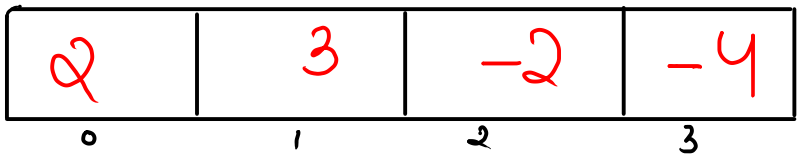
```java
public static int maxProduct(int[] arr) {
    int n = arr.length;
    int maxsf = 1;
    int minsf = 1;
    int result = 0;
    for (int i = 0; i < n; i++) {
        if ( arr[i] > 0 ) {
            maxsf = maxsf * arr[i];
            minsf = Math.min(minsf * arr[i], 1);
        } else if ( arr[i] == 0 ) {
            maxsf = 1;
            minsf = 1;
        } else {
            int temp = maxsf;
            maxsf = Math.max( minsf * arr[i], 1 );
            minsf = temp * arr[i];
        }

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

a

b

c



maxsf = 1
minsf = 1
result = 0 2 6

i=0, maxsf = 1*2 = 2
minsf = (1*2, 1) = 1

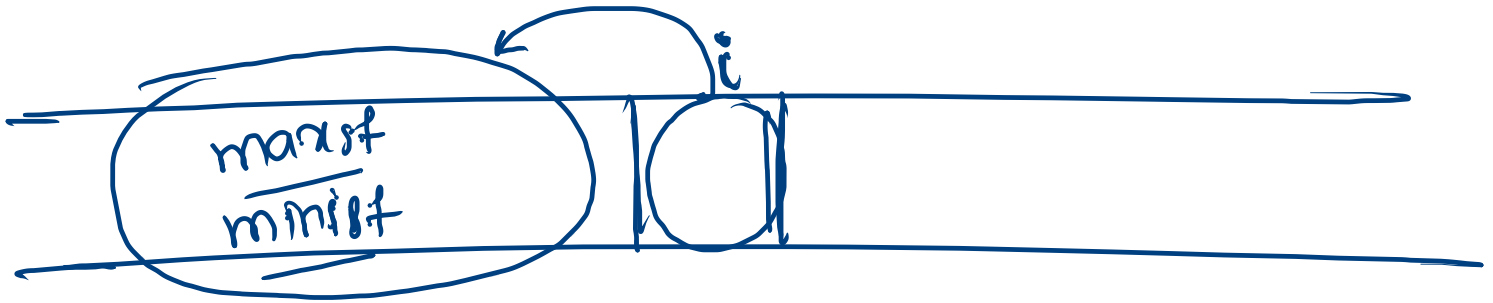
i=1, maxsf = 2*3 = 6
minsf = (1*3, 1) = 1

i=2, maxsf = (1*-2, 1) = 1
minsf = -12
temp = 1

i=3, maxsf = (-12*-4, 1) = 48
minsf = 1*-4 = -4

2	3	-2	4

↳ possible condition for a index 'i'



→ maxsf = max(arr[i] * maxsf, arr[i] * minsf, arr[i]);

→ minsf = min (arr[i] * maxsf, arr[i] * minsf, arr[i]);

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(maxProduct(arr));
}

public static int maxProduct(int[] arr) {
    int minisf = arr[0];
    int maxisf = arr[0];
    int overallAns = arr[0];
    for (int i = 1; i < arr.length; i++) {
        int temp = maxisf;
        maxisf = Math.max( arr[i] , Math.max( arr[i] * maxisf, arr[i] * minisf )  );
        minisf = Math.min( arr[i] , Math.min( arr[i] * temp, arr[i] * minisf ) ) ;
        overallAns = Math.max( overallAns, maxisf );
    }
    return overallAns;
}
```

```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(maxProduct(arr));
}

public static int maxProduct(int[] arr) {
    int minisf = arr[0];
    int maxisf = arr[0];
    int overallAns = arr[0];
    for (int i = 1; i < arr.length; i++) {
        int temp = maxisf;
        maxisf = Math.max( arr[i] , Math.max( arr[i] * maxisf, arr[i] * minisf )  );
        minisf = Math.min( arr[i] , Math.min( arr[i] * temp, arr[i] * minisf ) ) ;
        overallAns = Math.max( overallAns, maxisf );
    }
    return overallAns;
}
```

update →

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

minisf = 2 overallans = ~~2~~ ~~8~~ ~~24~~ 36

maxisf = 2

i=1, maxsf = max(-8, -8, -4) = -4
minisf = min(-8, -8, -4) = -8

i=2, maxsf = max(4, 8, -1) = 8
minisf = min(4, 8, -1) = -1

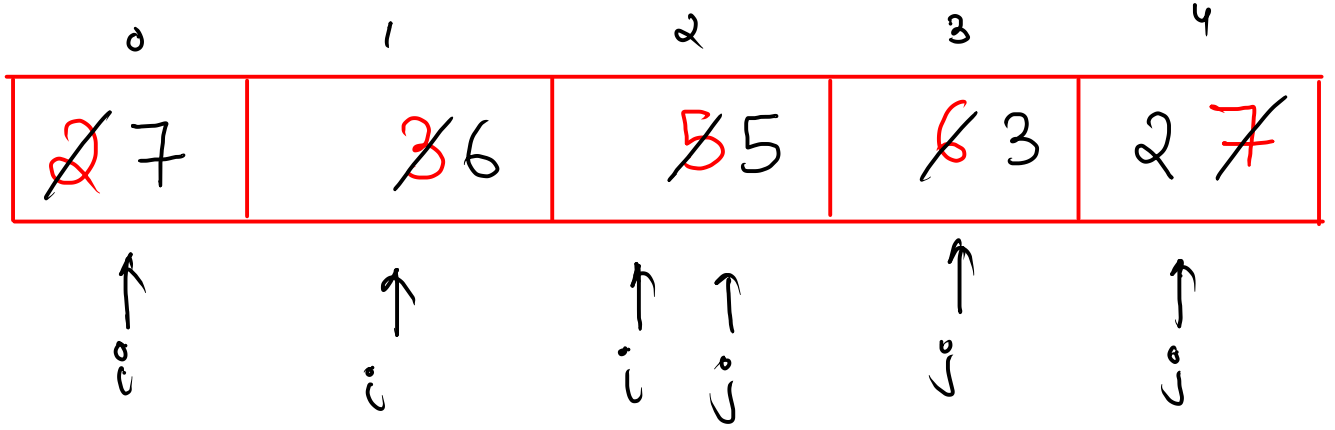
i=3, maxsf = max(24, -3, 3) = 24
minisf = min(24, -3, 3) = -3

i=4, maxsf = max(-48, 6, -2) = 6
minisf = min(-48, 6, -2) = -48

i=5, maxsf = max(36, -288, 6) = 36
minisf = min(36, -288, 6) = -288

⇒ Two Pointer

(pointer is used to point a value so that we will not lose it)



0 <= 4 , swap

1 <= 3 , swap

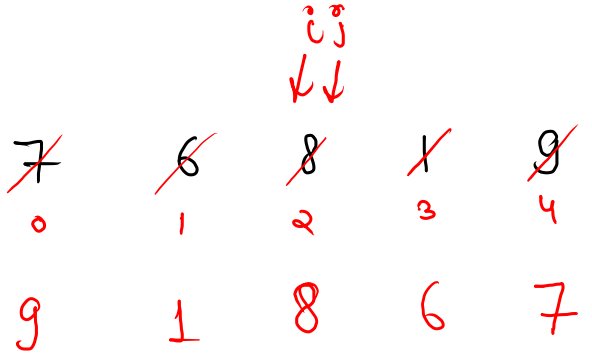
2 <= 2 , swap

(i <= j)

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

        i++;
        j--;
    }

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



for(int i=0, j=n-1 ; i<=j ; i++, j--){
swap (——);
}

Interleaving x and y Elements

$2N = 10$

$N = 5$

$[x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad y_1 \quad y_2 \quad y_3 \quad y_4 \quad y_5]$

$[x_1 \quad y_1 \quad x_2 \quad y_2 \quad x_3 \quad y_3 \quad x_4 \quad y_4 \quad x_5 \quad y_5]$

6 → 2N

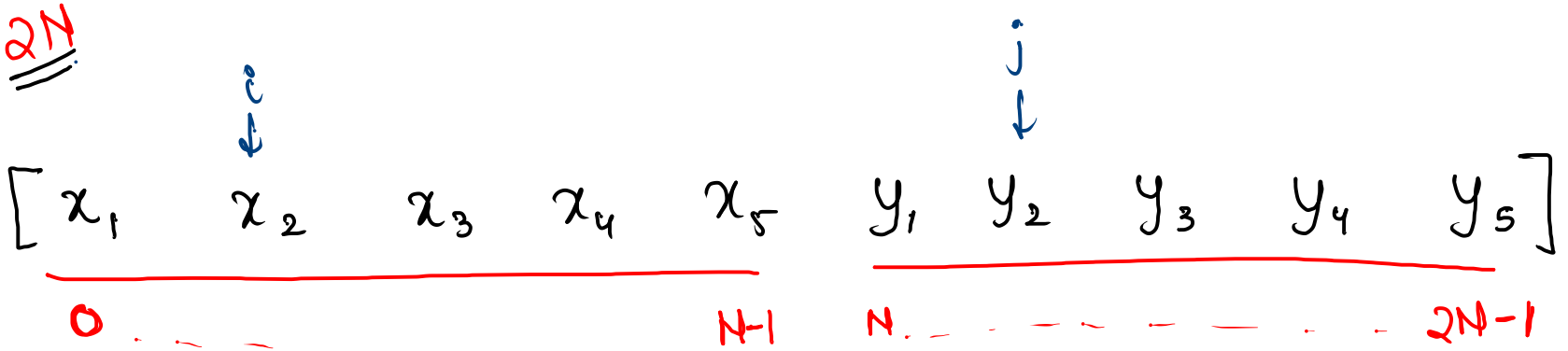
2 5 1 3 4 7

x_1 x_2 x_3 y_1 y_2 y_3

$N = 3$

2 3 5 4 1 7

size = 2N



ans = K

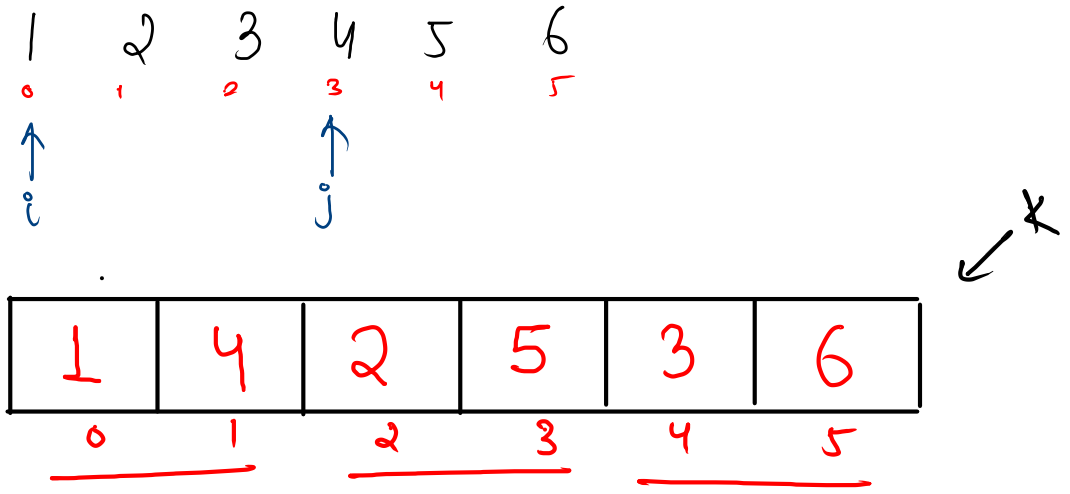
```
while ( ———— ) {
    ans[K] = arr[i];
    ans[K+1] = arr[j];
    i++, j++;
}
```

input = 2N

size = 2N/2 = N

```java
public static int[] interleaving(int[] arr, int size) {
    int i = 0;
    int j = size;
    int[] ans = new int[2 * size];
    int k = 0;
    while (k < ans.length) {
        ans[k++] = arr[i++];
        ans[k++] = arr[j++];
    }

    return ans;
}
```



iteration 1 :

K=0 , i=0
K=1 , i=1
] line 1

K=1, j=3
K=2, j=4
] line 2

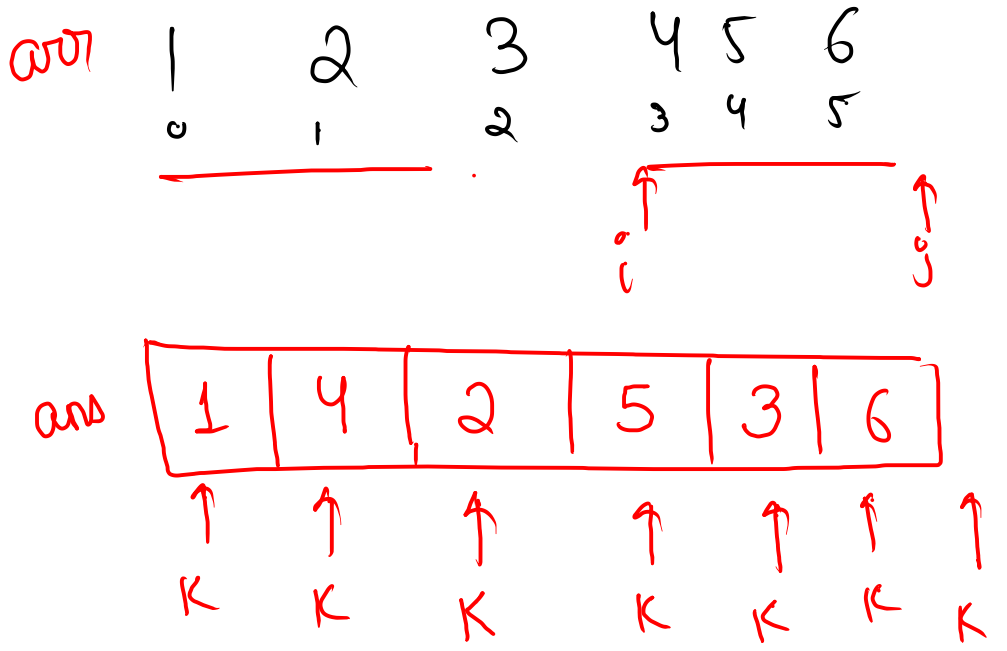
2N = 6

N = 3

```java
public static int[] interleaving(int[] arr, int size) {
    int i = 0;
    int j = size;
    int[] ans = new int[2 * size];
    int k = 0;
    while (k < ans.length) {
        ans[k] = arr[i];
        k++;
        i++;

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

    return ans;
}
```



for each loop iteration, 'k' is moving 2 times while i and j are moving once

k=0, i=0
k=1, i=1

Time complexity

O(N)