<https://leetcode.com/problems/minimum-moves-to-equal-array-elements/description/>

Given an integer array nums of size n, return *the minimum number of moves required to make all array elements equal*.

In one move, you can increment n - 1 elements of the array by 1.

**Example 1:**

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

**Output:** 3

**Explanation:** Only three moves are needed (remember each move increments two elements):

[1,2,3] => [2,3,3] => [3,4,3] => [4,4,4]

**Example 2:**

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

**Output:** 0

**Constraints:**

n == nums.length

1 <= nums.length <= 105

-109 <= nums[i] <= 109

The answer is guaranteed to fit in a **32-bit** integer.

**Attempt 1: 2025-04-21**

**Solution 1: Math (10min)**

class Solution {

    public int minMoves(int[] nums) {

        int min = Integer.MAX\_VALUE;

        int sum = 0;

        for(int num : nums) {

            sum += num;

            min = Math.min(min, num);

        }

        return sum - min \* nums.length;

    }

}

Time Compelxity: O(n)

Space Compelxity: O(1)

**Refer to Deepseek**

**Problem Understanding**

The problem requires us to find the minimum number of moves needed to make all array elements equal, where a move is defined as incrementing n-1 elements by 1.

**Approach**

The key insight is recognizing that:

**Incrementing n-1 elements by 1 is equivalent to decrementing 1 element by 1 (in terms of relative difference between elements)**

**We need to make all elements equal to the smallest element in the array**

**The number of moves is the sum of differences between each element and the smallest element**

**Solution Code**

public class Solution {

public int minMoves(int[] nums) {

if (nums == null || nums.length == 0) {

return 0;

}

int min = Integer.MAX\_VALUE;

// Find the minimum element in the array

for (int num : nums) {

if (num < min) {

min = num;

}

}

int moves = 0;

// Calculate the sum of differences between each element and the minimum

for (int num : nums) {

moves += num - min;

}

return moves;

}

}

**Explanation**

First, we find the smallest element in the array. This will be the target value that all other elements need to reach.

Then, for each element in the array, we calculate how much it needs to be reduced (which is equivalent to the number of moves needed for that element to reach the minimum value).

The sum of all these differences gives us the total number of moves required, since each move effectively reduces one element's difference from the minimum by 1.

This approach efficiently solves the problem in O(n) time with O(1) space complexity, making it optimal.

**Refer to**

[L462.Minimum Moves to Equal Array Elements II](note://WEBf26377967abd88cf360cde7f55549d27)

[L2137.Pour Water Between Buckets to Make Water Levels Equal (Ref.L462)](note://WEBdffa45243572b5365e52996a594326fb)

[L2491.Divide Players Into Teams of Equal Skill](note://WEB45f3a02d945beba7abe9d13b138ce982)