⇒ Kadane's algorithm

↳ used to find "maximum sum of sub array"

arr =

5	-2	3	-1	4

all subarrays

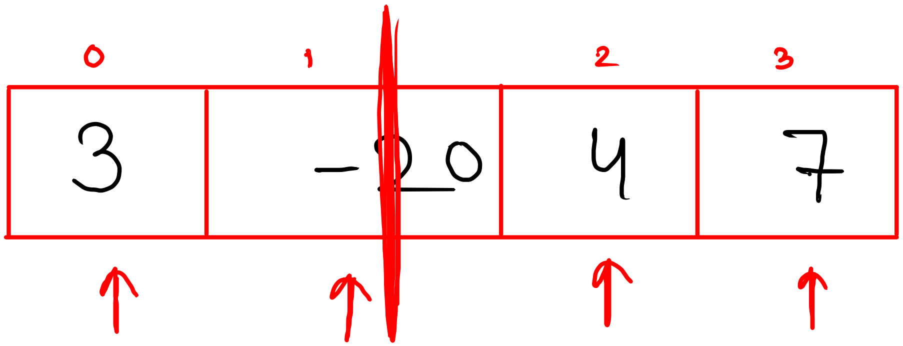
maximum

sum					
5	5				
3	5	-2			
6	5	-2	3		
5	5	-2	3	-1	
9	5	-2	3	-1	4
-2	-2				
1	-2	3			
0	-2	3	-1		
4	-2	3	-1	4	
3	3				
2	3	-1			
6	3	-1	4		
-1	-1				
3	-1	4			
4	4				

Brute force :- $O(N^3)$

Kadane's algo :- $O(N)$

arr =



maxSum = $\cancel{-\infty}$ $\cancel{3}$ $\cancel{4}$ 11

sum_so_far = $\cancel{0}$ $\cancel{3}$ $\cancel{-17}$ $\cancel{4}$ 11

Max Subarray 2

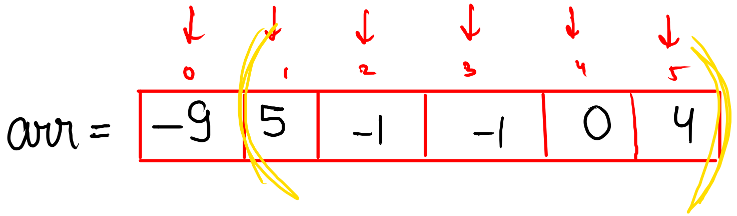
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    System.out.println(kadanesAlgo(arr, n));
}
public static int kadanesAlgo(int[] arr, int n) {
    int maxSum = Integer.MIN_VALUE;
    int sumsf = 0;
    for (int i = 0; i < n; i++) {
        if ( sumsf < 0 ) {
            sumsf = arr[i]; // reseting
        } else {
            sumsf = sumsf + arr[i]; // proceeding
        }

        if ( sumsf > maxSum ) { // check for better answer
            maxSum = sumsf;
        }
    }
    return maxSum;
}
```

T.C = O(N)

S.C = O(1)



maxSum = ~~-∞~~ ~~-9~~ ~~5~~ 7

sumsf = ~~0~~ ~~-9~~ ~~5~~ ~~4~~ ~~3~~ ~~3~~ 7

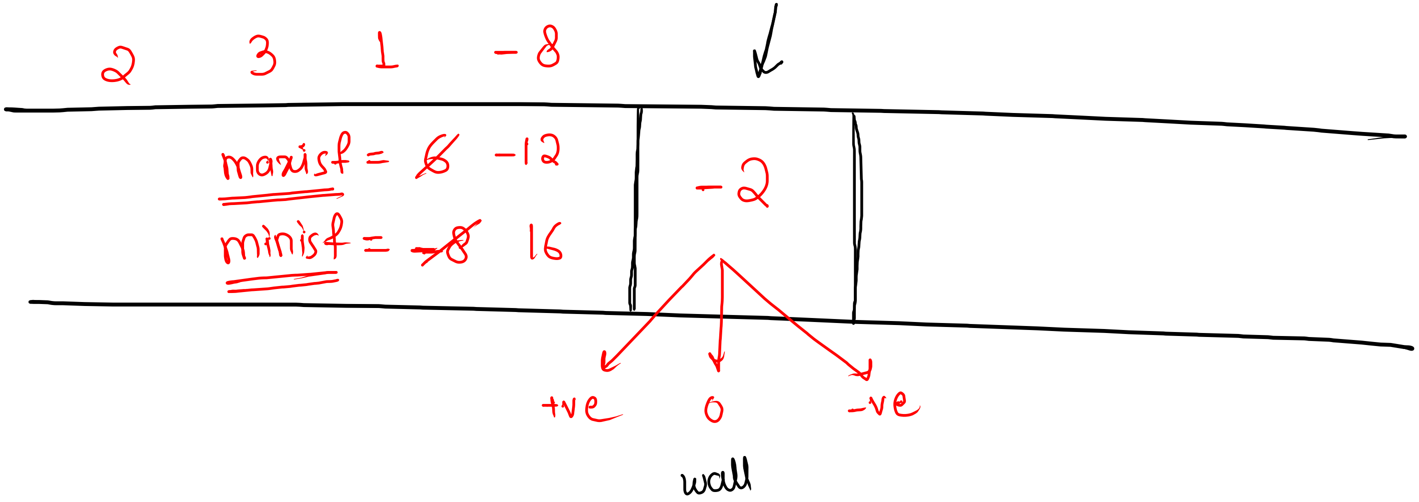
Maximum Product Subarray 2 (V. Imp)

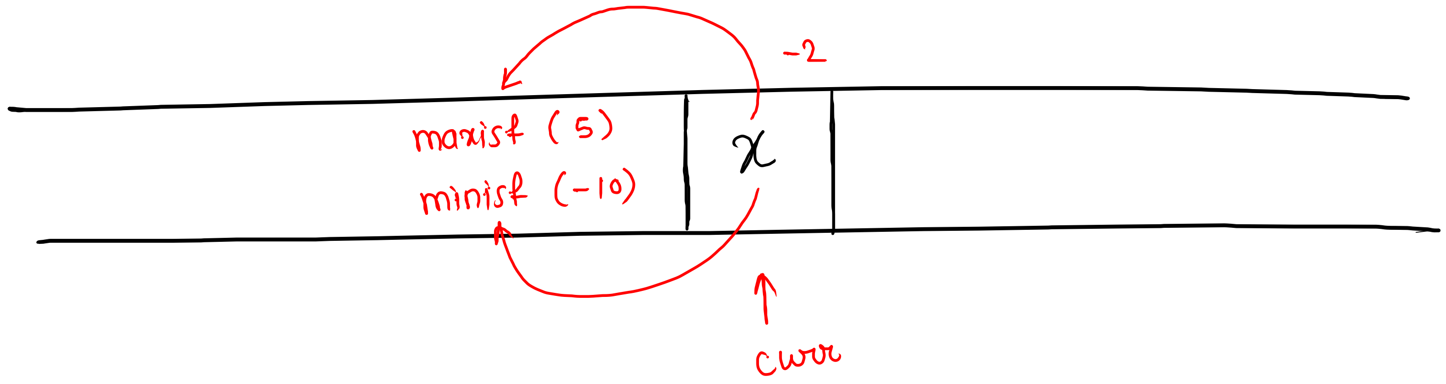
$n = 4$

arr =

2	3	-2	4	-1

ans = $2 * 3 * (-2) * 4 * (-1)$





if $x > 0$, maxisf ↑ing, minisf ↓ing

if $x < 0$, maxisf = -10, minisf = 20

---

maxi sf = max (curr * maxisf , curr * minisf , curr);

mini sf = min (curr * maxisf , curr * minisf , curr);

code

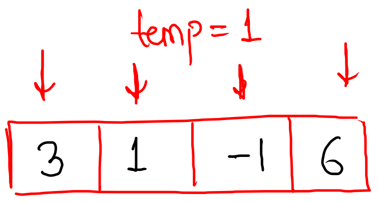
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    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;
}
```

dry run



i=0 (3, 3×1, 3×1)

maxisf = ~~1~~ ~~3~~ ~~3~~ ~~-1~~ 6

i=1 (1, 1*3, 1*3)

minisf = ~~1~~ ~~3~~ ~~1~~ ~~-3~~ -18

i=2 (-1, -1*3, -1*1)

maxProd = ~~-∞~~ ~~3~~ 6

i=3 (6, 6*-1, 6*-3)