

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** ( $\oplus$ ) 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]  $\oplus$  original[0]`.
- Otherwise, `derived[i] = original[i]  $\oplus$  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**

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

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]  $\oplus$  original[1] = 0  $\oplus$  1 = 1`  
`derived[1] = original[1]  $\oplus$  original[2] = 1  $\oplus$  0 = 1`  
`derived[2] = original[2]  $\oplus$  original[0] = 0  $\oplus$  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]  $\oplus$  original[1] = 1`  
`derived[1] = original[1]  $\oplus$  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 <= 105`
- The values in `derived` are either **0's** or **1's**

JavaScript

📄

↺

⚙️


```
1 const aeq = (a, b) => a.length === b.length && a.every((v, i) => v === b[i]);
2
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;
7     b2[0] = 1;
8     go(a, b)
9     if (aeq(test(b), a)) return true;
10    go(a, b2);
11    if (aeq(test(b2), a)) return true;
12    return false;
13 };
14
15 ▾ const go = (a, b) => {
16     let n = a.length;
```

```
17 ▾    for (let i = 0; i < n; i++) {
18 ▾        if (i == n - 1) {
19             b[i] = a[i] ^ b[0];
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) => {
28     let n = a.length, res = Array(n).fill(-1);
29     for (let i = 0; i + 1 < n; i++) res[i] = a[i] ^ a[i + 1];
30     res[n - 1] = a[n - 1] ^ a[0];
31     return res;
32 };
```

☐ Custom Testcase[Use Example Testcases](#)[Run](#)[Submit](#)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 \(/region\)](#)