3354. Make Array Elements Equal to Zero

Easy | Topics | Companies | Hint

You are given an integer array `nums`.

Start by selecting a starting position `curr` such that `nums[curr] == 0`, and choose a movement **direction** of either left or right.

After that, you repeat the following process:

- If `curr` is out of the range `[0, n - 1]`, this process ends.
- If `nums[curr] == 0`, move in the current direction by **incrementing** `curr` if you are moving right, or **decrementing** `curr` if you are moving left.
- Else if `nums[curr] > 0`:
  - Decrement `nums[curr]` by 1.
  - **Reverse** your movement direction (left becomes right and vice versa).
  - Take a step in your new direction.

A selection of the initial position `curr` and movement direction is considered **valid** if every element in `nums` becomes 0 by the end of the process.

Return the number of possible **valid** selections.

Example 1:

> **Input:** `nums = [1,0,2,0,3]`
>
> **Output:** 2
>
> **Explanation:**
>
> The only possible valid selections are the following:

- Choose `curr = 3`, and a movement direction to the left.
  - `[1,0,2,0,3] -> [1,0,2,0,3] -> [1,0,1,0,3] -> [1,0,1,0,3] -> [1,0,1,0,2] -> [1,0,1,0,2] -> [1,0,0,0,2] -> [1,0,0,0,2] -> [1,0,0,0,1] -> [1,0,0,0,1] -> [1,0,0,0,1] -> [1,0,0,0,1] -> [0,0,0,0,1] -> [0,0,0,0,1] -> [0,0,0,0,1] -> [0,0,0,0,0].`
- Choose `curr = 3`, and a movement direction to the right.
  - `[1,0,2,0,3] -> [1,0,2,0,3] -> [1,0,2,0,2] -> [1,0,2,0,2] -> [1,0,1,0,2] -> [1,0,1,0,2] -> [1,0,1,0,1] -> [1,0,1,0,1] -> [1,0,0,0,1] -> [1,0,0,0,1] -> [1,0,0,0,0] -> [1,0,0,0,0] -> [1,0,0,0,0] -> [1,0,0,0,0] -> [0,0,0,0,0].`

Example 2:

> **Input:** `nums = [2,3,4,0,4,1,0]`
>
> **Output:** `0`
>
> **Explanation:**
>
> There are no possible valid selections.

Constraints:

- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 100`
- There is at least one element `i` where `nums[i] == 0`.

Python:

```python
class Solution:
    def countValidSelections(self, nums: List[int]) -> int:
        n, res = len(nums), 0
        left, right = [0 for _ in range(n)], [0 for _ in range(n)]
        for i in range(1, n):
            left[i] = left[i - 1] + nums[i - 1]
            right[-i - 1] = right[-i] + nums[-i]
        for i, num in enumerate(nums):
            if num != 0: continue
            if left[i] == right[i]: res += 2
            if abs(left[i] - right[i]) == 1: res += 1
        return res
```

JavaScript:

```javascript
const countValidSelections = nums => {
    const n = nums.length;
    let res = 0;
    const left = new Array(n).fill(0);
    const right = new Array(n).fill(0);

    for (let i = 1; i < n; i++) {
        left[i] = left[i - 1] + nums[i - 1];
        right[n - i - 1] = right[n - i] + nums[n - i];
    }

    for (let i = 0; i < n; i++) {
        if (nums[i] !== 0) continue;
        if (left[i] === right[i]) res += 2;
        else if (Math.abs(left[i] - right[i]) === 1) res += 1;
    }

    return res;
};
```

Java:

```java
class Solution {
    public int countValidSelections(int[] nums) {
        int n = nums.length;
        int res = 0;
        int[] left = new int[n];
        int[] right = new int[n];
```

```
        for (int i = 1; i < n; ++i) {
            left[i] = left[i - 1] + nums[i - 1];
            right[n - i - 1] = right[n - i] + nums[n - i];
        }

        for (int i = 0; i < n; ++i) {
            if (nums[i] != 0) {
                continue;
            }
            if (left[i] == right[i]) {
                res += 2;
            }
            if (Math.abs(left[i] - right[i]) == 1) {
                res += 1;
            }
        }
        return res;
    }
}
```