Q: Given an array, write an algorithm to reverse the same array.

Note: You should not create a new **array.**

Ex → [5, 9, 1, 8, 2, 3]

ans → [3, 2, 8, 1, 9, 5]$_{n}$

B.f [5, 9, 1, 8, 2, 3]

index→ 0 1 2 3 4 5

A.f [3, 2, 8, 1, 9, 5] $n=6$

$i=0$ → $6-0-1$ → 5

$i=1$ → $6-1-1$ → 4

$i=2$ → $6-2-1$ → 3

$i=3$ → $6-3-1$ → 2

Before Reversing → element → index → i

After Reversing → element → index → $n-i-1$

We need to somehow move the elements from their original index i to $n-i-1$

0	1	2	3	4	5
3	2	8	1	9	5

[3, 2, 8, 1, 9, 5]

0	1	2	3	4	5
3	2	8	1	9	5

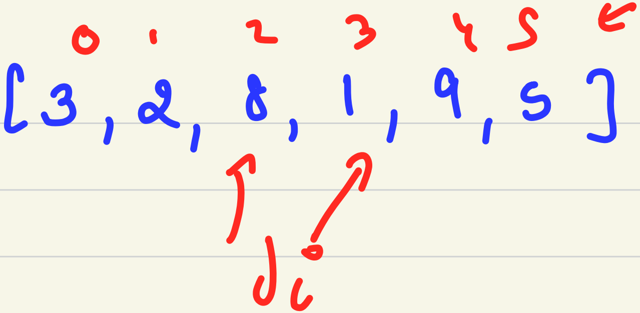
[3, 2, 8, 1, 9, 5]

	element	index
B.f →	5	0
	3	5
→	9	1
	2	4

	element	index
Af	5	5
	3	0
	9	4
	2	1

Both the termal elements are Swapping their Spots.

Swap (arr, i, j)



```
while (i <= j) {
    swap(arr, i, j);
    i+=1;
    j-=1;
}
```

We are tracking stock price of Amazon

2nd 4th

[7, 1, 5, 3, 6, 4] → prices

0th 1st 3rd 5th

prices[i] → what is the cost of the amazon stock on the ith day.

Buy the stock on any one day & sell it later.

$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 7, & 1, & 5, & 3, & 6, & 4 \end{array}$$

if we buy now

sell here

$$\left(\underbrace{-1}_{} + \underbrace{6}_{}\right) \rightarrow \boxed{5}$$

Brute force

↳ [7, 1, 5, 3, 6, 4]
0 1 2 3 4 5

① buy → 7
② buy → 1
③ Buy → 5

↳ if we just try to consider all possible casesprofit = ~~0~~ ~~4~~ 5

↓ candidate

[7, 1, 5, 3, 6, 4]

0 1 2 3 4 5

curr_min = ~~7~~ 1

↳ last best price to buy

↑i → should I sell

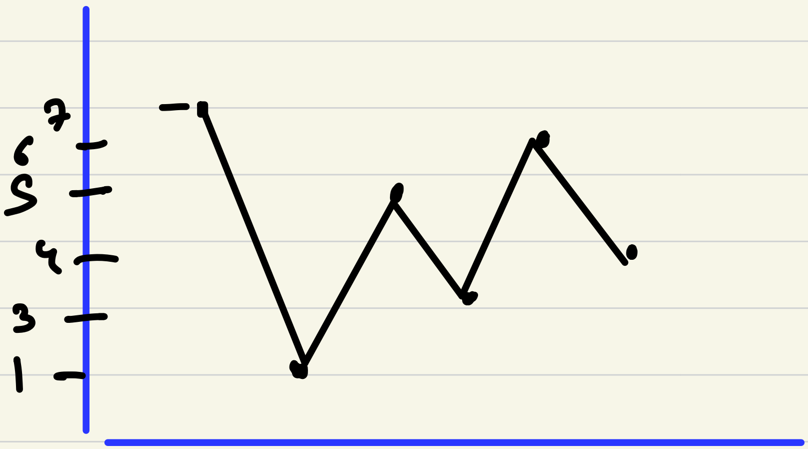
↳ we should buy as low as possible

↳ we should sell as high as possible, after buy.

#- Start tracking min values.

[7, 1, 5, 3, 6, 4]

0 1 2 3 4 5



Intuition

Small value are Gtr to buy.

Profit = ~~0~~ 7

```
  0  1  2  3  4  5  6
[ 7, 6, 4, 3, 1, 8, 6 ]
                    ↑
                    i -
```

Curr_min = ~~7~~ ~~6~~ 4, ~~3~~ 1

Best time to Buy -

```
if (price[i] > curr_min) {
    profit = Math.max(profit, price[i] - curr_min);

}
if (price[i] < curr_min) {
        curr_min = price[i];

}
```

On any ith day, I ask a question, should I sell today ??

for selling we need min stock price before the ith day.

nums1 $\underbrace{[1, 2, 3, 4, 5,]}_{\text{increasing}}$ nums2 $\underbrace{[2, 3, 8, 9]}_{\text{increasing}}$

$m = 5$ $n = 4$

$C = [1, \underbrace{2, 2, 3, 3, 4, 5, 8, 9}_{\text{increasing order}}]$

dont return C

i

nums1 [1, 3, 4, 8, 9] m=5

0 1 2 3 4

nums2 [0, 3, 5,]

0 1 2 3

j

```
if (nums1[i] < nums2[j])
{ result[k] = nums1[i];
  k++, i++;
} else {
  result[k] = nums2[j];
  k++;
  j++;
}
```

inc

result → [0, 1, 3, 3, 4, 5, 8, 9] m+n → 8

0, 1, 2, 3, 4, 5, 6, 7, 8 . . .

k

after we complete the algorithm how many elements will be inside result.

On the 0th index, Smallest among both the arrays will be present.

nums1 [a, b, c, d] → inc

nums2 [d, e, f] → inc

result → $\underbrace{[a, b, \qquad]}$

```
while (i<m && j<n) {
    if (nums1[i] <nums2[j])
  // { result[k] = nums1[i];
      K++, i++;
    } else {
      result[K] = nums2[j]
  //   K++;
      j++;
    }
}
```

this condition will be false when nums2 is exhausted.

```
while (i<m) {
    result[K] = nums1[i]
    i++; K++
}
```

this will be true if & only if we have elements in nums1

if nums2 still has elements

```
while (j<n) {
    result[K] = nums2[j];
    K++; j++;
}
```