

Previous

↳ 1. Max. Sum Subarray

4 -12 4 1 -2 3  
                    ↘  
                    (5)

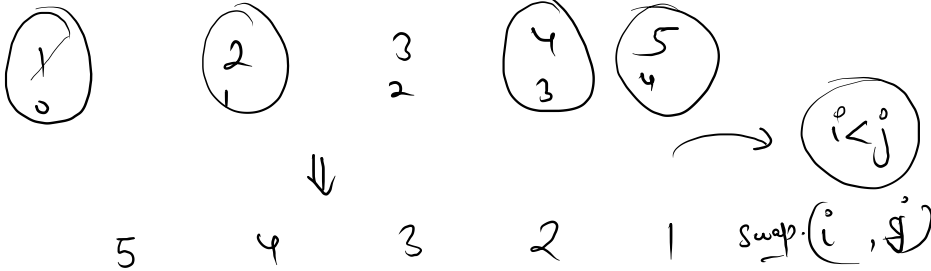
$csum < 0$

↳ start new sub

$csum \geq 0$

↳ contribute to previous sub.

2. Max. Product Subarray,  
3. reverse an array,



$\left. \begin{array}{l} i \geq 0 \\ j = n-1 \end{array} \right\}$

Rotate Right

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

Sample Input 0



1 2 3 4 5 6 7  
0 1 2 3 4 5 6

$k=2$

Sample Output 0

6 7 1 2 3 4 5

↓  
6 7 1 2 3 4 5

1 2 3 4 5 6 7

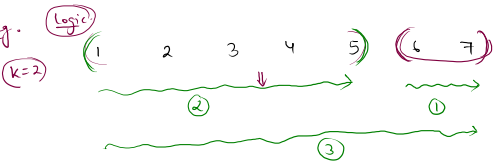
$k=3$

↓  
5 6 7 1 2 3 4

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

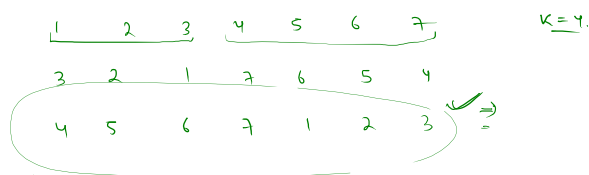
↓  
→ 4 5 6 7 1 2 3

$k \geq 7$   
 $k < 7$

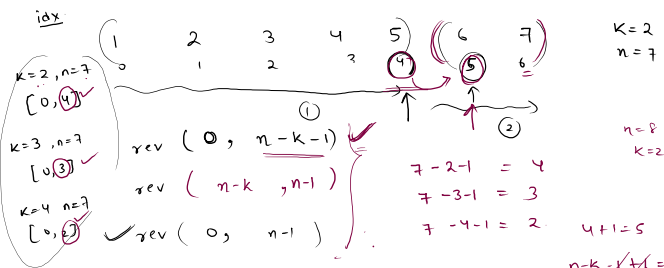
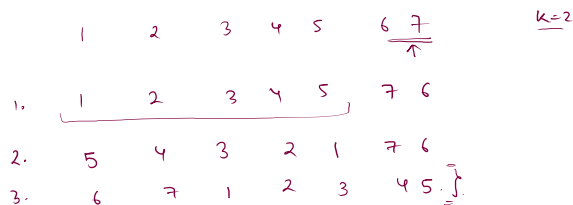


(1 2 3 4 5) (6 7)  
5 4 3 2 1 7 6  
↓  
{ 6 7 1 2 3 4 5 }

(1 2 3 4 5) (6 7)  $k=2$   
5 4 3 2 1 7 6  
6 7 1 2 3 4 5



logic.

$$\left\{ \begin{array}{l} \text{reverse}(\frac{?}{?}, \frac{?}{?}) \\ \text{reverse}(\frac{?}{?}, \frac{?}{?}) \\ \text{reverse}(0, n-1) \end{array} \right\}$$


```

import java.io.*;
import java.util.*;

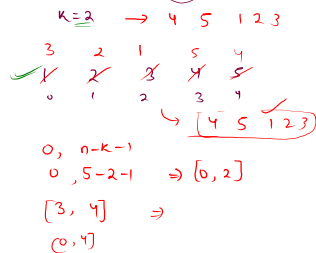
public class Solution {
    public static void reverse(int [] arr, int i, int j){
        while(i < j){
            int t = arr[i];
            arr[i] = arr[j];
            arr[j] = t;
            i++;
            j--;
        }
    }

    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();

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

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

```



# Zeros and Ones

Given an array having n elements consisting of only 0s and 1s. Sort the array in  $O(n)$  time complexity.

Sample Input 0

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

Sample Output 0

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

Sort.

Array sort  $\rightarrow O(n \log n)$

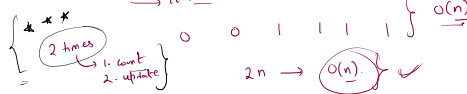
$\left. \begin{matrix} B \\ I \\ S \end{matrix} \right\} \rightarrow O(n^2)$

Logic:

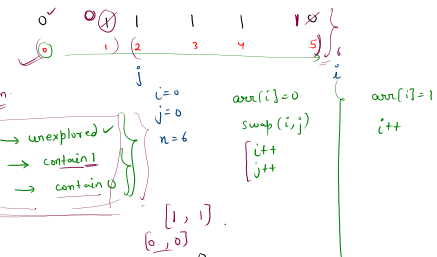


$z = 2$   
 $0 = 4$

iterate



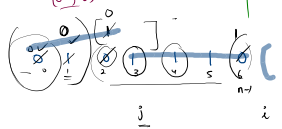
2 pointer



region:

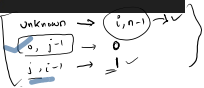
1.  $i, n-1 \rightarrow$  unexplored
2.  $j, i-1 \rightarrow$  contain 1
3.  $0, j-1 \rightarrow$  contain 0

$[1, 1]$   
 $[0, 0]$



```
int i = 0;
int j = 5;
while(i < j){
    if(arr[i] == 0){
        swap(arr, i, j);
        i++;
        j--;
    }
    else{
        i++;
    }
}
for(i = 0; i < arr.length; i++){
    System.out.print(arr[i] + " ");
}
```

eg.



```
import java.util.*;

public class Solution {
    public static void swap(int[] arr, int i, int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

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

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

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

(Dutch National flag)

### Sample Input 0

5  
1 0 1 2 0

### Sample Output 0

0 0 1 1 2

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

$$\begin{cases} [i \rightarrow k] \rightarrow 0 \\ 0 \rightarrow j-1 \rightarrow 0 \\ j \rightarrow i-1 \rightarrow 1 \\ k \rightarrow n-1 \rightarrow 2 \end{cases}$$

```
import java.io.*;
import java.util.*;

public class Solution {
    public static void swap(int[] arr, int i, int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];
        for(int i = 0; i < n; i++){
            arr[i] = sc.nextInt();
        }
        int i = 0;
        while(i < n){
            int k = n;
            while(k > i){
                if(arr[i] == 0){
                    swap(arr, i, j);
                    i++;
                    k--;
                }
            }
            else if(arr[i] != 0){
                i++;
            }
        }
        for (k = 0; k < arr.length; k++){
            System.out.print(arr[k] + " ");
        }
    }
}
```

Handwritten notes and diagrams illustrating the execution of the `isick` function:

- Initial state: `isick` is marked as `code`.
- Iteration 1: `j` is 1, `k` is 2. The array is `0 0 0 1 2 2 1 0 0`. The element at index 1 (0) is marked with a circle and a cross.
- Iteration 2: `j` is 2, `k` is 3. The element at index 2 (0) is marked with a circle and a cross.
- Iteration 3: `j` is 3, `k` is 4. The element at index 3 (1) is marked with a circle and a cross.
- Iteration 4: `j` is 4, `k` is 5. The element at index 4 (2) is marked with a circle and a cross.
- Iteration 5: `j` is 5, `k` is 6. The element at index 5 (2) is marked with a circle and a cross.
- Iteration 6: `j` is 6, `k` is 7. The element at index 6 (1) is marked with a circle and a cross.
- Iteration 7: `j` is 7, `k` is 8. The element at index 7 (0) is marked with a circle and a cross.
- Iteration 8: `j` is 8, `k` is 9. The element at index 8 (0) is marked with a circle and a cross.

