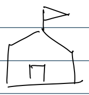
19 July 2025 09:50



Isha
mayank
Hitesh
Anmol

Aastha
Aishwarya

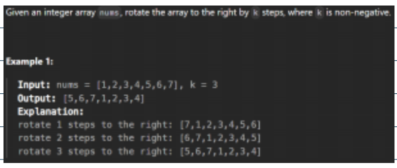
⇒

Aastha
Aishwarya
Isha
mayank
Hitesh
Anmol

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

⇒

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



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]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]
```

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

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

	0	1	2	3	4	5	6
ip arr →	1	2	3	4	5	6	7
op arr →	5	6	7	1	2	3	4
	0	1	2	3	4	5	6

k = 3

op[0] = ip[4]
op[1] = ip[5]
op[2] = ip[6]

⇓

[0 - k-1] ⇒ [4 - 6]

op[3] = ip[0]
op[4] = ip[1]
op[5] = ip[2]
op[6] = ip[3]

⇓

[k → n] ⇒ [0 - 3]

1	2	3	4	5	6	7

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 : 8	⇒	7	1	2	3	4	5	6

$$\begin{tikzcd} & arr \arrow[dl] \arrow[dr] & \\ k:1 & & k:8 \end{tikzcd}$$

⇓

k = k % n
k = 8 % 7
k : 1

$$\begin{tikzcd} & k = 13 \Rightarrow k = k \% n = k = 13 \% 7 = 6 \\ arr \arrow[ur] \arrow[dr] & \\ & k = 6 \end{tikzcd}$$

$$\begin{tikzcd} & k = 9 \Rightarrow 9 \% 7 = 2 \\ arr \arrow[ur] \arrow[dr] & \\ & k = 29 \Rightarrow 29 \% 7 = 1 \end{tikzcd}$$

	0	1	2	3	4	5	6
ip arr →	1	2	3	4	5	6	7
op arr →	5	6	7	1	2	3	4
	0	1	2	3	4	5	6

k = 3

n : 7
k : 3

op[0] = ip[4]
op[1] = ip[5]

op[3] = ip[0]
op[4] = ip[1]

n = 7
k = 3

op[0] = ip[4]
op[1] = ip[5]
op[2] = ip[6]

op[3] = ip[0]
op[4] = ip[1]
op[5] = ip[2]
op[6] = ip[3]

0
1
2
~~3~~

7-3 = 4-1 = 3

```
for( int i=0; i<n-k; i++)
                 -1
 {
    op[i] = op[i+4 ]
 }
```

3
4
5
6

3 7

```
for( int j=k; j<n; j++) {

}
```

i → 0
  → 1
  → 2

i+4 → 4
    → 5
    → 6

op[3] = ip[0]
op[4] = ip[1]
op[5] = ip[2]
op[6] = ip[3]

3
4
5
6

j → 3
  → 4
  → 5
  → 6

j-k → 0
    → 1
    → 2
    → 3

3 7

```
for( int j=k; j<n; j++) {
       op[j]  = ip[ j-k];
}
```

Steps to solve the problem

understanding requirement
Intution
Logic building
~~coding~~

ip → | 1 2 3 4 | 5 6 7 |

op → 5 6 7 1 2 3 4

| 4 3 2 1 | 5 6 7 |

| 7 6 5 | 1 2 3 4 |

| 5 6 7 1 2 3 4 |

k = 3

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

reverse   reverse

4 3 2 1   7 6 5

reverse

5 6 7 1 2 3 4   → output

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

k = 3

ip → 1 2 3 4 5

op → 3 4 5 1 2

k = 3

0 1 2 3 4 5 6 7

ip → 1 2 3 4 5 6 7 8

op → 6 7 8 1 2 3 4 5

(0, n-k)
(n-k, n-1)
(0, n-1)

5 4 3 2 1 8 7 6

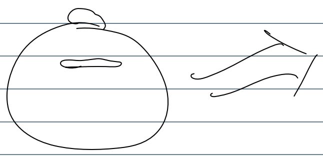
6 7 8 1 2 3 4 5

```
public class Main {
 public static void main(String[] args) {

     int[] ip = {1,2,3,4,5,6,7};
     int n = ip.length;
     int[] op = new int[n];

     int k = 3;

     for(int i=0;i<n-k-1;i++){
         op[i] = ip[i+k+1];
     }

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

     System.out.println();
     System.out.println("************");

     for(int j=k;j<n;j++){
         op[j] = ip[j-k];
     }

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

 }
}
```

```
 class Solution {
     public void rotate(int[] nums, int k) {
         int n = nums.length;
         k = k%n;
         reverseArrayPart(nums, 0, n-k-1);
         reverseArrayPart(nums, n-k, n-1);
         reverseArrayPart(nums, 0, n-1);
     }
     public static void reverseArrayPart(int[] arr, int i, int j) {
         while (i < j) {
             int temp = arr[i];
             arr[i] = arr[j];
             arr[j] = temp;
             i++;
             j--;
         }
     }
 }
```

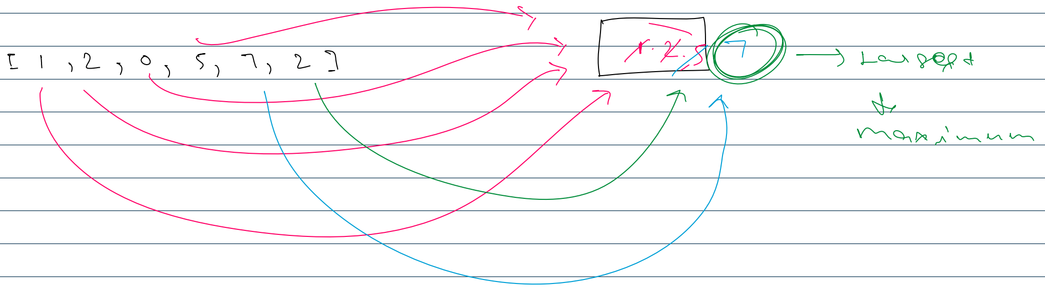
Find maximum in an array

Find maximum in an array



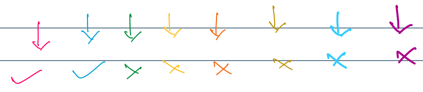
Piggy bank

It contains coin



$-\infty$ => Smallest value => -2^{31} => Integer.MIN_VALUE
∞ => largest value => 2^{31} => Integer.MAX_VALUE

[5, 7, 6, 2, 1, -4, -7, -8]



max → ~~$-\infty$~~ ~~5~~ 7

$\Downarrow$

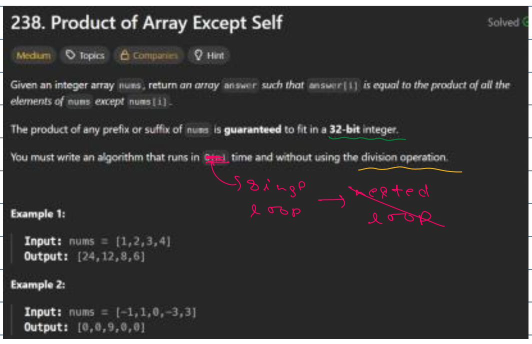
$\boxed{\text{max = 7}}$

```
if(max< arr[i]) {
    max = arr[i]
}
```

Find minimum in the array, while ignoring negative elements.

```
public class Main {

    public static void main(String[] args) {
        // int[] arr = { 3, 5, 1, 7, 8, 6, 9, -11, 5, 3 , 2, -27 };
        int[] arr = {-5, -4, -3, -2, 0};
        System.out.println(maximum1(arr));
        System.out.println(maximum2(arr));
        // System.out.println(Math.max(4, 5));

    }

    public static int maximum1(int[] arr){
        int n = arr.length;
        int max = Integer.MIN_VALUE;
        for(int i=0; i<n; i++){
            if(arr[i] > max){
                max = arr[i];
            }
        }
        return max;
    }

    public static int maximum2(int[] arr){
        int n = arr.length;
        int maxVal = Integer.MIN_VALUE;
        for(int i=0; i<n; i++){
            maxVal = Math.max(maxVal, arr[i]);
        }
        return maxVal;
    }


}


public class Main {

    public static void main(String[] args) {
        // int[] arr = { 3, 5, 1, 7, 8, 6, 9, -11, 5, 3 , 2, -27 };
        int[] arr = {-5, -4, -3, -2, 0};
        System.out.println(minimum(arr));

    }

    public static int minimum(int[] arr){
        int minVal = Integer.MAX_VALUE;
        int n = arr.length;

        for(int i=0; i<n; i++){

            // if(arr[i] < 0){
            //     continue;
            // }
            // if(arr[i] < minVal){
            //     minVal = arr[i];
            // }

            if(arr[i] >= 0){
                if(arr[i] < minVal){
                    minVal = arr[i];
                }
            }

        }
        return minVal;
    }

}
```

Product of Array Except Self



238. Product of Array Except Self

Solved

Medium · Topics · Companies · Hint

Given an integer array `nums`, return *an array* `answer` *such that* `answer[i]` *is equal to the product of all the elements of* `nums` *except* `nums[i]`.

The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

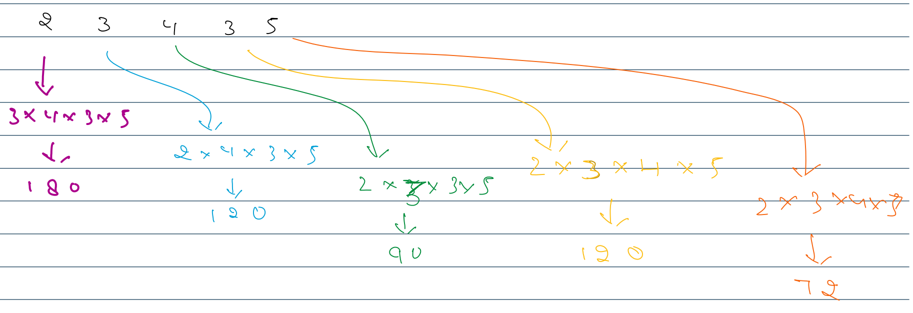
You must write an algorithm that runs in `O(n)` time and without using the division operation.

Example 1:

```
Input: nums = [1,2,3,4]
Output: [24,12,8,6]
```

Example 2:

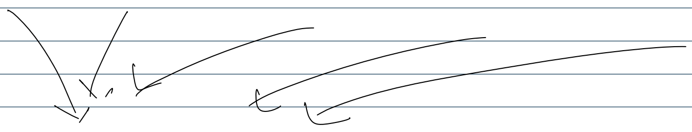
```
Input: nums = [-1,1,0,-3,3]
Output: [0,0,9,0,0]
```



op → 180 120 90 120 72

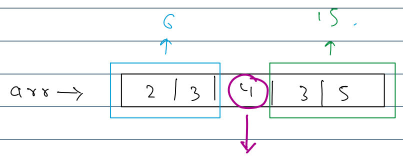
2 3 4 3 5 ⇒ product = 360

$$\frac{360}{2} \quad \frac{360}{3} \quad \frac{360}{4} \quad \frac{360}{3} \quad \frac{360}{5} = \boxed{180 \quad 120 \quad 90 \quad 120 \quad 72}$$



$$\frac{\text{product}}{arr[i]}$$

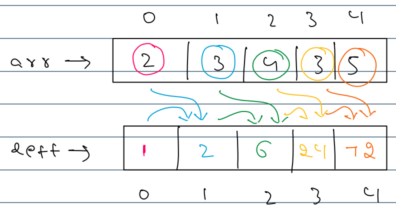
```java
public class Main {

    public static void main(String[] args) {
        int[] arr = {2, 3, 4, 3, 5};
        // System.out.println(product(arr));

        int[] nums = productOfArrayExceptSelf(arr);

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

    }

    public static int product(int[] arr){
        int prod = 1;
        int n = arr.length;
        for(int i=0; i<n; i++){
            prod = prod*arr[i];
        }
        return prod;
    }

    public static int[] productOfArrayExceptSelf(int[] arr){
        int prod = product(arr);
        int n = arr.length;
        int[] op = new int[n];

        for(int i=0;i<n;i++){
            op[i] = prod/arr[i];
        }

        return op;
    }


}
```

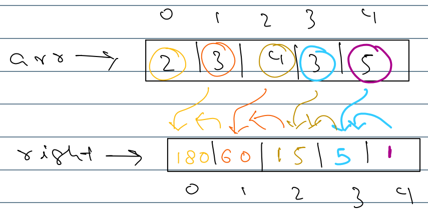


op[2] = product of left array × product of right array

6 × 15

→ 90





left[0] = 1 ×

right [n-1] = 1

```
left[1] = left[0] * arr[0]
left[2] = left[1] * arr[1]
left[3] = left[2] * arr[2]
left[4] = left[ 3] * arr[3]

for(int i = 1; i<n; i++) {
    left[i] = left[i-1] * arr[i-1]
}
```

```
right[3] = right[4] * arr[4]
right[2] = right[3] * arr[ 3]
right[1] = right[2] * arr [2]
right[0] = right[1] * arr [1]

for(int i = n-2; i >= 0; i--) {
    right[i] = right[i+1] * arr[i+1]
}
```