5227. Maximize the Topmost Element After K Moves

My Submissions (/contest/weekly-contest-284/problems/maximize-the-topmost-element-after-k-moves/submissions/)

Back to Contest (/contest/weekly-contest-284/)

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

You are given a **0-indexed** integer array `nums` representing the contents of a **pile**, where `nums[0]` is the topmost element of the pile.

In one move, you can perform **either** of the following:

- If the pile is not empty, **remove** the topmost element of the pile.
- If there are one or more removed elements, **add** any one of them back onto the pile. This element becomes the new topmost element.

You are also given an integer `k`, which denotes the total number of moves to be made.

Return *the **maximum value** of the topmost element of the pile possible after **exactly*** `k` *moves*. In case it is not possible to obtain a non-empty pile after `k` moves, return `-1`.

Example 1:

```
Input: nums = [5,2,2,4,0,6], k = 4
Output: 5
Explanation:
One of the ways we can end with 5 at the top of the pile after 4 moves is as follows:
- Step 1: Remove the topmost element = 5. The pile becomes [2,2,4,0,6].
- Step 2: Remove the topmost element = 2. The pile becomes [2,4,0,6].
- Step 3: Remove the topmost element = 2. The pile becomes [4,0,6].
- Step 4: Add 5 back onto the pile. The pile becomes [5,4,0,6].
Note that this is not the only way to end with 5 at the top of the pile. It can be shown that 5 is the largest answer possib
```

Example 2:

```
Input: nums = [2], k = 1
Output: -1
Explanation:
In the first move, our only option is to pop the topmost element of the pile.
Since it is not possible to obtain a non-empty pile after one move, we return -1.
```

Constraints:

- `1 <= nums.length <=` 10^5
- `0 <= nums[i], k <=` 10^9

JavaScript

```javascript
const maximumTop = (a, k) => {
    let n = a.length;
    if (n == 1) {
        if (k % 2 != 0) {
            return -1;
        } else {
           return a[0];
        }
    }
    if (k > n) { // possible max in whole array
        k %= n;
        let max = Math.max(...a);
        return max;
    } else if (k < n) { // possible max in subarray [0, k]
        if (k == 0) return a[0];
        /*
```

```
          remove k - 1, max(k-1) element, add max to top
          remove k, currentTop
         */
        let d = a.slice(0, k - 1);
        let currentTop = a[k];
        return Math.max(Math.max(...d), currentTop);
    } else {
        let max = Math.max(...a.slice(0, -1));
        return max;
    }
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/658909190/) More Details (/submissions/detail/658909190/)

Share your acceptance!

2