1. 📘 Problem Statement

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

In one move, you can increment or decrement a single element by 1.

Constraints:

- $1 <= nums.length <= 10^5$
- $-10^9 <= nums[i] <= 10^9$

2. 💡 Intuition

To make all elements equal with the fewest number of increment/decrement operations, we want to minimize the total sum of differences. This naturally leads us to the median, which is the statistical middle value of a sorted list.

3. 🔑 Key Observations

- The sum of absolute differences $|xi - m|$ is minimized when m is the median.
- Changing all values to the median will require fewer operations than changing to the mean or any arbitrary value.
- This approach works for both even and odd length arrays.

4. □ Approach

- Sort the array to easily find the median.
- Choose the median element.
- Compute the sum of absolute differences between each element and the median.
- Return this sum as the minimum number of moves.

5. ⚠ Edge Cases

- If the array has only one element, no move is needed.
- If all elements are already equal, result is 0.
- Works for negative, zero, or large positive integers as long as within range.

6. 📊 Complexity Analysis

 Time Complexity:

- Sorting takes O(n log n).
- Calculating the sum of differences takes O(n).
- Total: O(n log n)

 Space Complexity:

- O(1) extra space (if sorting in place), otherwise O(n) if a new list is used.

7. 🔁 Alternative Approaches

a. Brute Force:

- Try making all elements equal to every possible number in the array.
- Time complexity: $O(n^2)$ → inefficient for large input.

b. QuickSelect for Median (Optimized):

- Use the QuickSelect algorithm to find the median in O(n) average time.
- Total complexity becomes O(n) on average.
- Useful if performance is critical for large inputs.

8. □ **Test Cases**

✅ Test Case 1:

Input: [1, 2, 3]

Output: 2

Explanation: [1,2,3] → [2,2,3] → [2,2,2]

✅ Test Case 2:

Input: [1, 10, 2, 9]

Output: 16

Explanation: All elements moved to 2 or 9 → sum(|nums[i] - median|)

✅ Test Case 3:

Input: [5]

Output: 0

Explanation: Only one element, no moves needed.

✅ Test Case 4:

Input: [-1, 0, 1]

Output: 2

Explanation: Median is 0. Total moves: $|-1-0| + |0-0| + |1-0| = 1 + 0 + 1 = 2$

9. □ **Final Thoughts**

- This problem beautifully demonstrates how median minimizes the sum of absolute differences.
- Sorting provides a straightforward and efficient solution.
- While there's an optimized linear-time version using QuickSelect, the sorted version is often sufficient for coding interviews and real-world use cases.