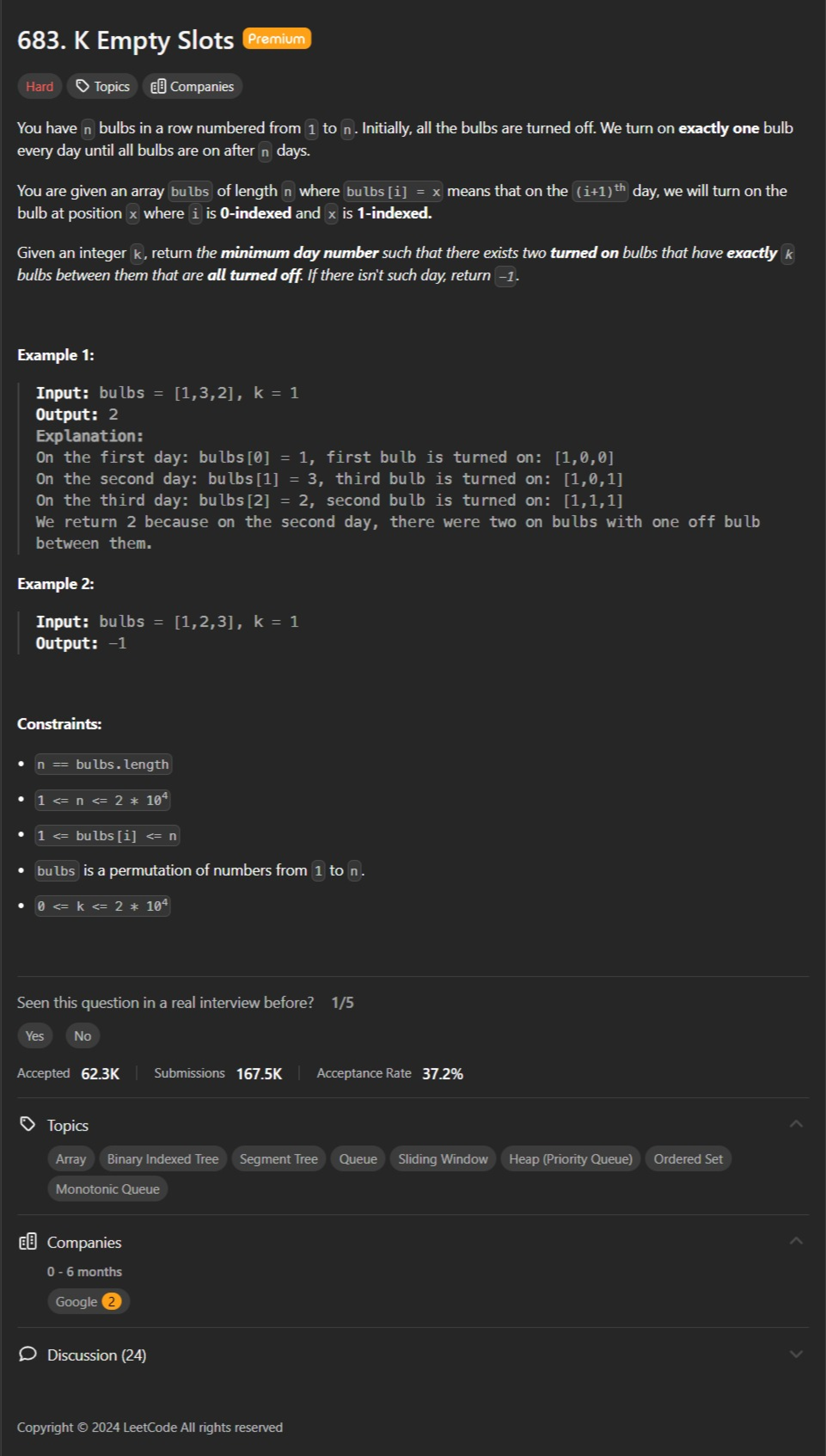

683. K Empty Slots

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`
- $1 <= n <= 2 * 10^{4}$
- `1 <= bulbs[i] <= n`
- `bulbs` is a permutation of numbers from `1` to `n`.
- $0 <= k <= 2 * 10^{4}$

Seen this question in a real interview before? 1/5

Accepted 62.3K | Submissions 167.5K | Acceptance Rate 37.2%

Topics

Array, Binary Indexed Tree, Segment Tree, Queue, Sliding Window, Heap (Priority Queue), Ordered Set, Monotonic Queue

Companies

0 - 6 months

Google 2

Discussion (24)