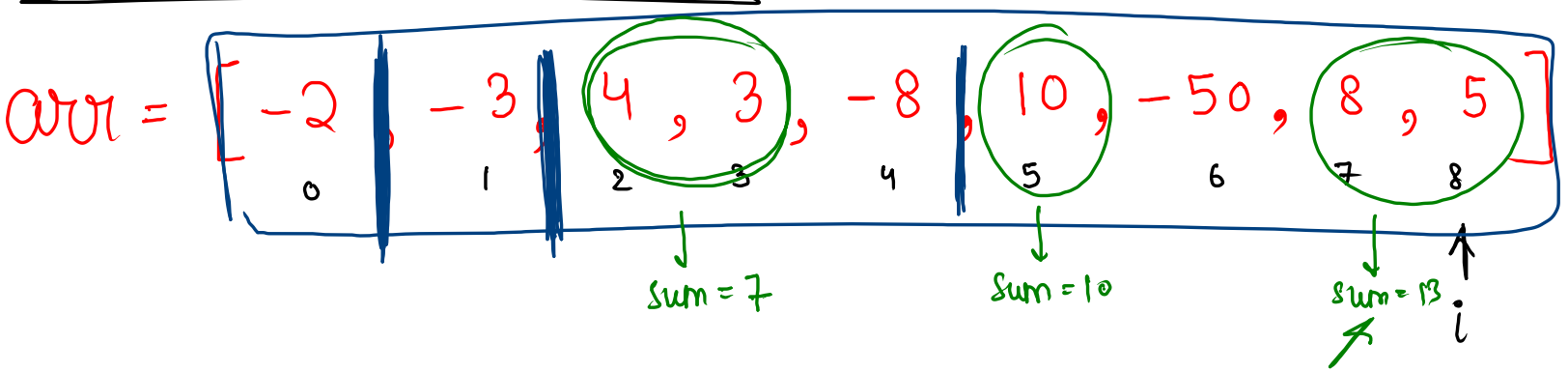
dry run of kadane's algo

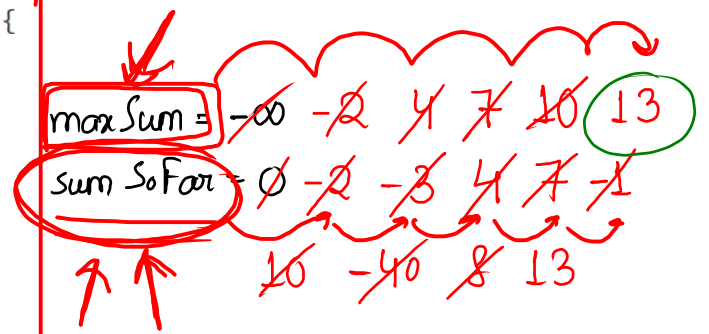



```java
public static int kadanesAlgorithm(int[] arr, int n) {
    int maxSum = Integer.MIN_VALUE;
    int sumSoFar = 0;
    for (int i = 0; i < n; i++) {
        if ( sumSoFar < 0 ) {
            sumSoFar = arr[i];
        } else {
            sumSoFar += arr[i];
        }

        if (sumSoFar > maxSum) {
            maxSum = sumSoFar;
        }
    }
    return maxSum;
}
```

a

b



Maximum Product Subarray 2 (Very Imp)

↳ O(N) time

n = 5

arr =

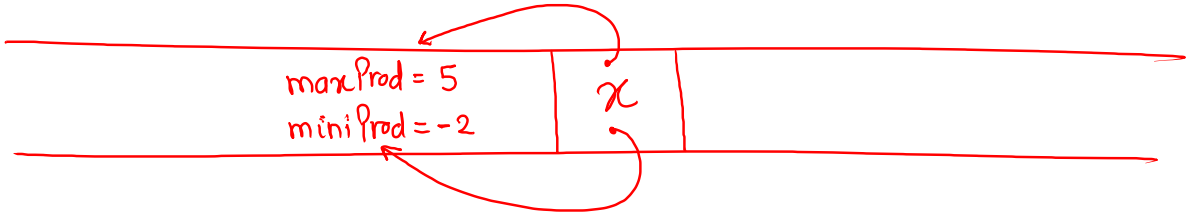
2	3	-2	4	-1

ans = 48

Note:- Here we need to store the maxProd as well as the miniProd.

Because miniProd can also give us a max answer.

Observation



$x = 2$ (+ve)

maxProd = 10
miniProd = -4

$x = -2$ (-ve)

maxProd = -10
miniProd = 4

$x = 0$

maxProd = 0
miniProd = 0

```
maxProd = max(curr * maxProd, curr * miniProd, curr);
minProd = min(curr * maxProd, curr * miniProd, curr);
```

dry run

arr = [2 , 3 , −4 , 1 , 7]
0 1 2 3 4

maxProd = 1
miniProd = 1

ans = ~~−∞~~ ~~2~~ ~~6~~ 7

i

i = 0, maxProd = (2, 2, 2) = 2
miniProd = (2, 2, 2) = 2

i = 1, maxProd = (6, 6, 3) = 6
miniProd = (6, 6, 3) = 3

i = 2, maxProd = (−24, −12, −4) = −4
miniProd = (−24, −12, −4) = −24

i = 3, maxProd = (−4, −24, 1) = 1
miniProd = (−4, −24, 1) = −24

i = 4, maxProd = (7, −168, 7) = 7
miniProd = (7, −168, 7) = −168

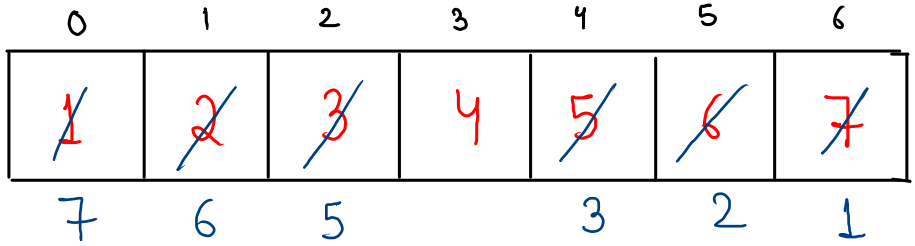
Code

```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();
    }
    int ans = maxProd(arr, n);
    System.out.println(ans);
}
public static int maxProd(int[] arr, int n) {
    int maxProd = 1;
    int miniProd = 1;
    int ans = Integer.MIN_VALUE;
    for (int i = 0; i < n; i++) {
        int curr = arr[i];
        int temp = maxProd;
        maxProd = Math.max( curr, Math.max( curr * maxProd, curr * miniProd ) );
        miniProd = Math.min( curr, Math.min( curr * temp, curr * miniProd ) );
        ans = Math.max( ans, maxProd );
    }
    return ans;
}
```

⇒ Two Pointers (Pointers are used to store location of a variable)

Note:- we can't use any extra memory and we have to reverse array in linear time

arr =



Note: **in-place** :- don't use extra memory

S.C = O(1)

psudo code

1) create pointer $i = 0$, $j = n-1$

2) traverse loop until $i < j$

   2.1) swap (arr[i], arr[j])

   2.2) 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();
    }
    reverseArray(arr, n);
    for (int i = 0; i < n; i++) {
        System.out.println(arr[i]);
    }
}
```

Logic

```java
public static void reverseArray(int[] arr, int n) {
    int i = 0;
    int j = n - 1;
    while ( i < j ) {
        swap(arr, i, j);
        i++;
        j--;
    }
}
```

operations = n/2

T.C = O(n)

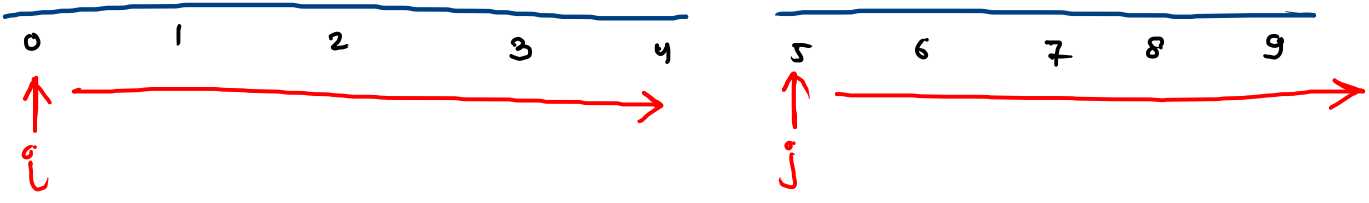
where n is size of array

```java
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

$n = 10$

arr = [$x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5$]



ans = [$x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5$]

loop n times

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();
    }
    int[] ans = interleavingXY(arr, n);
    for (int i = 0; i < n; 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];
        j++;
        k++;
    }
    return ans;
}
```