Maximum Product Subarray 2

	0	1	2	3	4
arr	2	3	-2	-2	-4

, ans = 24

result = ~~0~~ ~~2~~ ~~6~~ 24

```
public static int kadanesAlgo(int[] arr, int n) {
    int maxisf = 1;
    int minisf = 1;
    int result = 0;
    for (int i = 0; i < n; i++) {
        if ( arr[i] > 0 ) {
            maxisf = Math.max(maxisf, maxisf * arr[i]);
            minisf = Math.min( minisf * arr[i], 1 );
        } else if ( arr[i] == 0 ) {
            maxisf = 1;
            minisf = 1;
        } else {
            int temp = maxisf;
            maxisf = Math.max( minisf * arr[i], 1 );
            minisf = temp * arr[i];
        }

        if ( result < maxisf ) {
            result = maxisf;
        }
    }
    return result;
}
```

a

b

c

maxisf = 1

minisf = 1

i=0 (2), maxisf = (1*2, 1) = 2
minisf = (1*2, 1) = 1

i=1 (3), maxisf = (2*3, 2) = 6
minisf = (1*3, 1) = 1

i=2 (-2), maxisf = (1*-2, 1) = 1
temp = 6, minisf = 6*-2 = -12

i=3 (-2) maxisf = (-12*-2, 1) = 24 ←
temp=1 minisf = 1*-2 = -2

i=4 (-4) maxisf = (-2*-4, 1) = 8 ←
temp = 24 minisf = 24*-4 = -96

(-2)

arr

0	-2	0

```
public static int kadanesAlgo(int[] arr, int n) {
    int maxisf = 1;
    int minisf = 1;
    int result = 0;
    for (int i = 0; i < n; i++) {
        if ( arr[i] > 0 ) {
            maxisf = Math.max(maxisf, maxisf * arr[i]);
            minisf = Math.min( minisf * arr[i], 1 );
        } else if ( arr[i] == 0 ) {
            maxisf = 1;
            minisf = 1;
        } else {
            int temp = maxisf;
            maxisf = Math.max( minisf * arr[i], 1 );
            minisf = temp * arr[i];
        }

        if ( result < maxisf ) {
            result = maxisf;
        }
    }
    return result;
}
```

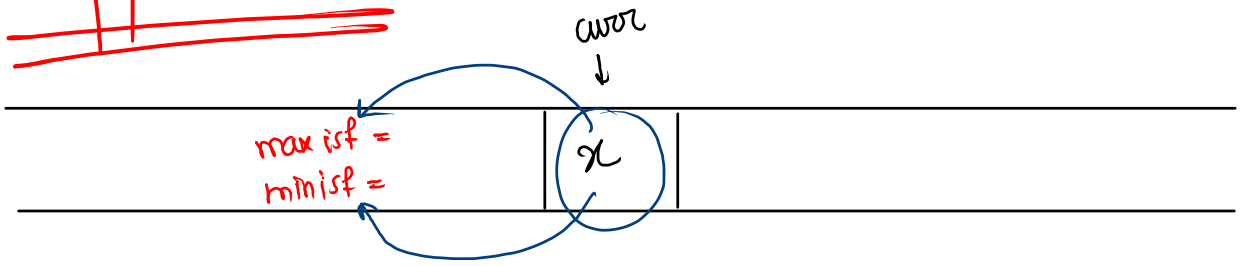
a

b

c

Note:-
work for all apart from in which ans should be zero

⇒ approch 2



maxisf = max (curr , curr * maxisf , curr * minisf)

minisf = min (curr , curr * maxisf , curr * minisf)

0	1	2
0	-2	0

result = 0

maxisf = 0
minisf = 0

(0, 0, 0)

maxisf = 0
minisf = -2

(-2, 0, 0)

maxisf = 0
minisf = 0

(0, 0, 0)

Code

```java
public static int kadanesAlgo(int[] arr, int n) {
    int maxsf = 1;
    int minisf = 1;
    int result = Integer.MIN_VALUE;
    for (int i = 0; i < n; i++) {
        int temp = maxsf;
        maxsf = Math.max( arr[i], Math.max( maxsf * arr[i], minisf * arr[i] ) );
        minisf = Math.min( arr[i], Math.min( temp * arr[i], minisf * arr[i] ) );
        result = Math.max( result, maxsf );
    }
    return result;
}
```

⇒ Two Pointer

0	1	2	3	4	5	6
7 ~~1~~	6 ~~2~~	5 ~~3~~	~~4~~ 4	~~5~~ 3	~~6~~ 2	~~7~~ 1
↑ left						↑ right

psudo code

```
left = 0, right = n-1;

while ( left < right) {

        swap( arr, left, right)
        left++;
        right--;
}
```

code

```
public static void reverseArray(int[] arr, int n) {
    int left = 0;
    int right = n - 1;
    while ( left < right ) {
        int temp = arr[left];
        arr[left] = arr[right];
        arr[right] = temp;

        left++;
        right--;
    }

    for (int i = 0; i < n; i++) {
        System.out.println(arr[i]);
    }
}
```

Imp

Interleaving x and y Elements

$n = 10$

0	1	2	3	4	5	6	7	8	9
x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5

left

right

ans

0	1	2	3	4	5	6	7	8	9
x_1	y_1	x_1	y_2	x_3	y_3	x_4	y_4	x_5	y_5

K

psudo code

create answer array

loop → K

↳ ans[K] = arr[left]

K++;

left;

↳ ans[K] = arr[right]

K++;

right;

Code

```java
public static void interleavingXY(int[] arr, int n) {
    int[] ans = new int[n];
    int left = 0;
    int right = n / 2;
    int k = 0;
    while ( k < n ) {
        ans[k] = arr[left];
        k++;
        left++;
        ans[k] = arr[right];
        k++;
        right++;
    }

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

dry run

$n = 8$

	left	left	left	left	left / right	right	right	right	right
arr	1	2	3	4	5	6	7	8	
ans	1	5	2	6	3	7	4	8	
	K	K	K	K	K	K	K	K	K

Zeroes and Ones (with linear T.C)

arr

0	1	1	0	0	1	0	1	0	1	1	0

arr

0	0	0	0	0	0	1	1	1	1	1	1

logic

arr

0	0	0	0	0	0	1	1	1	1	1	1

i

j

psudo code

check j element each time

a) ↳ if j ele. is 1, j++

b) ↳ else j ele. is 0, swap(i, j)
i++, j++

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();
    }

    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 if ( arr[j] == 0 ) {
            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;
}
```