2659. Make Array Empty

My Submissions (/contest/biweekly-contest-103/problems/make-array-empty/submissions/)

You are given an integer array nums containing **distinct** numbers, and you can perform the following operations **until the**array is empty:

Back to Contest (/contest/biweekly-contest-103/)

User Accepted: 443

- If the first element has the **smallest** value, remove it
- Otherwise, put the first element at the **end** of the array.

(/) Explore(/explore/) Problems(/problemset/all/) Interview Contest

Return an integer denoting the number of operations it takes to make nums empty.

User Accepted:	443
User Tried:	4948
Total Accepted:	509
Total Submissions:	11283
Difficulty:	Hard

Example 1:

Input: nums = [3,4,-1]
Output: 5

Operation	Array	
1	[4, -1, 3]	
2	[-1, 3, 4]	
3	[3, 4]	
4	[4]	
5	[]	

Example 2:

Input: nums = [1,2,4,3]

Output: 5

Operation	Array
1	[2, 4, 3]
2	[4, 3]
3	[3, 4]
4	[4]
5	[]

Example 3:

Input: nums = [1,2,3]

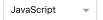
Output: 3

Operation	Array
1	[2, 3]
2	[3]
3	

Constraints:

- 1 <= nums.length <= 10^5
- $-10^9 \le nums[i] \le 10^9$
- All values in nums are distinct.

Discuss (https://leetcode.com/problems/make-array-empty/discuss)





```
2 v functètnaFenMirck((n)){fill(0);
        return { query, update, rangeSum, tree }
 4
        function query(i) {
 5
             let sum = 0;
 6
             for (i++; i > 0; i = parent(i)) sum += a[i];
 7
             return sum;
 8
 9,
        function update(i, v) {
10
             for (i++; i < n; i = next(i)) a[i] += v;
11
        function rangeSum(l, r) {
12 •
13
             return query(r) - query(l - 1);
14
15 ▼
        function parent(x) {
             return x - lowestOneBit(x);
16
17
18
         function next(x) {
             return x + lowestOneBit(x);
19
20
21 •
        function lowestOneBit(x) {
            return x & -x;
22
23
24 ▼
        function tree() {
25
             return a;
26
27
    }
28
    const countOperationsToEmptyArray = (a) => {
29
        let n = a.length, fen = new Fenwick(n + 3), res = n, pre = 0;
30
31
        a = a.map((x, i) \Rightarrow [x, i + 1]).sort((x, y) \Rightarrow x[0] - y[0] || x[1] - y[1]);
32
        a.map(e \Rightarrow {
            let [x, idx] = e;
33
             res += idx - 1 - fen.rangeSum(0, idx - 1);
34
             if (pre < idx) {
35 ▼
36
                 res -= pre - fen.rangeSum(0, pre);
37 ▼
            } else {
38
                 res += (n - pre) - (fen.rangeSum(0, n) - fen.rangeSum(0, pre));
39
40
            pre = idx:
41
             fen.update(idx, 1);
42
        })
43
        return res;
   };
44
```

☐ Custom Testcase

United States (/region)

Use Example Testcases

