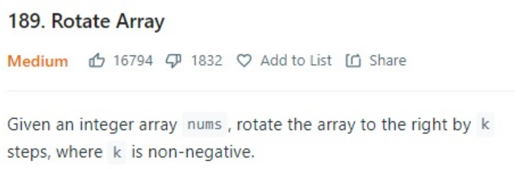
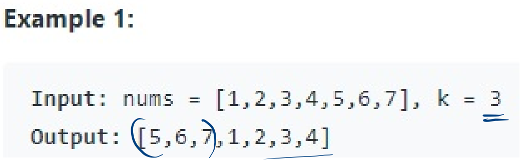


189. Rotate Array

Medium 👍 16794 👎 1832 ♡ Add to List ⎘ Share

Given an integer array `nums`, rotate the array to the right by `k` steps, where `k` is non-negative.

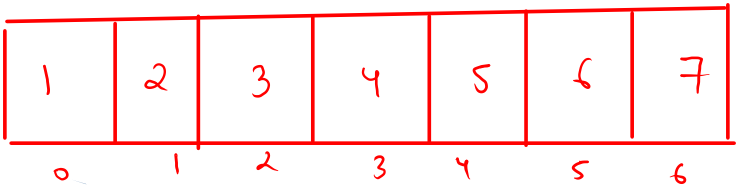


Example 1:

```
Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
```

K=3.

k=0



k=1	7 1 2 3 4 5 6
k=2	6 7 1 2 3 4 5
k=3	5 6 7 1 2 3 4
k=4	4 5 6 7 1 2 3
k=5	3 4 5 6 7 1 2
k=6	2 3 4 5 6 7 1

k=7 1 2 3 4 5 6 7

k=

1

K=8

k = 0 <7

k=8 → k=k%n

***

n unique rotations.

K=K% n

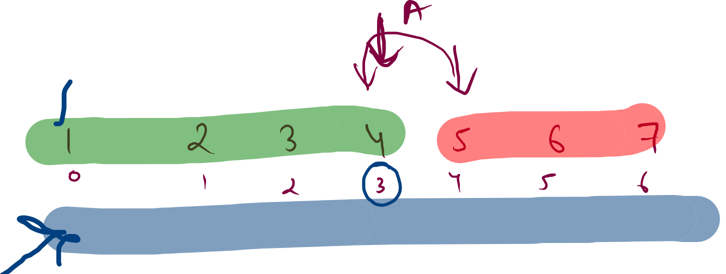
K=3.

n = 7

k = 3

n−k

7 − 3 − 1



green

red

blue

4 3 2 1 7 6 5

5 6 7 1 2 3 4

ans.

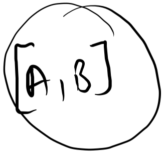
✓ reverse (0, n-k-1)

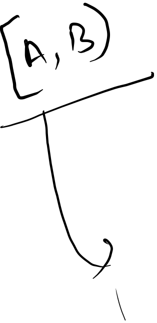
reverse(n-k, n-1)

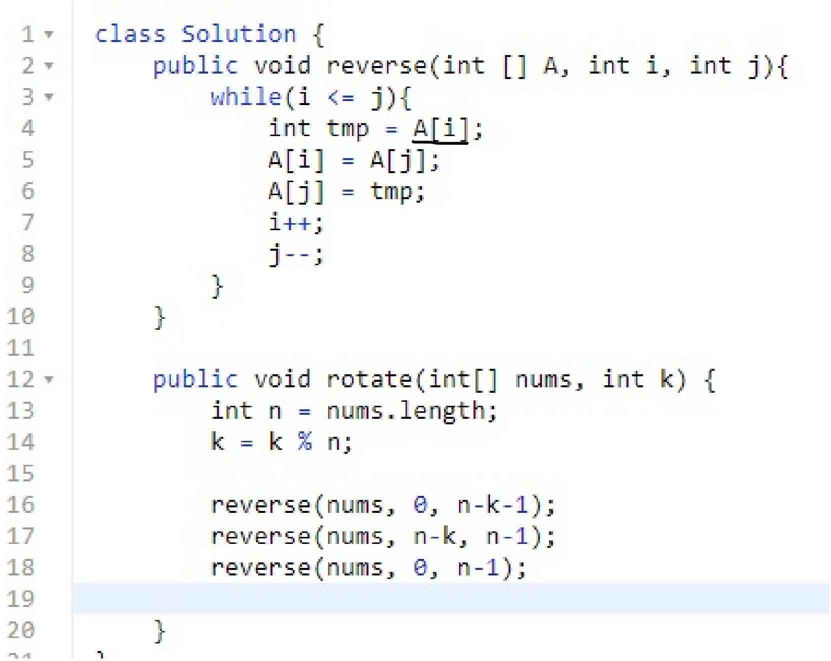
reverse (0, n-1)

n-k → ?

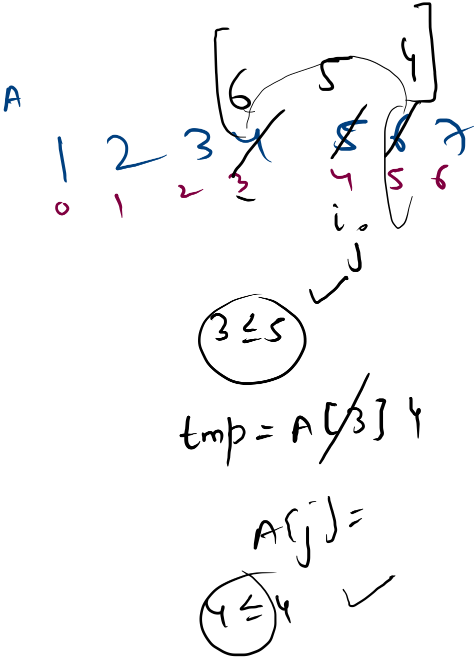
✓ n-k-1 → ?



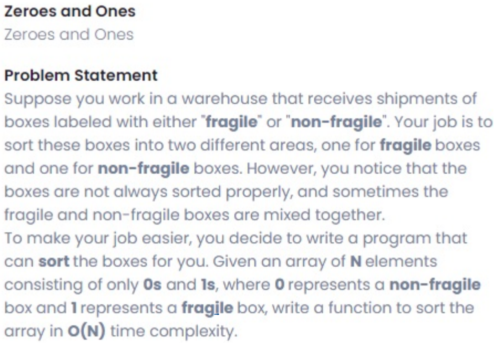




```
class Solution {
    public void reverse(int [] A, int i, int j){
        while(i <= j){
            int tmp = A[i];
            A[i] = A[j];
            A[j] = tmp;
            i++;
            j--;
        }
    }

    public void rotate(int[] nums, int k) {
        int n = nums.length;
        k = k % n;

        reverse(nums, 0, n-k-1);
        reverse(nums, n-k, n-1);
        reverse(nums, 0, n-1);

    }
}
```



Arrays.sort → O(n log n).



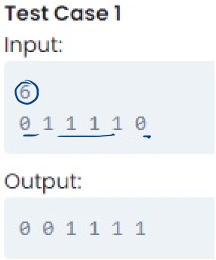
Zeroes and Ones

Zeroes and Ones

Problem Statement

Suppose you work in a warehouse that receives shipments of boxes labeled with either "**fragile**" or "**non-fragile**". Your job is to sort these boxes into two different areas, one for **fragile** boxes and one for **non-fragile** boxes. However, you notice that the boxes are not always sorted properly, and sometimes the fragile and non-fragile boxes are mixed together.

To make your job easier, you decide to write a program that can **sort** the boxes for you. Given an array of **N** elements consisting of only **0s** and **1s**, where **0** represents a **non-fragile** box and **1** represents a **fragile** box, write a function to sort the array in **O(N)** time complexity.



Test Case 1

Input:

```
6
0 1 1 1 1 0
```

Output:

```
0 0 1 1 1 1
```

0 1 1 1 1 0

↓

0 0 1 1 1 1

1. sort } inbuilt } bubble

2. Counting. ⇝ O(n).

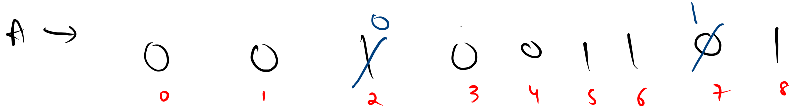
3. swap. ?

count zero = 2.

count one = n − count zero

= 4

00000 1111



i

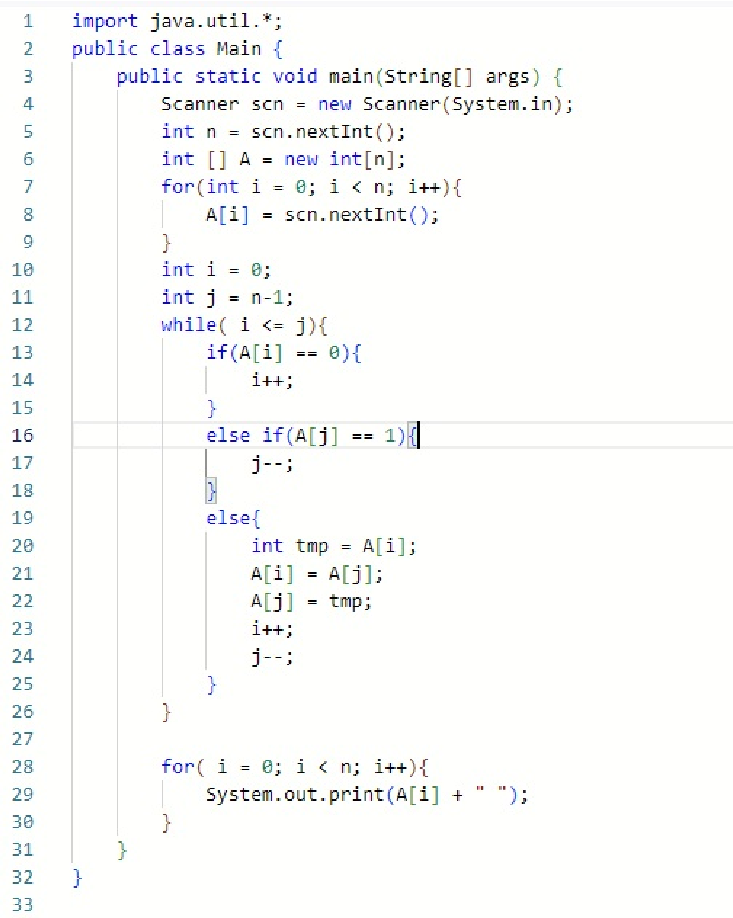
j

sort by Parity

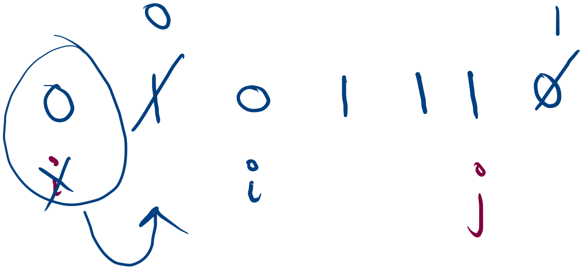
else
{
swap.
}

if (A[i] == 0)
{
i++
}

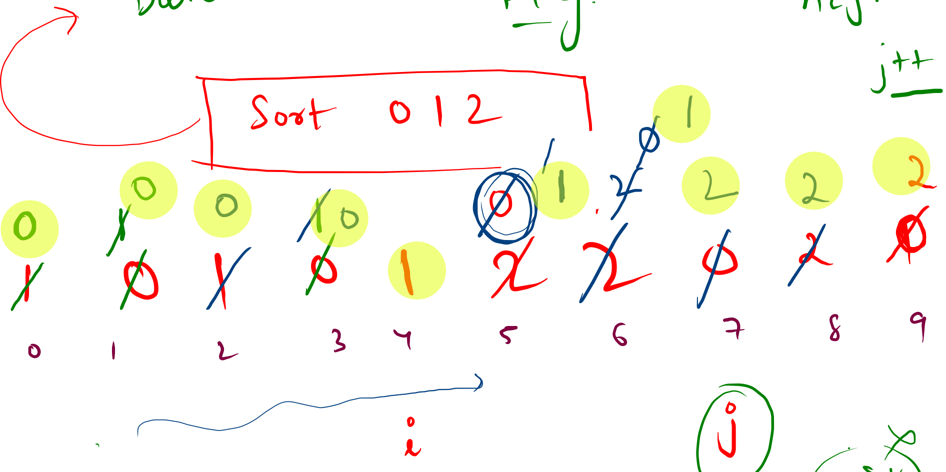
else if (A[j] == 1)
{
}



```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int i = 0;
        int j = n-1;
        while( i <= j){
            if(A[i] == 0){
                i++;
            }
            else if(A[j] == 1){
                j--;
            }
            else{
                int tmp = A[i];
                A[i] = A[j];
                A[j] = tmp;
                i++;
                j--;
            }
        }

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



Dutch National Flag.



A[j]==1

j++

0 to i-1 → 0

i to j-1 → 1

k+1 to n-1 → 2

j to k → unknown

j

~~j>k~~

j≤k ✓

A[j]==2

swap(j,k)

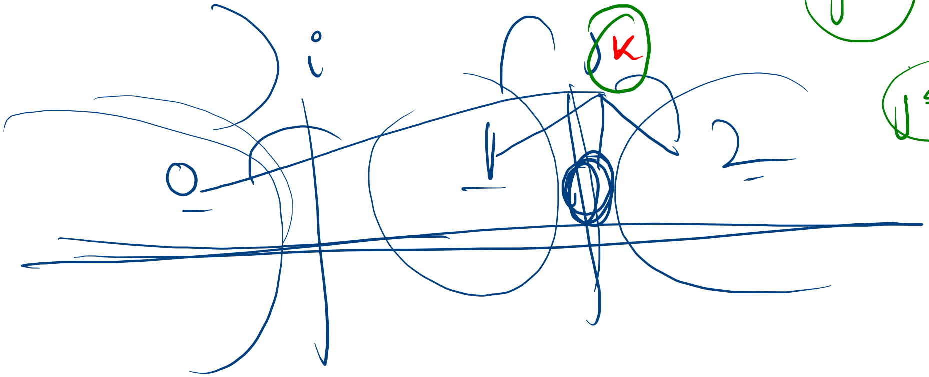
k--

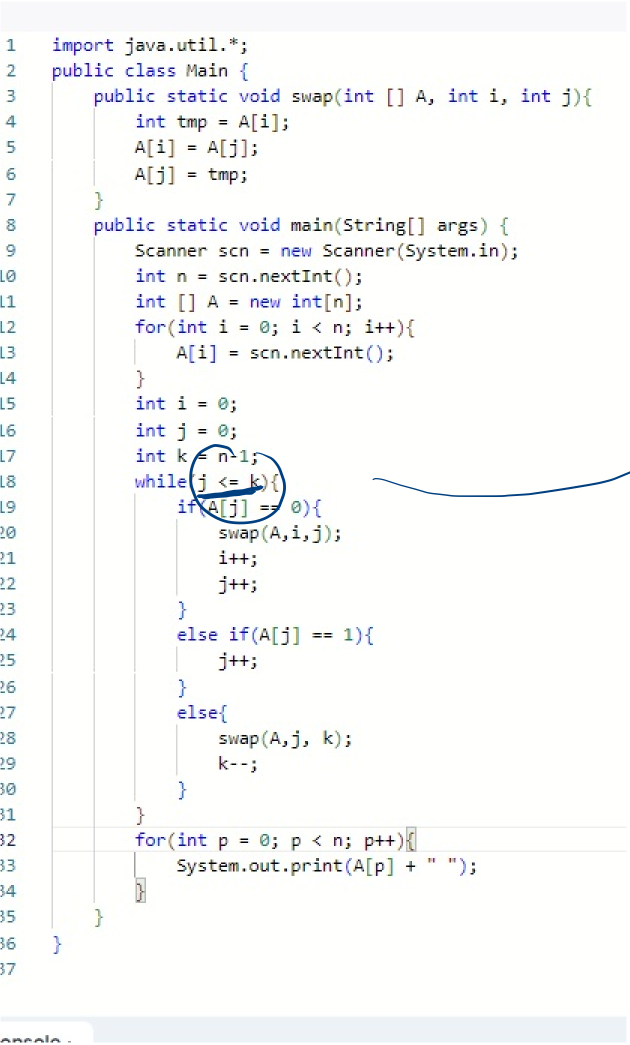
A[j]==0

Swap(i,j)

i++

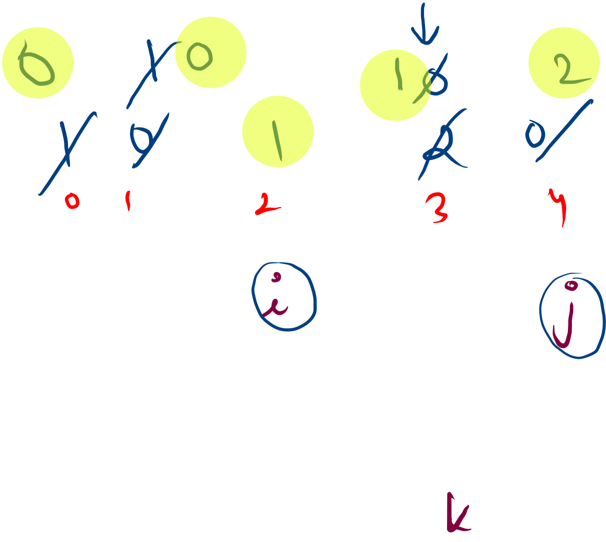
j++





```java
import java.util.*;
public class Main {
    public static void swap(int [] A, int i, int j){
        int tmp = A[i];
        A[i] = A[j];
        A[j] = tmp;
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int i = 0;
        int j = 0;
        int k = n-1;
        while(j <= k){
            if(A[j] == 0){
                swap(A,i,j);
                i++;
                j++;
            }
            else if(A[j] == 1){
                j++;
            }
            else{
                swap(A,j, k);
                k--;
            }
        }
        for(int p = 0; p < n; p++){
            System.out.print(A[p] + " ");
        }
    }
}
```

$j<k$?



$j \le k$

$4 \le 3$ ✗