6431. Neighboring Bitwise XOR

My Submissions (/contest/weekly-contest-345/problems/neighboring-bitwise-xor/submissions/)

Back to Contest (/contest/weekly-contest-345/)

A **0-indexed** array derived with length n is derived by computing the **bitwise XOR** (\odot) of adjacent values in a **binary array** original of length n.

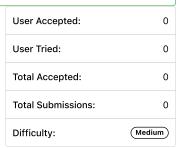
Specifically, for each index i in the range [0, n-1]:

- If i = n 1, then derived[i] = original[i] ⊕ original[0].
- Otherwise, derived[i] = original[i] ⊕ original[i + 1].

Given an array derived, your task is to determine whether there exists a valid binary array original that could have formed derived.

Return true if such an array exists or false otherwise.

• A binary array is an array containing only 0's and 1's



Example 1:

```
Input: derived = [1,1,0]
Output: true
Explanation: A valid original array that gives derived is [0,1,0].
derived[0] = original[0] ** original[1] = 0 ** 1 = 1
derived[1] = original[1] ** original[2] = 1 ** 0 = 1
derived[2] = original[2] ** original[0] = 0 ** 0 = 0
```

Example 2:

```
Input: derived = [1,1]
Output: true
Explanation: A valid original array that gives derived is [0,1].
derived[0] = original[0] ** original[1] = 1
derived[1] = original[1] ** original[0] = 1
```

Example 3:

```
Input: derived = [1,0]
Output: false
Explanation: There is no valid original array that gives derived.
```

Constraints:

- n == derived.length
- 1 <= n <= 10^5
- The values in derived are either 0's or 1's

```
JavaScript
                                                                                                                                  C
    const aeq = (a, b) \Rightarrow a.length === b.length && a.every((v, i) <math>\Rightarrow v === b[i]);
3 ▼
    const doesValidArrayExist = (a) => {
4
        let n = a.length;
 5
        let b = Array(n).fill(-1), b2 = Array(n).fill(-1);
 6
        b[0] = 0;
        b2[0] = 1;
 7
 8
        go(a, b)
9
         if (aeq(test(b), a)) return true;
10
        go(a, b2);
         if (aeq(test(b2), a)) return true;
11
        return false;
12
13
    };
14
15 	 v const qo = (a, b) ⇒ {
16
        let n = a.length;
```

```
17 ▼
         for (let i = 0; i < n; i++) {
             if (i == n - 1) {
b[i] = a[i] ^ b[0];
18 ▼
19
20 ▼
             } else {
21
                  // a[i] = b[i] ^ b[i+1]
22
                  b[i + 1] = a[i] ^ b[i];
23
24
         }
25
    };
26
27 ▼
    const test = (a) \Rightarrow \{
         let n = a.length, res = Array(n).fill(-1);
28
         for (let i = 0; i + 1 < n; i++) res[i] = a[i] \land a[i + 1];
29
         res[n - 1] = a[n - 1] ^ a[0];
30
31
         return res;
32
    };
```

 $\ \square$ Custom Testcase

Use Example Testcases

Submission Result: Accepted (/submissions/detail/949948543/) @

More Details > (/submissions/detail/949948543/)

Share your acceptance!

Copyright © 2023 LeetCode

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)

△ Submit

Run