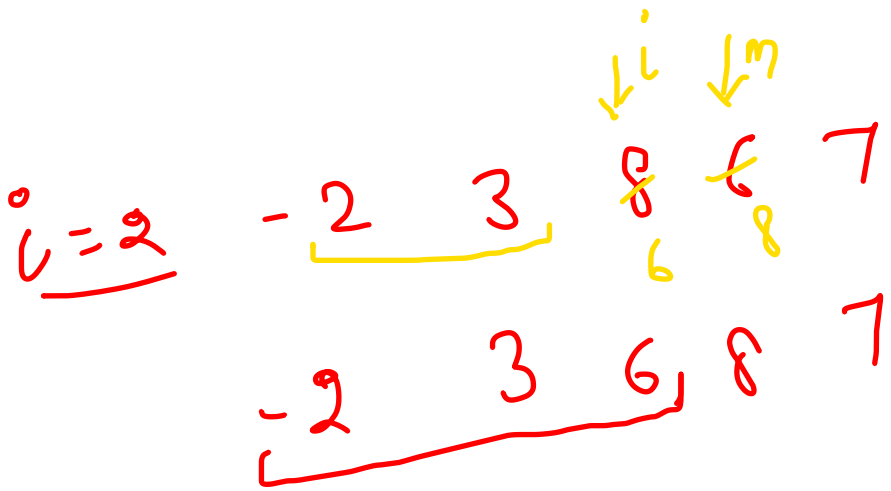
i=0



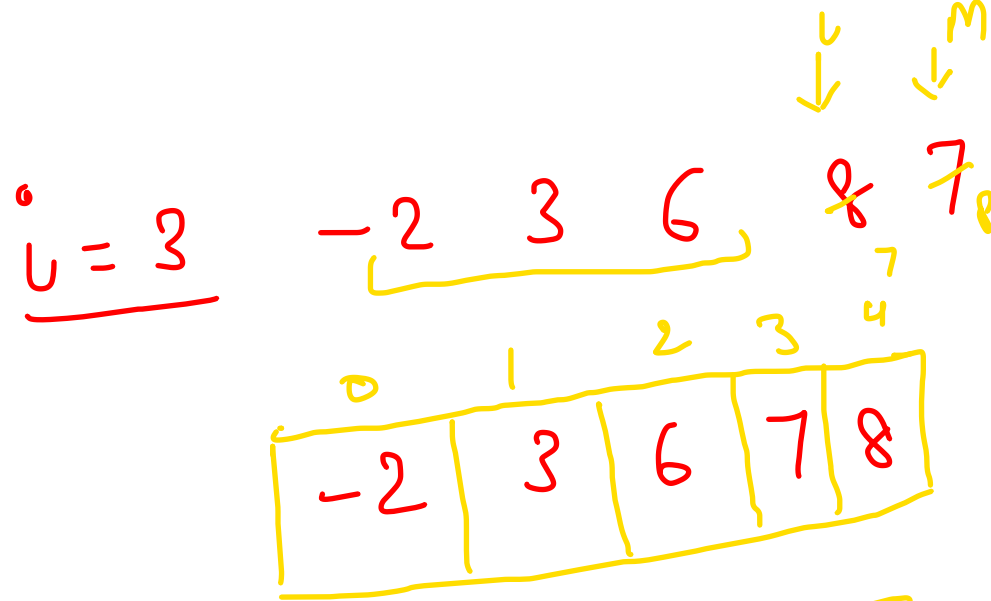
i=1



i=2



i=3



[n-1 sounds sorted]

pseudo code.

1) traverse from 0 to n-1
   int min = i
2) traverse from i+1 to n
3) if (arr[j] < arr[min])
   min = j
4) swap(i, min)

TC.

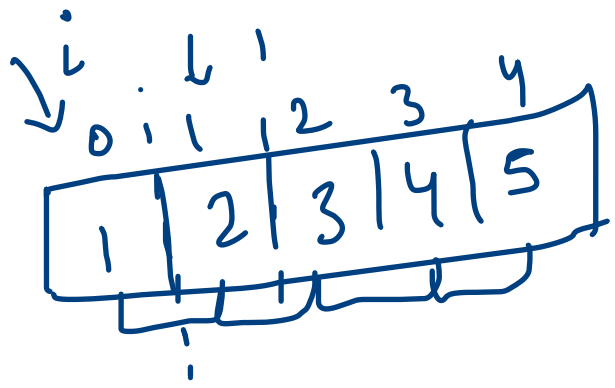
Best → $O(n^2)$

Avg. → $O(n^2)$

Worst → $O(n^2)$

SC. → $O(1)$

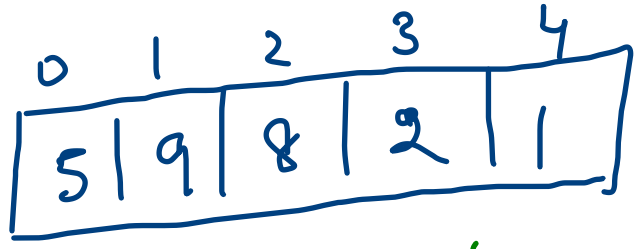
→



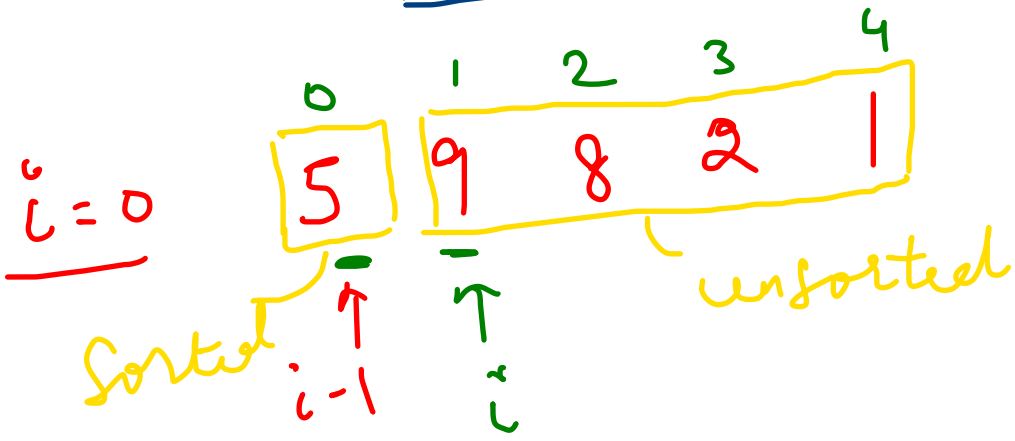
→ n^2 ✓ ✗

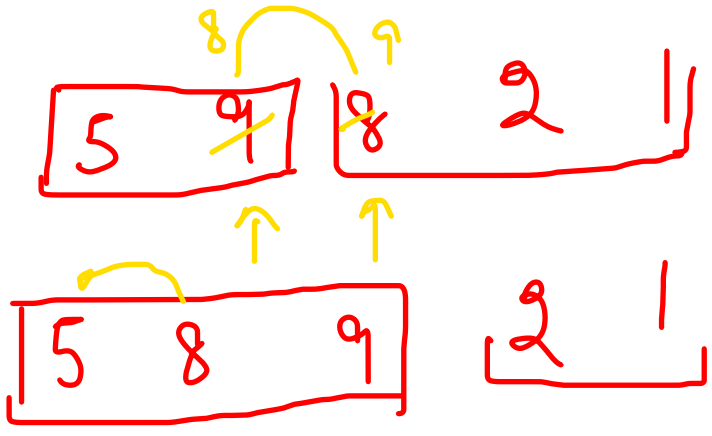
Insertion Sort – Assume that first element is already sorted and others are not. (shifting)

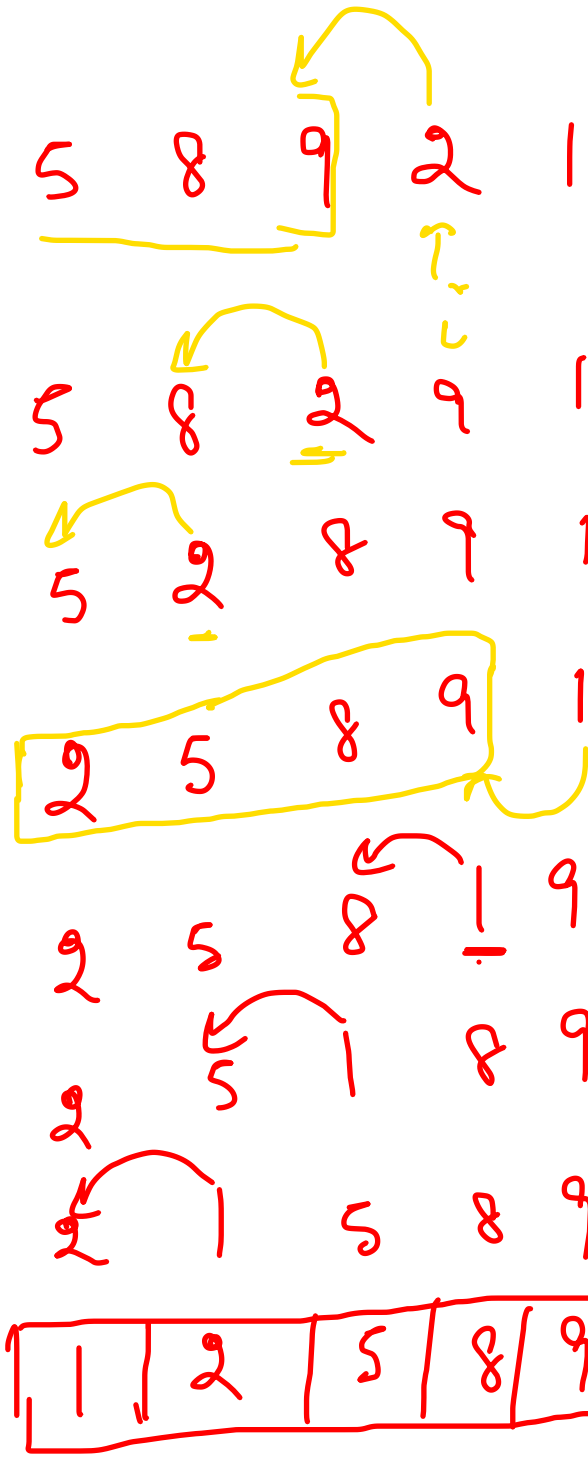
arr =



i = 0



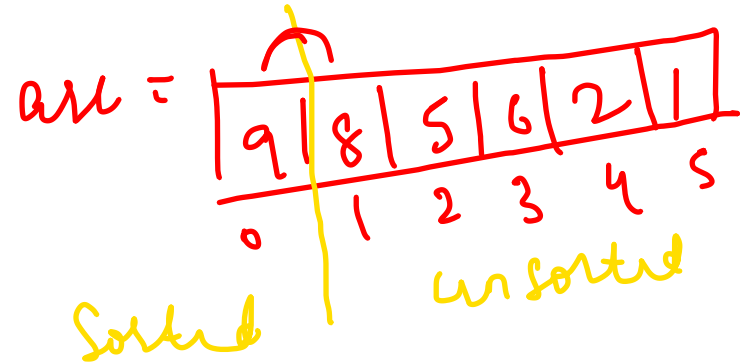


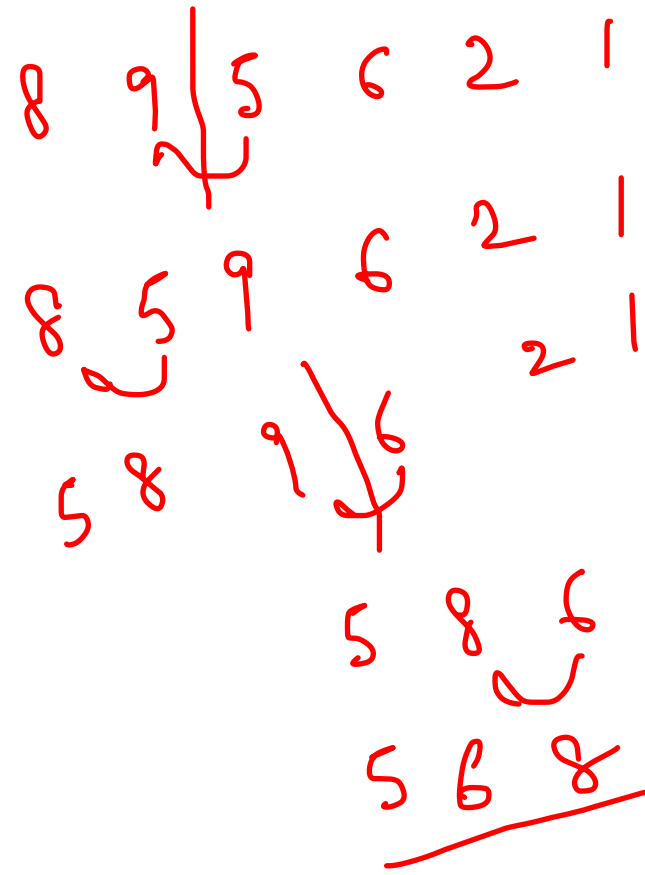


Pseudo - code.

```
for(int i=1; i<n; i++){
    for(int j = i; j>0; j--){
        if (arr[j-1] > arr[j]){
            swap(j, j-1)
        }
        else {
            break;
        }
    }
}
```

```java
public static void sort(int arr[], int n){
    for(int i = 1; i < n; i++){
        for(int j = i; j > 0; j--){
            if(arr[j-1] > arr[j]){
                swap(j, j-1);
            }
            else{
                break;
            }
        }
    }
    // print
```





i = 1 < 6

j = 1 > 0

arr[j-1] = 9 > 8 T

i = 2 < 6

j = 2 > 0

arr[j-1] = 9 > 5 T

j = 1 > 0

arr[j-1] = 8 > 5 T

i = 3 < 6

j = 3 > 0

arr[j-1]

9 > 6 T

j = 2 > 0

arr[j-1] = 8 > 6 T

j = 1 > 0

arr[j-1]

5 > 6 F

n-1 iterations

SC. → O(1)

TC.

Best - O(n)

Avg - O(n²)

Worst - O(n²)

	TC			SP
	B	A	W	
Bubble	O(n)	O(n²)	O(n²)	O(1)
Selection	O(n²)	O(n²)	O(n²)	O(1)
Insertion	O(n)	O(n²)	O(n²)	O(1)