<https://leetcode.com/problems/find-pivot-index/description/>

Given an array of integers nums, calculate the **pivot index** of this array.

The **pivot index** is the index where the sum of all the numbers **strictly** to the left of the index is equal to the sum of all the numbers **strictly** to the index's right.

If the index is on the left edge of the array, then the left sum is 0 because there are no elements to the left. This also applies to the right edge of the array.

Return **the leftmost pivot index**. If no such index exists, return -1.

**Example 1:**

**Input:** nums = [1,7,3,6,5,6]

**Output:** 3

**Explanation:**

The pivot index is 3.

Left sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11

Right sum = nums[4] + nums[5] = 5 + 6 = 11

**Example 2:**

**Input:** nums = [1,2,3]

**Output:** -1

**Explanation:**

There is no index that satisfies the conditions in the problem statement.

**Example 3:**

**Input:** nums = [2,1,-1]

**Output:** 0

**Explanation:**

The pivot index is 0.Left sum = 0 (no elements to the left of index 0)Right sum = nums[1] + nums[2] = 1 + -1 = 0

**Constraints:**

1 <= nums.length <= 10^4

-1000 <= nums[i] <= 1000

**Note:** This question is the same as 1991: <https://leetcode.com/problems/find-the-middle-index-in-array/>

**Attempt 1: 2024-05-04**

**Wrong Solution**

**we should not use sum / 2 == ... because / 2 will encounter issue when sum is odd number**

**Below still failed at [-1,-1,-1,-1,0,0], output 4, expected -1.**

class Solution {

public int pivotIndex(int[] nums) {

int sum = 0;

for(int n : nums) {

sum += n;

}

if(sum == nums[0]) {

return 0;

}

for(int i = 0; i < nums.length; i++) {

int half = 0;

for(int j = 0; j < i; j++) {

half += nums[j];

// (sum - nums[i]) % 2 == 0 test out by [-1,-1,-1,-1,-1,-1]

if(i != nums.length - 1 && (sum - nums[i]) % 2 == 0 && half == (sum - nums[i]) / 2) {

return i;

}

}

}

return -1;

}

}

**Solution 1: Brute Force (10 min)**

class Solution {

public int pivotIndex(int[] nums) {

int totalSum = 0; // Total sum of all array elements

for (int num : nums) {

totalSum += num;

}

int leftSum = 0; // Running sum of elements to the left of the current index

for (int i = 0; i < nums.length; i++) {

// Calculate the right sum as: totalSum - leftSum - nums[i]

if (leftSum == totalSum - leftSum - nums[i]) {

return i; // Found the pivot index

}

leftSum += nums[i]; // Update left sum for the next index

}

return -1; // No pivot index found

}

}

Time Complexity: O(n)

Space Complexity: O(1)