Previous.

↳ 1. Max prod. of 3 ele.

$-4 \quad -2 \quad 1 \quad 7 \quad 2$

$\Downarrow$

_ _ _

_ _ _

_ _ _

$\boxed{-4 \quad -2} \quad \boxed{1 \quad 2 \quad \textcircled{7}}$

$$\left.\begin{array}{l} \rightarrow arr[0] \cdot arr[1] \cdot arr[n-1] \\ \rightarrow arr[n-1] \cdot arr[n-2] \cdot arr[n-3] \end{array}\right\}$$

2. Wave Sorting.

$-4 \quad -2 \quad 7 \quad 1 \quad 2$

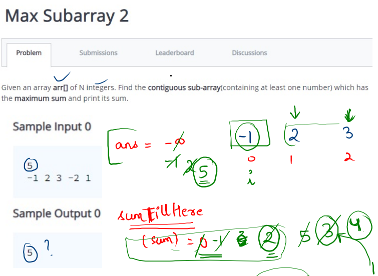
$\Downarrow$

$-4 \quad -2 \quad 1 \quad 2 \quad 7$

✓

$-2 \geq -4 \leq 2 \geq 1 \leq 7$

Max. Sum Subarray. (Kadane's Algo).



Max Subarray 2

Problem Submissions Leaderboard Discussions

Given an array **arr[]** of N integers. Find the **contiguous sub-array**(containing at least one number) which has the **maximum sum** and print its sum.

Sample Input 0

```
5
-1 2 3 -2 1
```

Sample Output 0

```
5
```

ans = $-\infty$ ~~-1~~ ~~2~~ 5

-1	2	3	-2	1
0	1	2	3	4
i				

sum Till Here
(sum) = ~~0~~ ~~-1~~ ~~2~~ 5 ~~3~~ 4

2 + 3

3 > -2

(sum < 0)
↳ start new subarray

sum ≥ 0
↳ contribute to previous subarray.

finalAns = $-\infty$ ~~-1~~ ~~2~~ 5

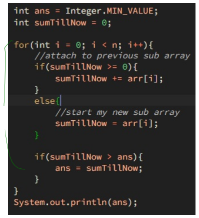
-1	2	3	-2	1
0	1	2	3	4
i	i	i	i	i

sum ≥ 0
add in previous

sum < 0
start new.

sum Till Here
= ~~0~~ ~~-1~~ ~~2~~ ~~5~~ ~~3~~ 4

3 + 1 = 4



```
int ans = Integer.MIN_VALUE;
int sumTillNow = 0;

for(int i = 0; i < n; i++){
    //attach to previous sub array
    if(sumTillNow >= 0){
        sumTillNow += arr[i];
    }
    else{
        //start my new sub array
        sumTillNow = arr[i];
    }

    if(sumTillNow > ans){
        ans = sumTillNow;
    }
}
System.out.println(ans);
```

-1	2	3	-2	15
0	1	2	3	4
i	i	i	i	i

≥ 0

sumTN = ~~0~~ ~~-1~~ ~~2~~ ~~5~~ ~~3~~ 18
ans = $-\infty$ ~~-1~~ ~~2~~ ~~5~~ 18

eg. -1 -2 -4 -23 -79

sumTN = ~~0~~ ~~-1~~ ~~-2~~ ~~-4~~ ~~-23~~ -79
ans = $-\infty$ -1

-1 > -23

-1	2	3	-2	1
0	1	2	3	4
i	i	i	i	i

ans = $-\infty$ ~~-1~~ ~~2~~ 5
sum TN = ~~0~~ ~~-1~~ ~~2~~ 5 ~~3~~ 4

1 < 5
2 < 5
3 < 5
4 < 5
~~5 < 5~~

5 -2 = 3

Maximum Product Subarray 2

Given an integer array nums, find the contiguous subarray within an array (containing at least one number) which has the largest product.

Note: According to testcases answer will not overflow long

Sample Input 0

```
4
2 3 -2 4
```

Sample Output 0

```
6
```

arr[0]   2   3   -2   4   -1
         0   1   2   3   4

ans = ~~2~~ 6

maxProd TiN = ~~2~~ ~~6~~ -~~2~~ ~~4~~ 48

minProd TiN = ~~2~~ ~~3~~ -1~~2~~ -4~~8~~ -4

max (-1 -4 , 48)

min (-1 , +4, 48)

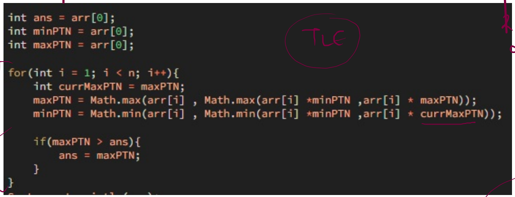
Code ?

max .
- arr[i].
- arr[i] x max PTN
- arr[i] x $\boxed{\text{minPTN}}$

min .
- arr[i]
- arr[i] * max PTN
- arr[i] * min PTN

TC

→ 2   3   -2   4
→ 2   3   2   4   -1
→ 2   3   -2   4   -1



```
int ans = arr[0];
int minPTN = arr[0];
int maxPTN = arr[0];

for(int i = 1; i < n; i++){
    int currMaxPTN = maxPTN;
    maxPTN = Math.max(arr[i] , Math.max(arr[i] *minPTN ,arr[i] * maxPTN));
    minPTN = Math.min(arr[i] , Math.min(arr[i] *minPTN ,arr[i] * currMaxPTN));

    if(maxPTN > ans){
        ans = maxPTN;
    }
}
```

TLE

3   2   4   -1
1   2   3   4

ans = ~~2~~ 6

max = ~~2~~ ~~6~~ ~~12~~ 48 -1

min = ~~2~~ ~~3~~ ~~2~~ ~~4~~ -48

arr[i], arr[i] x max, arr[i] x min

max (-1, -1 x 48, -1 x 4)

st → n

ed → n

[st , ed]

1   2   3
0   1   2

O(n³)

3
2

1 0
0

1   2 1
0

0 1   2   3 2

2
1

2   3
1

GKSTR32 Reverse_Array (2 pointer approach)

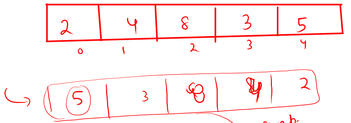
Given an integer **n**, the task is to define an array **arr[]** of size **n** & Print all the elements of the array in **reverse** order.

Sample Input 0

```
5
2
4
8
3
5
```

Sample Output 0

```
5
3
8
4
2
```

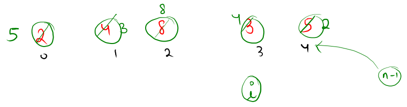


swap.

0 ⟷ n-1
1 ⟷ n-2
2 ⟷ n-3

→ 2 4 8 3 5
→ 5 3 8 4 2

without extra. space → const. space.

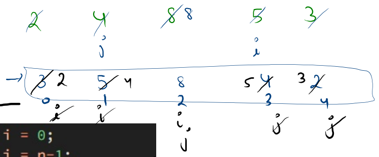


j

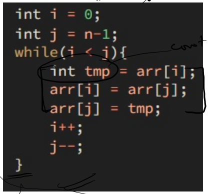
fill $i \le j$ → work.

$i > j$ → stop.

5 3 8 4 2



n = 5 90%



```
int i = 0;
int j = n-1;
while(i < j){
    int tmp = arr[i];
    arr[i] = arr[j];
    arr[j] = tmp;
    i++;
    j--;
}
```

const

0 < 4

i = 2 = j

$i < j$

?

$O(n) \rightarrow \frac{N}{2}$ $\frac{1}{2}$

$O(k.N) \approx O(N)$

2 4 8 5 3
3 5 8 4 2

HW Reverse an array with function