Search In A Row Column Sorted Matrix

Given a matrix of size n x m, where every row and column is **sorted in increasing order**, and a number **x.** Find whether element x is present in the matrix or not.

```
Input:
n = 3, m = 3, x = 62
matrix[][] = {{ 3, 30, 38},
              {36, 43, 60},
              {40, 51, 69}}
Output: 0
Explanation:
62 is not present in the matrix,
so output is 0.
```

```
Input:
n = 1, m = 6, x = 55
matrix[][] = {{18, 21, 27, 38, 55, 67}}
Output: 1
Explanation: 55 is present in the matrix.
```

PREFIX SUM:

Maximum Occurred Integer In N Ranges

Given **n** integer ranges, the task is to find the maximum occurring integer in these ranges. If more than one such integer exits, find the smallest one. The ranges are given as two arrays **L**[] and **R**[]. L[i] consists of starting point of range and R[i] consists of corresponding end point of the range.

For example consider the following ranges.
L[] = {2, 1, 3}, R[] = {5, 3, 9)
Ranges represented by above arrays are.
[2, 5] = {2, **3**, 4, 5}
[1, 3] = {1, 2, **3**}
[3, 9] = {**3**, 4, 5, 6, 7, 8, 9}
The maximum occurred integer in these ranges is 3.

```
Input:
n = 4
L[] = {1,4,3,1}
R[] = {15,8,5,4}
Output: 4
Explanation: The given ranges are [1,15]
 [4, 8] [3, 5] [1, 4]. The number that
is most common or appears most times in
the ranges is 4.
```

Constraints:
$1 \leq n \leq 10^6$
$0 \leq L[i], R[i] \leq 10^6$

(IMPORTANT)

Product Of Array Except Self

Given an integer array `nums`, return *an array* `answer` *such that* `answer[i]` *is equal to the product of all the elements of* `nums` *except* `nums[i]`.

The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

You must write an algorithm that runs in `O(n)` time and without using the division operation.

Example 1:

```
Input: nums = [1,2,3,4]
Output: [24,12,8,6]
```

Example 2:

```
Input: nums = [-1,1,0,-3,3]
Output: [0,0,9,0,0]
```

Constraints:

- $2 <= nums.length <= 10^5$
- `-30 <= nums[i] <= 30`

TWO-POINTER TECHNIQUE

Pair with a given sum in a sorted array

Given a sorted array and a target number, check whether there exists a pair with a sum equal to the target.

Example One:
A[] = {1, 2, 5, 6, 10}
target = 8

Output: True

Example Two:
A[] = {1, 2, 5, 6, 10}
target = 9

Output: False

Triplet Sum In Array

Given an array arr of size n and an integer X. Find if there's a triplet in the array which sums up to the given integer X.

Example One:
A[] = {1, 4, 45, 6, 10, 8}
X = 13

Output: True
{1, 4, 8}

Example Two:
A[] = {1, 4, 45, 6, 10, 8}
X = 30

Output: False

Remove Duplicates From Sorted Array

Given a sorted array **A** consisting of duplicate elements.

Your task is to remove all the duplicates and return a sorted array of distinct elements consisting of all distinct elements present in **A**.

But, instead of returning an answer array, you have to **rearrange the given array in-place** such that it resembles what has been described above.
Hence, return a single integer, the index(1-based) till which the answer array would reside in the given array **A**.

Note: This integer is the same as the number of integers remaining inside **A** had we removed all the duplicates.
Look at the example explanations for better understanding.

Example Input

Input 1:

```
A = [1, 1, 2]
```

Input 2:

```
A = [1, 2, 2, 3, 3]
```

Example Output

Output 1:

```
2
```

Output 2:

```
3
```

Example Explanation

Explanation 1:

```
Updated Array: [1, 2, X] after rearranging. Note
that there could be any number in place of x since
we dont need it.
We return 2 here.
```

Explanation 2:

```
Updated Array: [1, 2, 3, X, X] after rearranging
duplicates of 2 and 3.
We return 3 from here.
```