Sorting —

Loop — → pass

n = 5

pass 1 :- 4 5 3 2 1

arr[0] < arr[1] 4 5 3 2 1 → j = 0

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

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

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

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

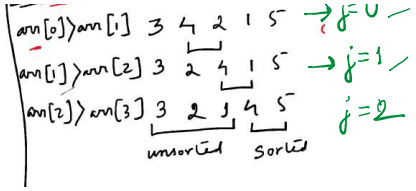
loop ?? = 2

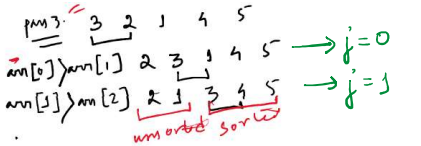
1 st loop → pass → 1 to n-1 (0 to n-2)

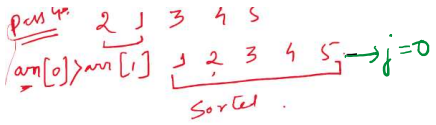
2nd loop → swapping 0 to n-1-pass

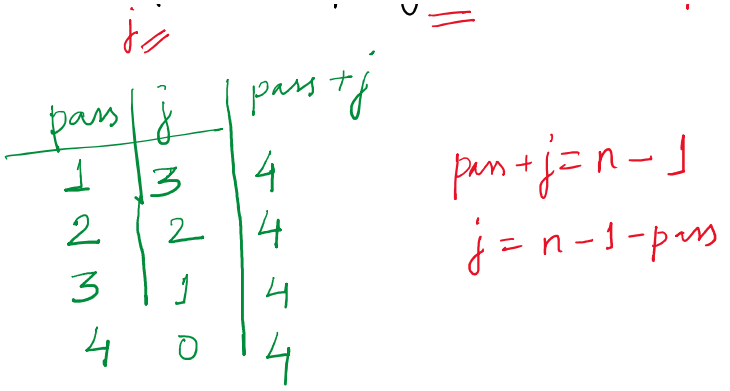
j =

1 pass + j





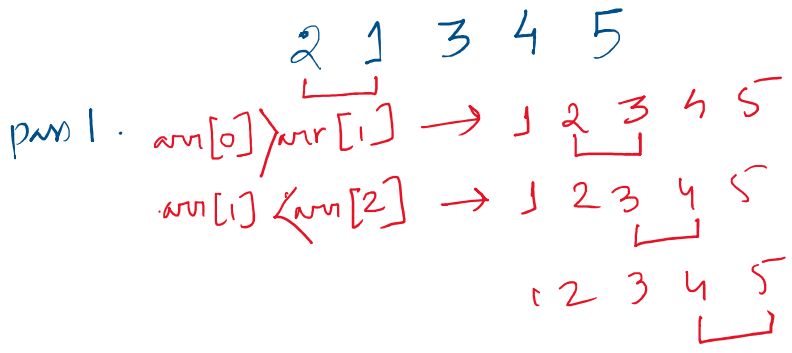


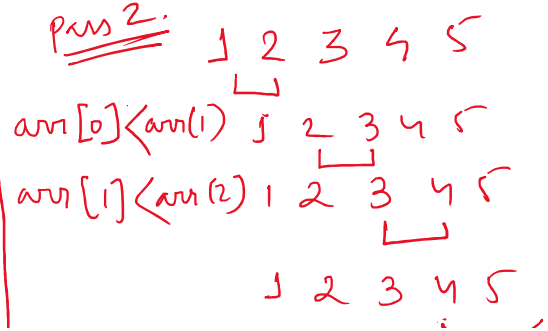



```
static void bubbleSort(int arr[], int n){
        for(int pass=1; pass<=n-1; pass++){
            for(int j= 0; j<= n-1-pass; j++){
                if(arr[j]>arr[j+1]){
                    //swapping
                    int temp= arr[j];
                    arr[j]= arr[j+1];
                    arr[j+1]=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

n
n
n x n

TC = ?? = O(n²)
SC = ?? O(1)





1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

Selection sort — select minimum.

4 5 3 2 1

i=0, j=1 2 3 4

min_ind = 0 2 3 4

Swapping - arr(i=0), arr(min_ind=4)

⇒ 1 5 3 2 4

sorted | unsorted

i=1, j=2 3 4

min_ind = 1 2 3

Swap (arr(i=1), arr(min_ind=3))

4 5 3 2 1

1 2 | 3 5 4

sorted | unsorted

i=2, j=3 4 5

min_ind = 2

Swap (arr(i=2), arr(min_ind=2))

1 2 3 | 5 4

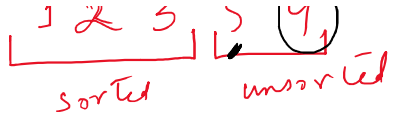
sorted | unsorted

Loop

2 →

i = 0 to n-2

j = i+1 to n-1



$i=3,\ j=4 \le 5$ ✓

min_ind = ~~3~~ 4

swap(arr(i=3), arr(min_ind=4))

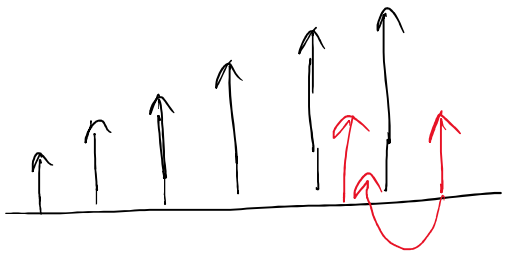
1 2 3 4 5

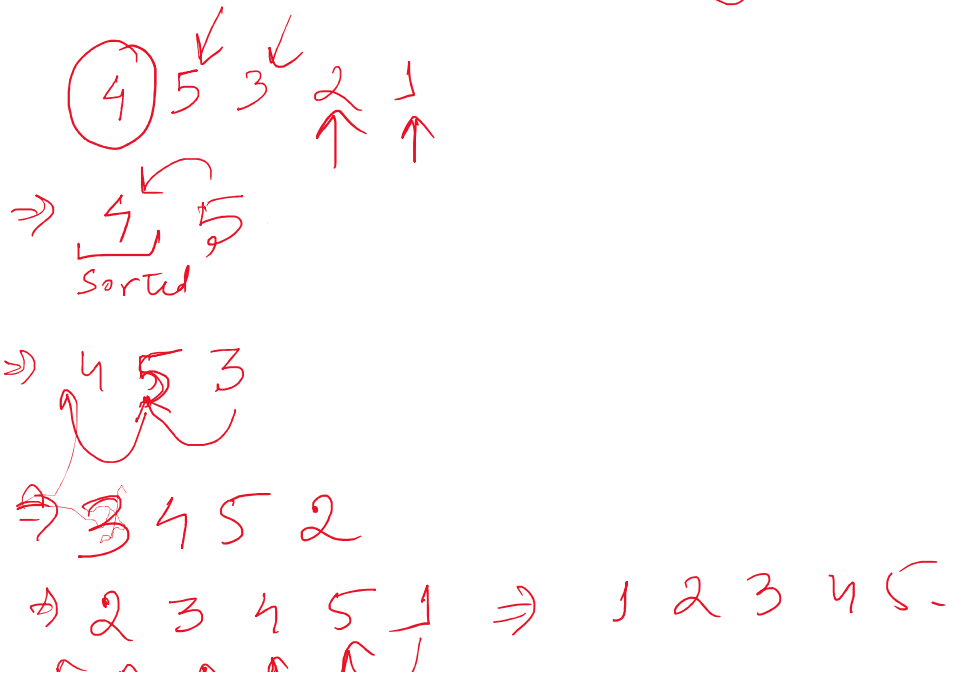
Sort.

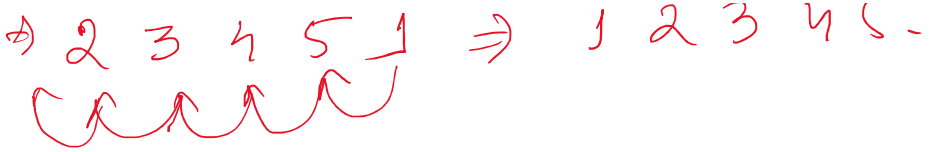
j = i+1 to n−1

min_ind = i

Insertion sort −






2 3 4 5 1
1 2 3 4 5