2659. Make Array Empty

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

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

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

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 <= $ `nums[i]` $<= 10^9$
- All values in `nums` are **distinct**.

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

JavaScript ▾

```
 1  function SegmentTreeRSQ(n) {
 2      let h = Math.ceil(Math.log2(n)), len = 2 * 2 ** h, a = Array(len).fill(0);
 3      return { update, query, rangeSum, tree }
 4      function update(pos, v) {
 5          a[n + pos] = v;
 6          for (let i = parent(n + pos); i >= 1; i = parent(i)) pushup(i);
 7      }
 8      function pushup(i) {
 9          a[i] = a[left(i)] + a[right(i)];
10      }
11      function query(l, r) { // [L, R)
12          let sum = 0;
13          if (l >= r) return 0;
14          l += n;
15          r += n;
16          for (; l < r; l = parent(l), r = parent(r)) {
17              if (l & 1) sum += a[l++];
18              if (r & 1) sum += a[--r];
19          }
20          return sum;
21      }
22      function rangeSum(l, r) {
23          return query(0, r + 1) - query(0, l);
24      }
25      function parent(i) {
26          return i >> 1;
27      }
28      function left(i) {
29          return 2 * i;
30      }
31      function right(i) {
32          return 2 * i + 1;
33      }
34      function tree() {
35          return a;
36      }
37  }
38
39  const countOperationsToEmptyArray = (a) => {
40      let n = a.length, st = new SegmentTreeRSQ(n + 3), res = n, pre = 0, f = Array(n + 1).fill(0);
41      a = a.map((x, i) => [x, i + 1]).sort((x, y) => x[0] - y[0] || x[1] - y[1]);
42      a.map(e => {
43          let [, idx] = e;
44          res += idx - 1 - st.rangeSum(0, idx - 1);
45          if (pre < idx) {
46              res -= pre - st.rangeSum(0, pre);
47          } else {
48              res += (n - pre) - (st.rangeSum(0, n) - st.rangeSum(0, pre));
49          }
50          pre = idx;
51          st.update(idx, ++f[idx]);
52      })
53      return res;
54  };
```

☐ **Custom Testcase**   (Use Example Testcases)

⏵ Run   ☁ Submit

Submission Result: *Accepted* (/submissions/detail/948714814/) ❓   (More Details ❯ (/submissions/detail/948714814/))

Share your acceptance!

◀ 3