Problem 683: K Empty Slots

Problem Information

Difficulty: Hard
Acceptance Rate: 37.90%
Paid Only: Yes
Tags: Array, Binary Indexed Tree, Segment Tree, Queue, Sliding Window, Heap (Priority Queue), Ordered Set, Monotonic Queue

Problem Description

You have `n` bulbs in a row numbered from `1` to `n`. Initially, all the bulbs are turned off. We turn on **exactly one** bulb every day until all bulbs are on after `n` days.

You are given an array `bulbs` of length `n` where `bulbs[i] = x` means that on the `(i+1)th` day, we will turn on the bulb at position `x` where `i` is **0-indexed** and `x` is **1-indexed.**

Given an integer `k`, return _the**minimum day number** such that there exists two **turned on** bulbs that have **exactly** `k` bulbs between them that are **all turned off**. If there isn't such day, return `-1`._

Example 1:

Input: bulbs = [1,3,2], k = 1 **Output:** 2 **Explanation:** On the first day: bulbs[0] = 1, first bulb is turned on: [1,0,0] On the second day: bulbs[1] = 3, third bulb is turned on: [1,0,1] On the third day: bulbs[2] = 2, second bulb is turned on: [1,1,1] We return 2 because on the second day, there were two on bulbs with one off bulb between them.

Example 2:

Input: bulbs = [1,2,3], k = 1 **Output:** -1

Constraints:

* `n == bulbs.length` * `1 <= n <= 2 * 104` * `1 <= bulbs[i] <= n` * `bulbs` is a permutation of numbers from `1` to `n`. * `0 <= k <= 2 * 104`

Code Snippets

C++:

```
class Solution {
public:
int kEmptySlots(vector<int>& bulbs, int k) {

}
};
```

Java:

```
class Solution {
public int kEmptySlots(int[] bulbs, int k) {

}
}
```

Python3:

```
class Solution:
def kEmptySlots(self, bulbs: List[int], k: int) -> int:
```