

6927. Minimum Index of a Valid Split

[My Submissions \(/contest/weekly-contest-354/problems/minimum-index-of-a-valid-split/submissions/\)](#)
[Back to Contest \(/contest/weekly-contest-354/\)](#)

An element x of an integer array arr of length m is **dominant** if $\text{freq}(x) * 2 > m$, where $\text{freq}(x)$ is the number of occurrences of x in arr . Note that this definition implies that arr can have **at most one** dominant element.

You are given a **0-indexed** integer array $nums$ of length n with one dominant element.

You can split $nums$ at an index i into two arrays $nums[0, \dots, i]$ and $nums[i + 1, \dots, n - 1]$, but the split is only **valid** if:

- $0 \leq i < n - 1$
- $nums[0, \dots, i]$, and $nums[i + 1, \dots, n - 1]$ have the same dominant element.

Here, $nums[i, \dots, j]$ denotes the subarray of $nums$ starting at index i and ending at index j , both ends being inclusive. Particularly, if $j < i$ then $nums[i, \dots, j]$ denotes an empty subarray.

Return the **minimum** index of a **valid split**. If no valid split exists, return -1 .

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

Example 1:

Input: $nums = [1,2,2,2]$

Output: 2

Explanation: We can split the array at index 2 to obtain arrays $[1,2,2]$ and $[2]$.

In array $[1,2,2]$, element 2 is dominant since it occurs twice in the array and $2 * 2 > 3$.

In array $[2]$, element 2 is dominant since it occurs once in the array and $1 * 2 > 1$.

Both $[1,2,2]$ and $[2]$ have the same dominant element as $nums$, so this is a valid split.

It can be shown that index 2 is the minimum index of a valid split.

Example 2:

Input: $nums = [2,1,3,1,1,1,7,1,2,1]$

Output: 4

Explanation: We can split the array at index 4 to obtain arrays $[2,1,3,1,1]$ and $[1,7,1,2,1]$.

In array $[2,1,3,1,1]$, element 1 is dominant since it occurs thrice in the array and $3 * 2 > 5$.

In array $[1,7,1,2,1]$, element 1 is dominant since it occurs thrice in the array and $3 * 2 > 5$.

Both $[2,1,3,1,1]$ and $[1,7,1,2,1]$ have the same dominant element as $nums$, so this is a valid split.

It can be shown that index 4 is the minimum index of a valid split.

Example 3:

Input: $nums = [3,3,3,3,7,2,2]$

Output: -1

Explanation: It can be shown that there is no valid split.

Constraints:

- $1 \leq nums.length \leq 10^5$
- $1 \leq nums[i] \leq 10^9$
- $nums$ has exactly one dominant element.

Java



```

1 class Solution {
2     public int minimumIndex(List<Integer> a) {
3         int n = a.size();
4         Map<Integer, Integer> rm = counter(a), lm = new HashMap<>();
5         TreeSet<Node> r = new TreeSet<>(), l = new TreeSet<>();
6         for (int x : rm.keySet()) r.add(new Node(x, rm.get(x)));
7         for (int i = 0; i < n; i++) {
8             int v = a.get(i), lcntOld = lm.getOrDefault(v, 0), rcntOld = rm.getOrDefault(v, 0);
9             l.remove(new Node(v, lcntOld));
10            l.add(new Node(v, lcntOld + 1));
11            r.remove(new Node(v, rcntOld));
12            r.add(new Node(v, rcntOld - 1));

```

```

13         lm.merge(v, 1, Integer::sum);
14         rm.merge(v, -1, Integer::sum);
15         int lenL = i + 1, rlenL = n - lenL;
16         //         tr(lm, rm);
17         //         debugNodeTreeSet(l);
18         //         debugNodeTreeSet(r);
19         //         tr("L", a.subList(0, i + 1), l.first().cnt, "lenL", lenL, "R", a.subList(i + 1, n), r.first().cnt);
20         if (l.first().v == r.first().v && dominant(l.first().cnt, lenL) && dominant(r.first().cnt, rlenL)) {
21             boolean atMostOne = true;
22             if (l.size() >= 2) {
23                 if (dominant(findKth(l, 2).cnt, lenL)) {
24                     atMostOne = false;
25                 }
26             }
27             if (r.size() >= 2) {
28                 if (dominant(findKth(r, 2).cnt, rlenL)) {
29                     atMostOne = false;
30                 }
31             }
32             if (atMostOne) return i;
33         }
34     }
35     return -1;
36 }
37
38 boolean dominant(int freq, int m) {
39     return freq * 2 > m;
40 }
41
42 Map<Integer, Integer> counter(List<Integer> a) {
43     Map<Integer, Integer> m = new HashMap<>();
44     for (int x : a) m.merge(x, 1, Integer::sum);
45     return m;
46 }
47
48 Node findKth(TreeSet<Node> ts, int k) {
49     Iterator<Node> it = ts.iterator();
50     int i = 0;
51     Node cur = null;
52     while (it.hasNext() && i < k) {
53         cur = it.next();
54         i++;
55     }
56     return cur;
57 }
58
59 class Node implements Comparable<Node> {
60     int v, cnt;
61
62     Node(int v, int cnt) {
63         this.v = v;
64         this.cnt = cnt;
65     }
66
67     @Override
68     public int compareTo(Node y) {
69         if (cnt != y.cnt) return y.cnt - cnt;
70         return v - y.v; // cannot return 0, some node will missing
71     }
72
73     public String toString() {
74         return this.v + "=" + this.cnt;
75     }
76 }
77 }

```

☐ Custom Testcase

 Submission Result: **Accepted** (/submissions/detail/995614456/) ⓘ

More Details > (/submissions/detail/995614456/)

Share your acceptance!