# 题目

给你一个下标从0开始的整数数组 nums ，请你找到最左边的中间位置middleIndex（也就是所有可能中间位置下标最小的一个）。

中间位置 middleIndex 是满足 nums[0] + nums[1] + ... + nums[middleIndex-1] == nums[middleIndex+1] + nums[middleIndex+2] + ... + nums[nums.length-1] 的数组下标。

如果 middleIndex == 0 ，左边部分的和定义为 0 。类似的，如果 middleIndex == nums.length - 1 ，右边部分的和定义为 0 。

请你返回满足上述条件 最左边 的 middleIndex ，如果不存在这样的中间位置，请你返回 -1 。

示例 1：

输入：nums = [2,3,-1,8,4]

输出：3

解释：

下标 3 之前的数字和为：2 + 3 + -1 = 4

下标 3 之后的数字和为：4 = 4

示例 2：

输入：nums = [1,-1,4]

输出：2

解释：

下标 2 之前的数字和为：1 + -1 = 0

下标 2 之后的数字和为：0

示例 3：

输入：nums = [2,5]

输出：-1

解释：

不存在符合要求的 middleIndex 。

示例 4：

输入：nums = [1]

输出：0

解释：

下标 0 之前的数字和为：0

下标 0 之后的数字和为：0

提示：

1 <= nums.length <= 100

-1000 <= nums[i] <= 1000

注意：本题与主站 724 题相同：https://leetcode-cn.com/problems/find-pivot-index/

# 分析

## 方法一：前缀和

**思路：**

记数组的全部元素之和为total，当遍历到第i个元素时，设其左侧元素之和为sum，则其右侧元素之和为total−numsi-sum。左右侧元素相等即为sum=total−numsi-sum，即2×sum+numsi=total。

当中心索引左侧或右侧没有元素时，即为零个项相加，这在数学上称作「空和」（empty sum）。在程序设计中我们约定「空和是零」。

**代码：**

class Solution {

public:

int findMiddleIndex(vector<int>& nums) {

int total = accumulate(nums.begin(), nums.end(), 0);

int sum = 0;

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

if (2 \* sum + nums[i] == total) {

return i;

}

sum += nums[i];

}

return -1;

}

};

**复杂度分析：**

时间复杂度：O(n)，其中n为数组的长度。

空间复杂度：O(1)。