5438. Minimum Number of Days to Make m Bouquets

ıbmissions (/contest/weekly-contest-193/problems/minimum-number-of-days-to-make-m-bouquets/submissions/)

:o Contest (/contest/weekly-contest-193/)

Given an integer array `bloomDay` , an integer `m` and an integer `k` .

We need to make `m` bouquets. To make a bouquet, you need to use `k` **adjacent flowers** from the garden.

The garden consists of `n` flowers, the `ith` flower will bloom in the `bloomDay[i]` and then can be used in **exactly one** bouquet.

Return *the minimum number of days* you need to wait to be able to make `m` bouquets from the garden. If it is impossible to make `m` bouquets return **-1**.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

Example 1:

```
Input: bloomDay = [1,10,3,10,2], m = 3, k = 1
Output: 3
Explanation: Let's see what happened in the first three days. x means flower bloomed and _
We need 3 bouquets each should contain 1 flower.
After day 1: [x, _, _, _, _]   // we can only make one bouquet.
After day 2: [x, _, _, _, x]   // we can only make two bouquets.
After day 3: [x, _, x, _, x]   // we can make 3 bouquets. The answer is 3.
```

Example 2:

```
Input: bloomDay = [1,10,3,10,2], m = 3, k = 2
Output: -1
Explanation: We need 3 bouquets each has 2 flowers, that means we need 6 flowers. We only
```

Example 3:

```
Input: bloomDay = [7,7,7,7,12,7,7], m = 2, k = 3
Output: 12
Explanation: We need 2 bouquets each should have 3 flowers.
Here's the garden after the 7 and 12 days:
After day 7: [x, x, x, x, _, x, x]
We can make one bouquet of the first three flowers that bloomed. We cannot make another bo
After day 12: [x, x, x, x, x, x, x]
It is obvious that we can make two bouquets in different ways.
```

Example 4:

```
Input: bloomDay = [1000000000,1000000000], m = 1, k = 1
Output: 1000000000
Explanation: You need to wait 1000000000 days to have a flower ready for a bouquet.
```

Example 5:

```
Input: bloomDay = [1,10,2,9,3,8,4,7,5,6], m = 4, k = 2
Output: 9
```

Constraints:

- `bloomDay.length == n`
- `1 <= n <= 10^5`
- `1 <= bloomDay[i] <= 10^9`
- `1 <= m <= 10^6`
- `1 <= k <= n`

JavaScript ▾

```
/**
 * @param {number[]} bloomDay
 * @param {number} m
 * @param {number} k
 * @return {number}
 */
var minDays = function(bloomDay, m, k) {

};
```

□ **Custom Testcase** Use Example Testcases