Maximum Product Subarray 2

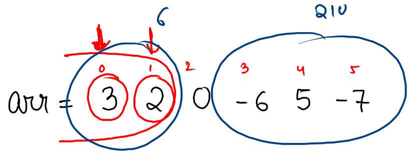
code

```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();
    }

    System.out.println(maxProductubarray(arr, n));
}

public static int maxProductubarray(int[] arr, int n) {
    int maxisf = 1;
    int minisf = 1;
    int maxProduct = Integer.MIN_VALUE;
    for (int i = 0; i < n; i++) {
        int curr = arr[i];
        int temp = maxisf;
        maxisf = Math.max( curr, Math.max( curr * maxisf, curr * minisf ));
        minisf = Math.min( curr, Math.min( curr * temp, curr * minisf ));

        maxProduct = Math.max( maxProduct, maxisf );
    }
    return maxProduct;
}
```



maxisf = 1   max Prod = ~~-∞~~ ~~3~~ ~~6~~ 210

minisf = 1

i=0, maxisf = 3 (3, 3, 3)
minisf = 3

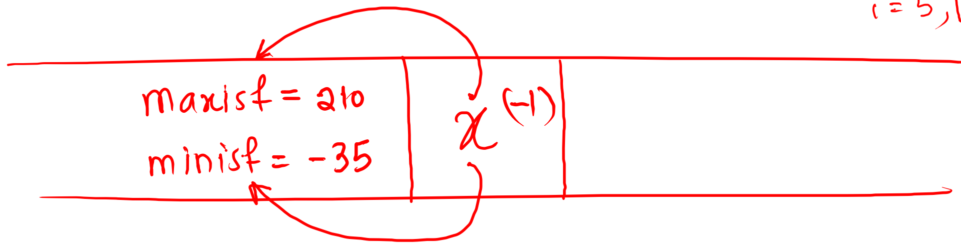
i=1, maxisf = 6 (6, 6, 2)
minisf = 2

i=2, maxisf = 0 (0, 0, 0)
minisf = 0

i=3, maxisf = 0 (0, 0, -6)
minisf = -6

i=4, $\boxed{\text{maxisf = 5}}$ (0, -30, 5)
minisf = -30

i=5, $\boxed{\text{maxisf = 210}}$ (-35, 210, -7)
minisf = -35



⇒ Two Pointers (used to save the location of something)

arr =

					a ↓
2	5	-3	5	4	7

GKSTR32 Reverse_Array

arr =

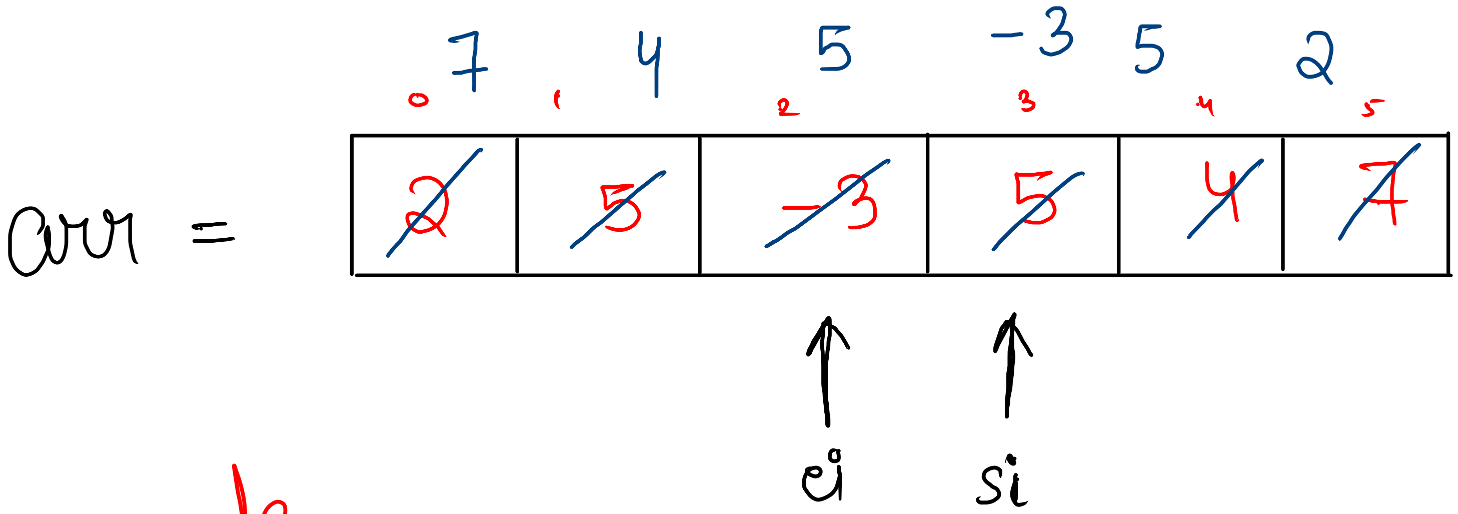
0	1	2	3	4	5
2	5	-3	5	4	7

(i/p)

arr =

7	4	5	-3	5	2

(o/p)



psudo code

```
1) make 2 pointer where si = 0, ei = n-1
2) loop until si < ei
    2.1) swap si value and ei value
    2.2) si++;
         ei--;
```

Code

$T.C = O(N)$
where N is size

$S.C = O(1)$

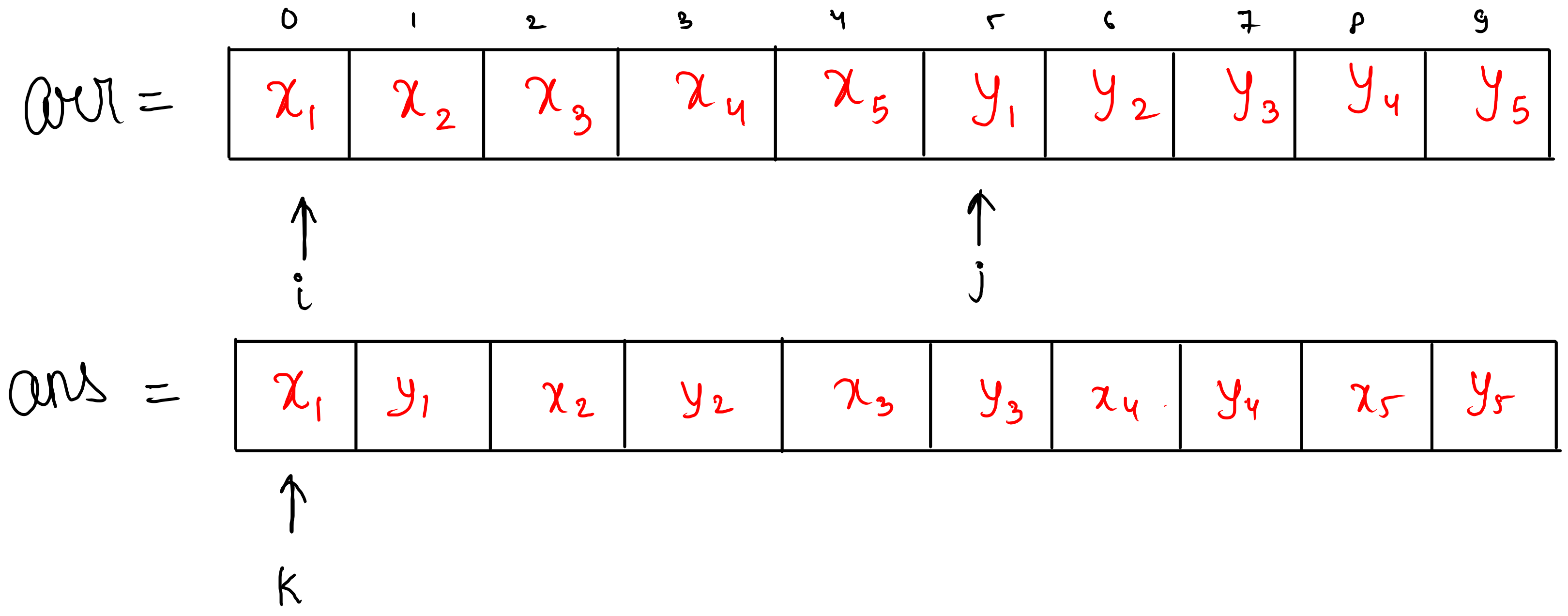
```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    reverseArray(arr, n);
}
public static void reverseArray(int[] arr, int n) {
    int si = 0;
    int ei = n - 1;
    while ( si < ei ) {
        swap( arr, si, ei );
        si++;
        ei--;
    }

    // print
    for (int i = 0; i < n; i++) {
        System.out.println(arr[i]);
    }
}
public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

Interleaving x and y Elements

$n = \cancel{10}\ 5$



0 1 2 3 4 5 6 7 8 9
arr =
x_1 x_2 x_3 x_4 x_5 y_1 y_2 y_3 y_4 y_5
i
j
ans =
x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4 x_5 y_5
k

Code

```
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[] ans = interleavingXY(arr, n);

    // print
    for (int i = 0; i < ans.length; i++) {
        System.out.print(ans[i] + " ");
    }
}
public static int[] interleavingXY(int[] arr, int n) {
    int i = 0;
    int j = n / 2;
    int k = 0;
    int[] ans = new int[n];
    while ( k < n ) {
        ans[k] = arr[i];
        i++;
        k++;

        ans[k] = arr[j];
        k++;
        j++;
    }
    return ans;
}
```

T.C = O(n)

S.C = O(n)

Rotate Right (Imp)

n = 7
arr = 1 2 3 4 5 6 7
K = 2

ans = 6 7 1 2 3 4 5

K=2

i/p arr =

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

step01 arr =

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

reverse last K elements

step02 arr =

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

reverse remaing elements

step03 arr =

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

reverse all elements

$n = 6$

imp

arr	1 2 3 4 5 6	$k = 0, 6, 12$
	6 1 2 3 4 5	$k = 1, 7, 13$
	5 6 1 2 3 4	$k = 2, 8, 14$
	4 5 6 1 2 3	$k = 3, 9, \bigcirc{15}$
	3 4 5 6 1 2	$k = 4, 10, 16$
	2 3 4 5 6 1	$k = 5, 11, 17$

$$\boxed{K = K \% n}$$

$K = 15 \% 6$

⇒ when need to rotate on left side

$\boxed{n=6}$

$k = -2 \Leftrightarrow 4$

$\boxed{K = n+K}$

arr 1 2 3 4 5 6   k=0,

6 1 2 3 4 5	k=1,	2 3 4 5 6 1,	k=-1
5 6 1 2 3 4	k=2,	3 4 5 6 1 2,	k=-2
4 5 6 1 2 3	k=3,	4 5 6 1 2 3,	k=-3
3 4 5 6 1 2	k=4,	5 6 1 2 3 4,	k=-4
2 3 4 5 6 1	k=5,	6 1 2 3 4 5,	k=-5

Code

```
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();
    int[] ans = rotateByK(arr, n, k);

    // print
    for (int i = 0; i < ans.length; i++) {
        System.out.print(ans[i] + " ");
    }
}
// main logic
public static int[] rotateByK(int[] arr, int n, int k) {
    k = k % n;
    k = n + k;
    reverseArray(arr, n - k, n - 1);
    reverseArray(arr, 0, n - k - 1);
    reverseArray(arr, 0, n - 1);
    return arr;
}

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