# 题目

给你一个长度为 n 的整数数组 nums ，返回使所有数组元素相等需要的最小操作数。

在一次操作中，你可以使数组中的一个元素加 1 或者减 1 。

示例 1：

输入：nums = [1,2,3]

输出：2

解释：

只需要两次操作（每次操作指南使一个元素加 1 或减 1）：

[1,2,3] => [2,2,3] => [2,2,2]

示例 2：

输入：nums = [1,10,2,9]

输出：16

提示：

n == nums.length

1 <= nums.length <= 10^5

-10^9 <= nums[i] <= 10^9

# 分析

## 方法一：排序

思路：

为了找到使所有数组元素相等所需的最小操作数，我们可以先将数组排序，这样所有元素都会按升序排列。排序后，数组中的中位数（或对于偶数个元素的数组，中间两个数的平均值）就是使所有元素相等所需的目标值。

为什么选择中位数作为目标值呢？因为中位数是将数组分为两个相等（或几乎相等）部分的数，因此从中位数出发，到数组中每个元素的距离之和（即操作数）会是最小的。

接下来，我们遍历排序后的数组，计算每个元素到中位数的距离（绝对值），并将这些距离累加起来，就得到了所需的最小操作数。

代码：

class Solution {

public:

int minMoves2(vector<int>& nums) {

int n = nums.size();

// 先对数组进行排序

sort(nums.begin(), nums.end());

// 找到中位数

int median = nums[n / 2];

// 计算每个元素到中位数的距离之和

int moves = 0;

for (int num : nums) {

moves += abs(num - median);

}

return moves;

}

};

在这个代码中，我们首先使用sort函数对nums数组进行排序。然后，我们找到中位数median，它是排序后数组中间的元素（对于奇数个元素的数组）或者是中间两个元素的平均值（对于偶数个元素的数组，但在这个问题中我们只需要其中一个中间元素作为目标值即可）。

接下来，我们遍历排序后的数组，对每个元素num，计算它到中位数median的绝对值距离，并将这些距离累加到moves变量中。最后，返回moves作为结果，即所需的最小操作数。

注意：在这个问题中，我们实际上并没有用到中间两个数的平均值，因为无论选择哪一个作为目标值，到另一个中间数的距离都会被计算两次（一次正一次负），相互抵消，所以最终的结果是一样的。因此，我们只需要简单地选择中间的一个数作为目标值即可。