6351. Find Score of an Array After Marking All Elements

My Submissions (/contest/biweekly-contest-100/problems/find-score-of-an-array-after-marking-all-elements/submissions/)

Back to Contest (/contest/biweekly-contest-100/)

You are given an array `nums` consisting of positive integers.

Starting with `score = 0`, apply the following algorithm:

- Choose the smallest integer of the array that is not marked. If there is a tie, choose the one with the smallest index.
- Add the value of the chosen integer to `score`.
- Mark **the chosen element and its two adjacent elements if they exist**.
- Repeat until all the array elements are marked.

Return *the score you get after applying the above algorithm*.

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

Example 1:

```
Input: nums = [2,1,3,4,5,2]
Output: 7
Explanation: We mark the elements as follows:
- 1 is the smallest unmarked element, so we mark it and its two adjacent elements: [2,1,3,4,5,2].
- 2 is the smallest unmarked element, so we mark it and its left adjacent element: [2,1,3,4,5,2].
- 4 is the only remaining unmarked element, so we mark it: [2,1,3,4,5,2].
Our score is 1 + 2 + 4 = 7.
```

Example 2:

```
Input: nums = [2,3,5,1,3,2]
Output: 5
Explanation: We mark the elements as follows:
- 1 is the smallest unmarked element, so we mark it and its two adjacent elements: [2,3,5,1,3,2].
- 2 is the smallest unmarked element, since there are two of them, we choose the left-most one, so we mark the one at index 0 a
- 2 is the only remaining unmarked element, so we mark it: [2,3,5,1,3,2].
Our score is 1 + 2 + 2 = 5.
```

Constraints:

- `1 <= nums.length <= 10^5`
- `1 <= nums[i] <= 10^6`

Java

```java
// package leetcode.biweekly.r100;

import java.util.*;

public class Solution {
    public long findScore(int[] a) {
        int n = a.length;
        long res = 0;
        TreeMap<Integer, TreeSet<Integer>> m = counter_value_in_indexA_in(a);
        while (m.size() > 0) {
            int x = m.firstKey();
            res += x;
            TreeSet<Integer> ia = m.get(x), iar = null, ial = null;
            int pickIdx = ia.first();
            Integer l = null, r = null;
            if (pickIdx + 1 < n) {
                r = a[pickIdx + 1];
                iar = m.get(r);
            }
            if (pickIdx - 1 >= 0) {
                l = a[pickIdx - 1];
```

```
                ial = m.get(l);
            }
            // tr(x, l, r, m);
            if (l != null) op(m, ial, pickIdx - 1, l);
            if (r != null) op(m, iar, pickIdx + 1, r);
            op(m, ia, pickIdx, x);
            // tr("remove", m);
        }
        return res;
    }

    void op(TreeMap<Integer, TreeSet<Integer>> m, TreeSet<Integer> ia, int idx, int v) {
        if (ia != null) {
            ia.remove(idx);
            if (ia.size() > 0) {
                m.put(v, ia);
            } else {
                m.remove(v);
            }
        }
    }

    TreeMap<Integer, TreeSet<Integer>> counter_value_in_indexA_in(int[] a) {
        TreeMap<Integer, TreeSet<Integer>> m = new TreeMap<>();
        for (int i = 0; i < a.length; i++) {
            if (!m.containsKey(a[i])) m.put(a[i], new TreeSet<>());
            m.get(a[i]).add(i);
        }
        return m;
    }
}
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/917593508/) | More Details > (/submissions/detail/917593508/)

Share your acceptance!

1