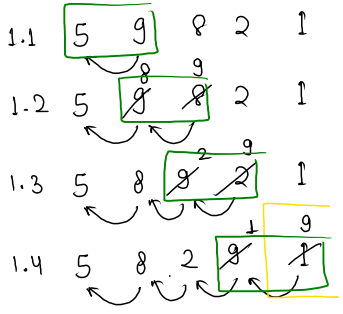
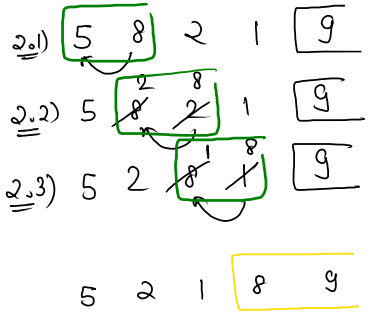
⇒ Bubble sort // trying to take largest element of array to the right

e.g., 5 9 8 2 1

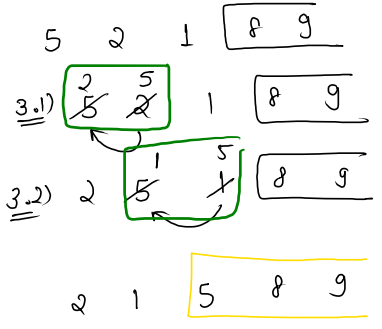
① i=0



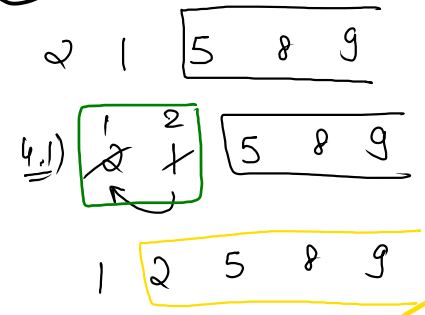
② i=1



③ i=2



④ i=3



count
i=0 → 4
i=1 → 3
i=2 → 2
i=3 → 1

pseudo code

compare consiqutive elements
and check if right one is smaller
then swap elements

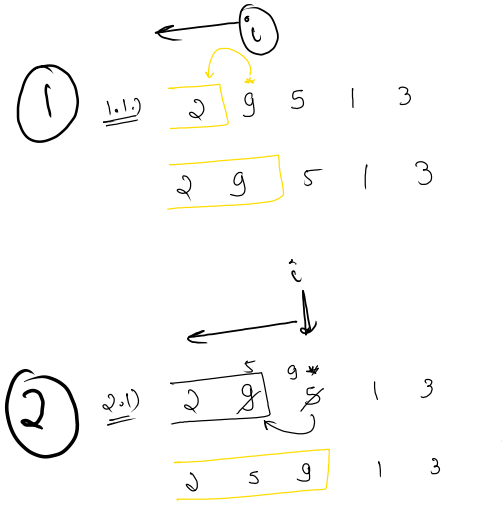
itr i=1, j = 0, 1, 2, 3 ✓ ④
itr i=2, j = 0, 1, 2 ✓ ③
itr=3, j = 0, 1 ✓ ②
itr=4, j = 0 ✓ ①

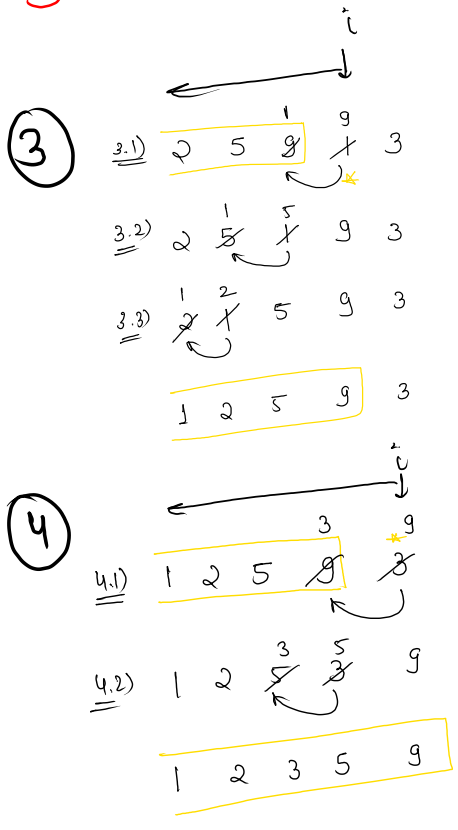
//bubble sort code

```java
public static void main(String[] args) {
    int[] arr = { 5, 9, 8, 2, 1 };
    int n = arr.length;
    for (int itr = 1; itr < arr.length; itr++) {
        for (int j = 0; j < arr.length - itr; j++) {
            if ( arr[j] > arr[j + 1] ) {
                swap(arr, j, j + 1);
            }
        }
    }

    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}

public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

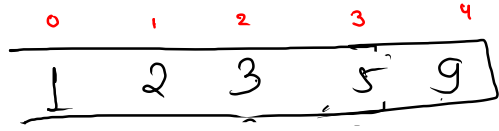
⇒ Insertion sort // pick an element and insert it in the correct position of sorted array

e.g., 2 9 5 1 3






```java
public static void main(String[] args) {
    int[] arr = { 2, 9, 5, 1, 3 };
    int n = arr.length;
    // main logic
    for (int i = 1; i < arr.length; i++) {
        for (int j = i - 1; j >= 0; j--) {
            if ( arr[j] > arr[j + 1] ) {
                swap(arr, j, j + 1);
            } else {
                break;
            }
        }
    }

    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}

public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```



i=1; j=0 (2 > 9) break

i=2; j=1 (9 > 5)
j=0 (2 > 5) break

i=3, j=2 (9 > 1)
j=1 (5 > 1)
j=0 (2 > 1)

i=4, j=3 (9 > 3)
j=2 (5 > 3)
j=1 (2 > 3) break

→ kth largest element

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int k = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    kthLargest(arr, n, k);
}

public static void kthLargest(int[] arr, int n, int k) {
    // insertion sorting
    for (int i = 1; i < arr.length; i++) {
        for (int j = i - 1; j >= 0; j--) {
            if ( arr[j] > arr[j + 1] ) {
                swap(arr, j, j + 1);
            } else {
                break;
            }
        }
    }

    int ans = arr[n - k];
    System.out.println(ans);
}

public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

Steps

1) Sort the array
2) Print k^{th} element from last