Rotate Right

n = 7

arr = [1, 2, 3, 4, 5, 6, 7]

K = 3

rotated array [5 6 7 1 2 3 4]

approach1

TLE

n = 7

$$arr = [\overset{7}{\cancel{1}}, \underset{\overset{\uparrow}{i}}{\overset{1}{\cancel{2}}}, \overset{2}{\cancel{3}}, \overset{3}{\cancel{4}}, \overset{4}{\cancel{5}}, \overset{5}{\cancel{6}}, \overset{6}{\cancel{7}}]$$

K = 1

temp = 7

ans = [7 1 2 3 4 5 6]

$arr = [1, 2, 3, 4, 5, 6, 7]$ $\underline{K = 3}$

trick

$\underline{n = 7}$

Step1 → reverse K elements from last

$arr = [\overset{0}{1}, \overset{1}{2}, \overset{2}{3}, \overset{3}{4}, \underline{\overset{4}{7}, \overset{5}{6}, \overset{6}{5}}]$ $\underline{(n-k, n-1)}$

Step2 → reverse remaining element

$arr = [\underline{4, 3, 2, 1}, 7, 6, 5]$ $\underline{(0, n-k-1)}$

Step3 → reverse entire array

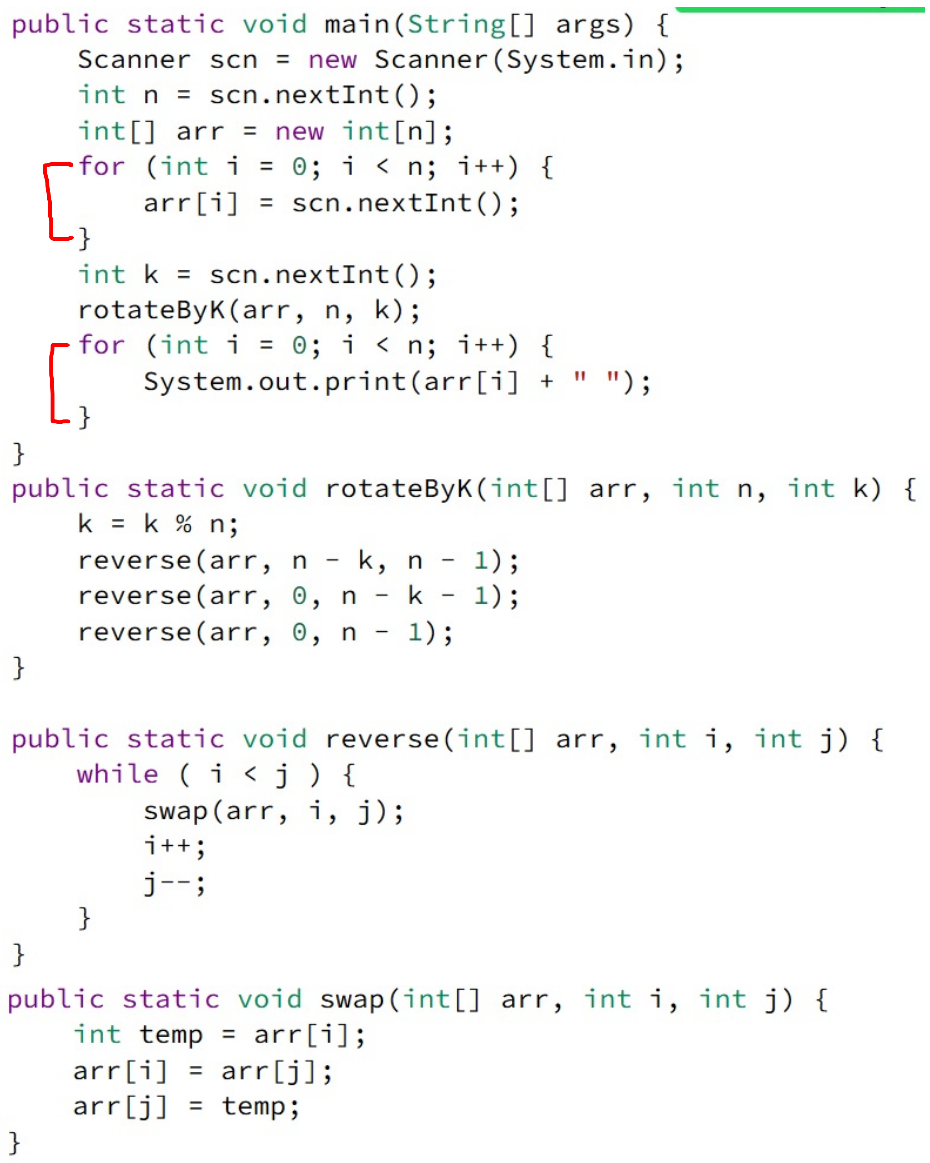
$arr = [\underline{5, 6, 7, 1, 2, 3, 4}]$ $(0, n-1)$

observation

K = 20

[1, 2, 3, 4, 5]	K = 0, 5, 10, 15, 20
[5, 1, 2, 3, 4]	K = 1, 6, 11, 16
[4, 5, 1, 2, 3]	K = 2, 7, 12, 17
[3, 4, 5, 1, 2]	K = 3, 8, 13, 18
[2, 3, 4, 5, 1]	K = 4, 9, 14, 19

Code



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int k = scn.nextInt();
    rotateByK(arr, n, k);
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}
public static void rotateByK(int[] arr, int n, int k) {
    k = k % n;
    reverse(arr, n - k, n - 1);
    reverse(arr, 0, n - k - 1);
    reverse(arr, 0, n - 1);
}

public static void reverse(int[] arr, int i, int j) {
    while ( i < j ) {
        swap(arr, i, j);
        i++;
        j--;
    }
}
public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

Zeroes and Ones

arr = [0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0]

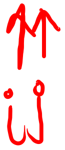
→ [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]

Note:- 1) Arrays.sort(arr) O(Nlog(N))

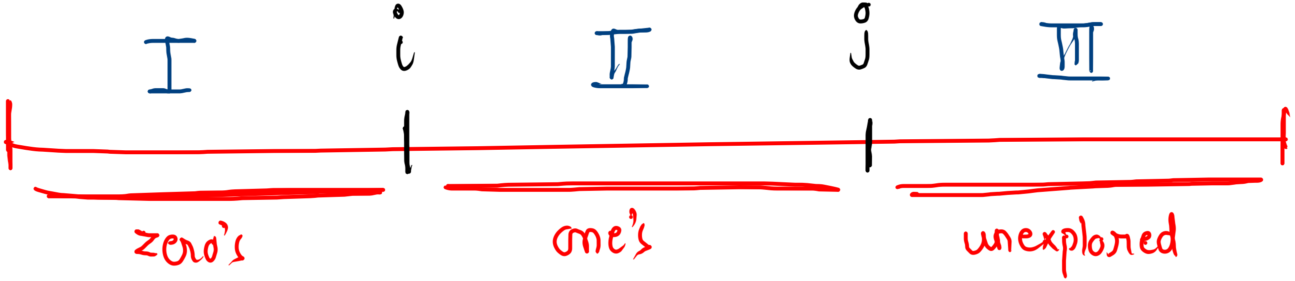
but we have to solve it in O(N)

2) You can traverse in array only once.

arr = [0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0]



faith



assumption

1) from 0 to i ⟶ always zero's
2) from i to j ⟶ always one's
3) from j to end ⟶ unexplored

arr = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]

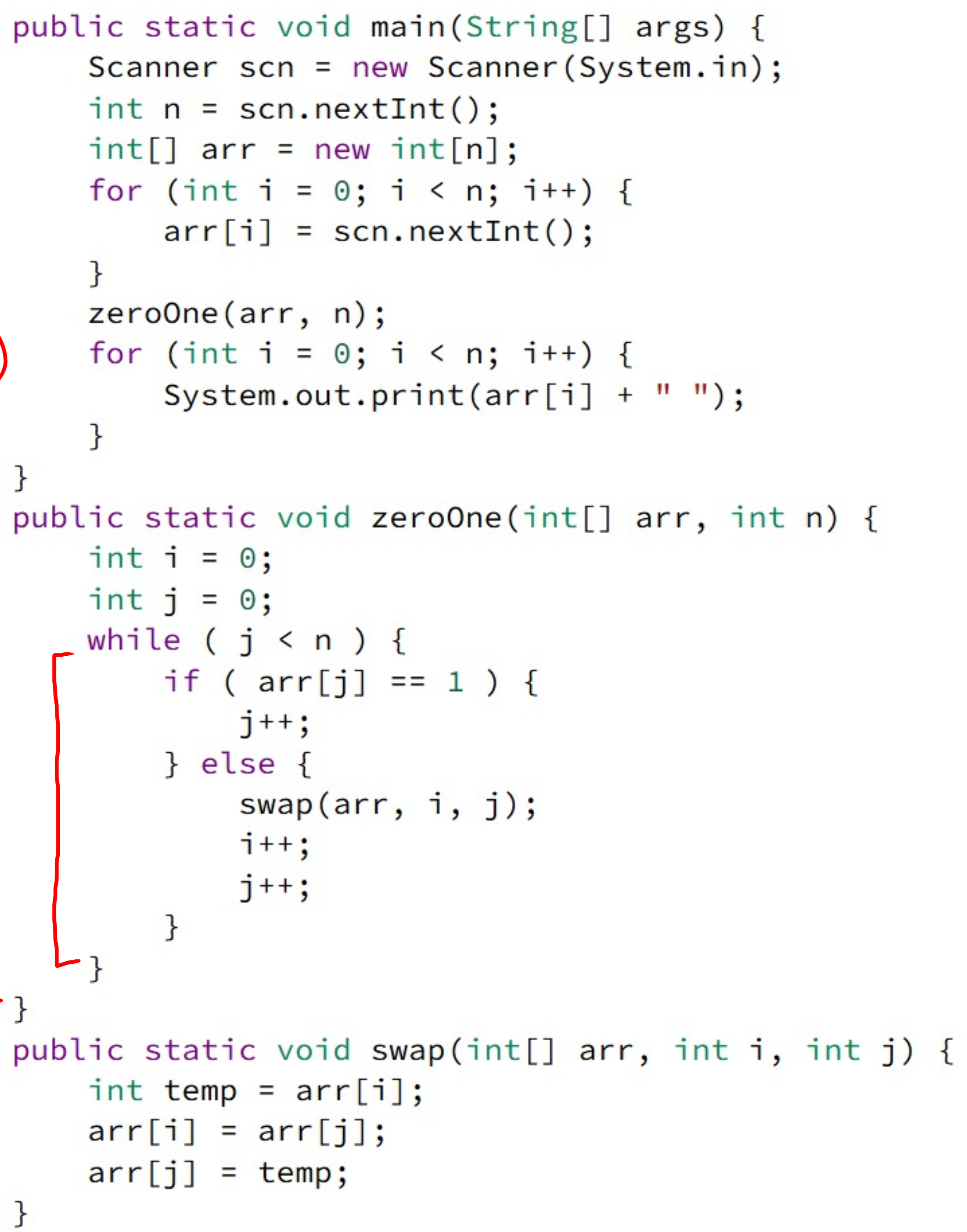
↑ i ↑ j

O(N)

```
if ( arr[j] == 0 ) {
        swap (i, j) ;
        i++, j++;
} else {
        j++;
}
```

Code

T.C = O(N)



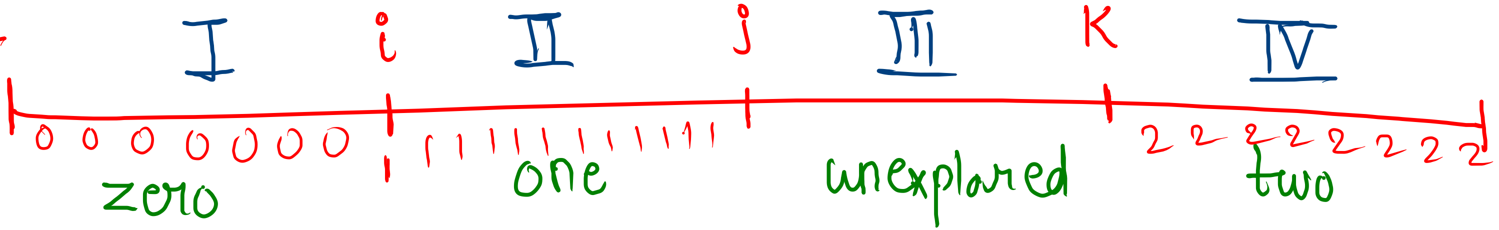
```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    zeroOne(arr, n);
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}
public static void zeroOne(int[] arr, int n) {
    int i = 0;
    int j = 0;
    while ( j < n ) {
        if ( arr[j] == 1 ) {
            j++;
        } else {
            swap(arr, i, j);
            i++;
            j++;
        }
    }
}
public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

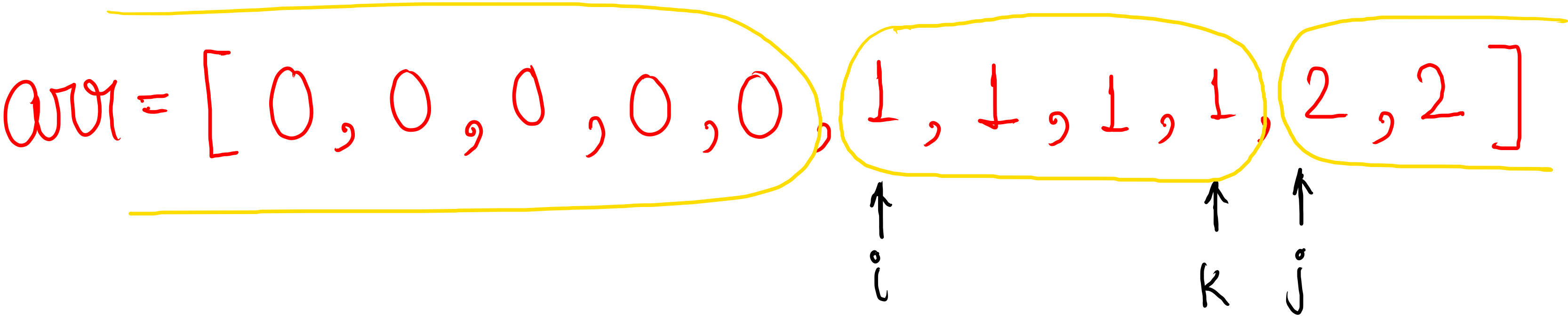
Ques Sort 0, 1, 2

arr = [0, 1, 0, 0, 1, 1, 2, 0, 1, 2, 0]

↳ [0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2]

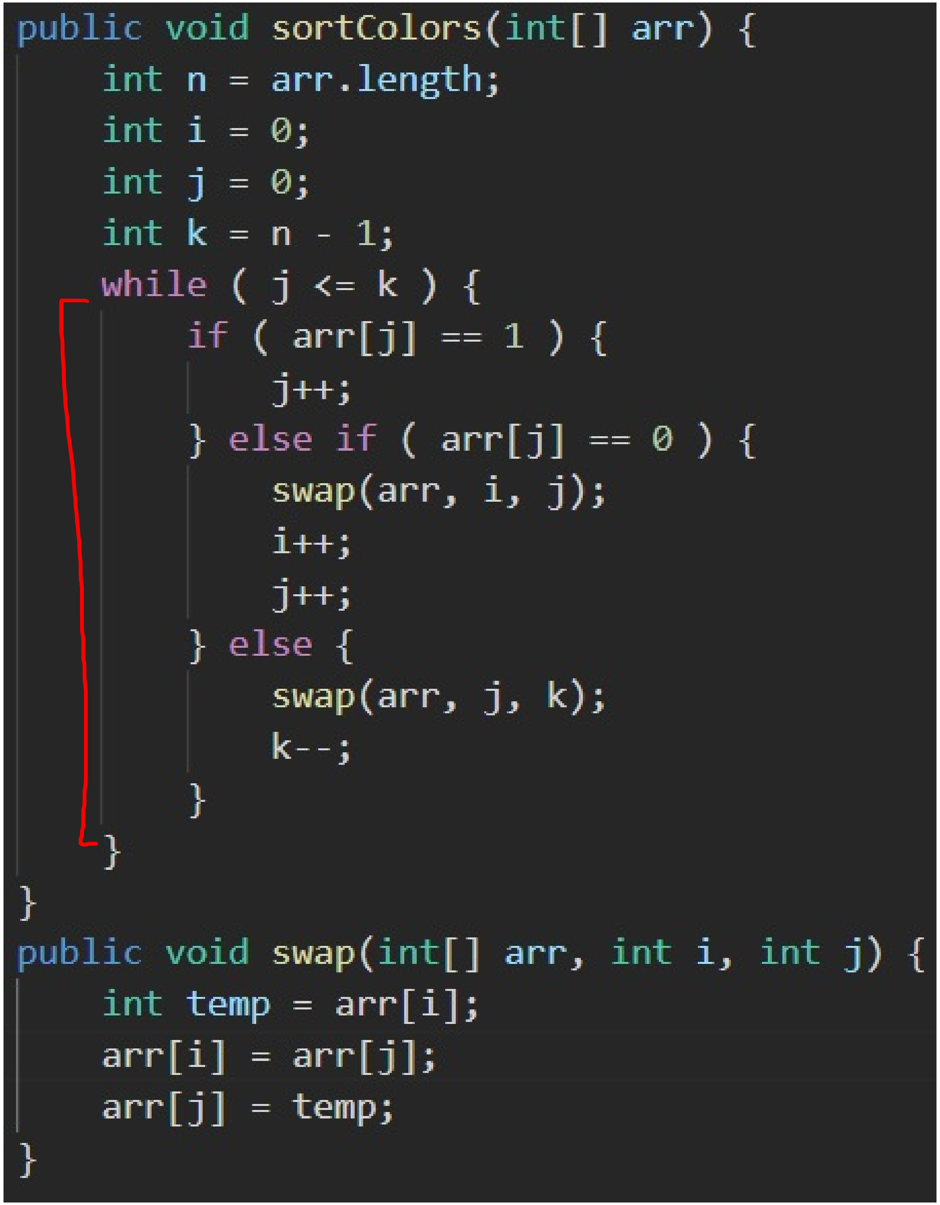
faith





```
  if ( arr[j] == 1 ){
a         j++;
  } else if (arr[j] == 0) {
          swap(i,j);
b         i++, j++;
  } else {
          swap(j,k);
c         k--;
  }
```

Code



```
public void sortColors(int[] arr) {
    int n = arr.length;
    int i = 0;
    int j = 0;
    int k = n - 1;
    while ( j <= k ) {
        if ( arr[j] == 1 ) {
            j++;
        } else if ( arr[j] == 0 ) {
            swap(arr, i, j);
            i++;
            j++;
        } else {
            swap(arr, j, k);
            k--;
        }
    }
}
public void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

T.C = O(N)