# 题目

给你一个整数数组nums，请计算数组的中心下标。

数组中心下标是数组的一个下标，其左侧所有元素相加的和等于右侧所有元素相加的和。

如果中心下标位于数组最左端，那么左侧数之和视为0，因为在下标的左侧不存在元素。这一点对于中心下标位于数组最右端同样适用。

如果数组有多个中心下标，应该返回 最靠近左边的那一个。如果数组不存在中心下标，返回-1。

示例 1：

输入：nums = [1, 7, 3, 6, 5, 6]

输出：3

解释：

中心下标是 3 。

左侧数之和 sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11 ，

右侧数之和 sum = nums[4] + nums[5] = 5 + 6 = 11 ，二者相等。

示例 2：

输入：nums = [1, 2, 3]

输出：-1

解释：

数组中不存在满足此条件的中心下标。

示例 3：

输入：nums = [2, 1, -1]

输出：0

解释：

中心下标是 0 。

左侧数之和 sum = 0 ，（下标 0 左侧不存在元素），

右侧数之和 sum = nums[1] + nums[2] = 1 + -1 = 0 。

提示：

1 <= nums.length <= 104

-1000 <= nums[i] <= 1000

注意：本题与主站 1991 题相同：https://leetcode-cn.com/problems/find-the-middle-index-in-array/

# 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 pivotIndex(vector<int>& nums) {

int total = 0;

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

total += nums.at(i);

}

int sum = 0;

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

if (sum == (total - sum - nums.at(i))) {

return i;

}

sum += nums.at(i);

}

return -1;

}

};

或：

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) {

// 或：if (sum = total – nums[i] - sum)

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)。