1538. Guess the Majority in a Hidden Array

Description

We have an integer array `nums`, where all the integers in `nums` are **0** or **1**. You will not be given direct access to the array, instead, you will have an **API** `ArrayReader` which have the following functions:

- `int query(int a, int b, int c, int d)`: where `0 <= a < b < c < d < ArrayReader.length()`. The function returns the distribution of the value of the 4 elements and returns:
  - **4**: if the values of the 4 elements are the same (0 or 1).
  - **2**: if three elements have a value equal to 0 and one element has value equal to 1 or vice versa.
  - **0**: if two element have a value equal to 0 and two elements have a value equal to 1.
- `int length()`: Returns the size of the array.

You are allowed to call `query()` **2 * n times** at most where n is equal to `ArrayReader.length()`.

Return **any** index of the most frequent value in `nums`, in case of tie, return -1.

Example 1:

```
Input: nums = [0,0,1,0,1,1,1,1]
Output: 5
Explanation: The following calls to the API
reader.length() // returns 8 because there are 8 elements in the hidden array.
reader.query(0,1,2,3) // returns 2 this is a query that compares the elements nums[0], nums[1], nums[2], nums[3]
// Three elements have a value equal to 0 and one element has value equal to 1 or viceversa.
reader.query(4,5,6,7) // returns 4 because nums[4], nums[5], nums[6], nums[7] have the same value.
we can infer that the most frequent value is found in the last 4 elements.
Index 2, 4, 6, 7 is also a correct answer.
```

Example 2:

```
Input: nums = [0,0,1,1,0]
Output: 0
```

Example 3:

```
Input: nums = [1,0,1,0,1,0,1,0]
Output: -1
```

Constraints:

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

Follow up: What is the minimum number of calls needed to find the majority element?