Super Washing Machines – Full Documentation

1. Problem Statement

You have n super washing machines arranged in a line. Each machine contains a certain number of dresses or may be empty.

Each move allows you to select any number of machines, and each selected machine can send only one dress to one adjacent machine (left or right) simultaneously.

You must find the minimum number of moves needed so that every machine ends up with the same numberof dresses.
If it's not possible, return -1.

2. Intuition

Balancing the dresses across machines is like balancing weight across scales. If we keep track of how much each machine differs from the target number of dresses and track how imbalance flows through the line, we can find the number of steps needed.

3. Key Observations

- If the total number of dresses is not divisible by n, it's impossible to evenly distribute them.
- The problem is not just about individual surplus/deficit, but also how imbalance propagates from one side to the other.
- We track the cumulative imbalance and compare it with the current difference to find the maximum move needed.

4. Approach

- Compute total dresses and check if they can be equally divided among n machines.
- Set a target: Each machine should have total // n dresses.

- Iterate over each machine and:
    - Calculate the difference between current dresses and target.
    - Maintain a running imbalance sum.
    - The max of absolute imbalance and the current difference gives the minimum number of moves required up to that point.
- Return the maximum of all such values.

5. Edge Cases

Case	Explanation
machines = [1,0,5]	Valid input, returns 3
machines = [0,2,0]	Total not divisible by 3 → return -1
machines = [1,1,1]	Already balanced, should return 0
machines = [10, 0, 0, 0, 0]	Needs multiple moves as one machine carries most of the load
machines = [0]	Only one machine, already balanced → return 0

6. Complexity Analysis

 Time Complexity

- O(n) – Single pass through the machines list.

 Space Complexity

- O(1) – Constant space; only variables are used for running totals.

7. Alternative Approaches

Approach	Notes
Brute Force BFS	Simulate each move, very slow (O(exponential)), impractical for n > 20
Segment Transfer	Try balancing using two-pointers or prefix sums, more complex
Greedy/Flow Model	The current approach is optimized and widely accepted

8. Test Cases

Input	Output	Explanation
[1, 0, 5]	3	Balance achieved in 3 moves
[0, 3, 0]	2	Distribute 3 across adjacent machines
[0, 2, 0]	-1	Total = 2, not divisible by 3
[10, 0, 0, 0, 0]	8	Dresses slowly pass from one end to the other
[1,1,1]	0	Already balanced

9. Final Thoughts

- The key insight is to model the dress distribution problem as a flow of imbalance.
- Instead of simulating moves, we rely on mathematical tracking of balance using greedy techniques.
- This is an excellent example of how tracking local and global state leads to optimal solutions.