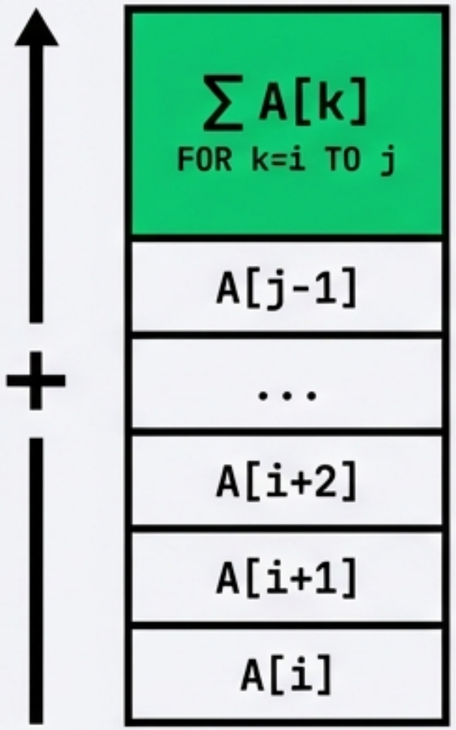
The Tale of Two Subarrays

Mastering Maximum Subarray & Maximum Product Subarray

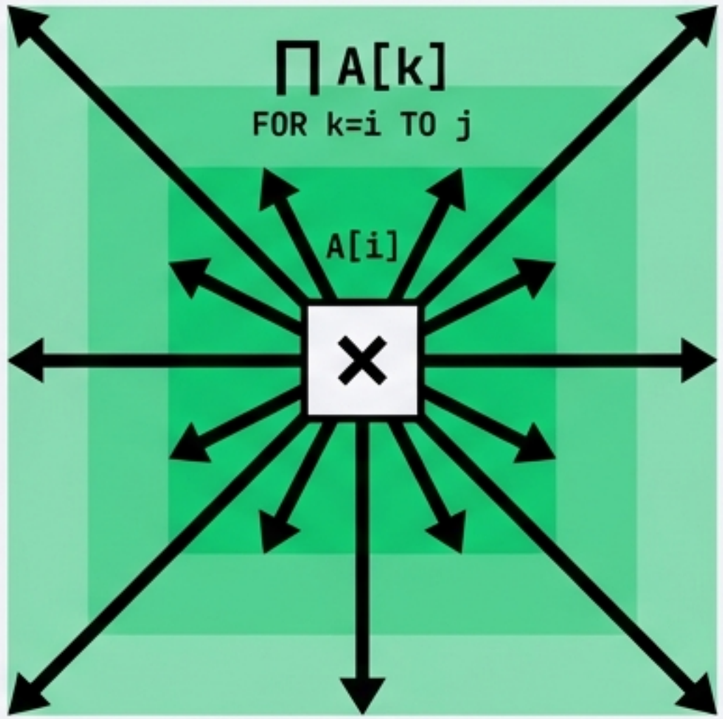
SUM



Linear accumulation. Adding elements one by one builds a total. The value grows incrementally.

Example: [-2, 1, -3, 4, -1, 2, 1, -5, 4] → Max Sum = 6.

PRODUCT



Exponential magnification. Multiplying elements causes rapid expansion. Positive and negative values significantly alter the result.

Example: [2, 3, -2, 4] → Max Product = 6. (Note: Negatives can turn products positive, e.g., [-2, 0, -1] → Max Product = 0 or [2, 3, -2, 4, -1] → Max Product = 48).

Same Input, Different Goals

Problem A: Max Sum

$$2 + 3 + (-2) + 4 = 7$$

Cumulative value. We tolerate the **dip (-2)** because the **recovery (4) results in a net gain.**

Problem B: Max Product

$$2 \times 3 = 6$$

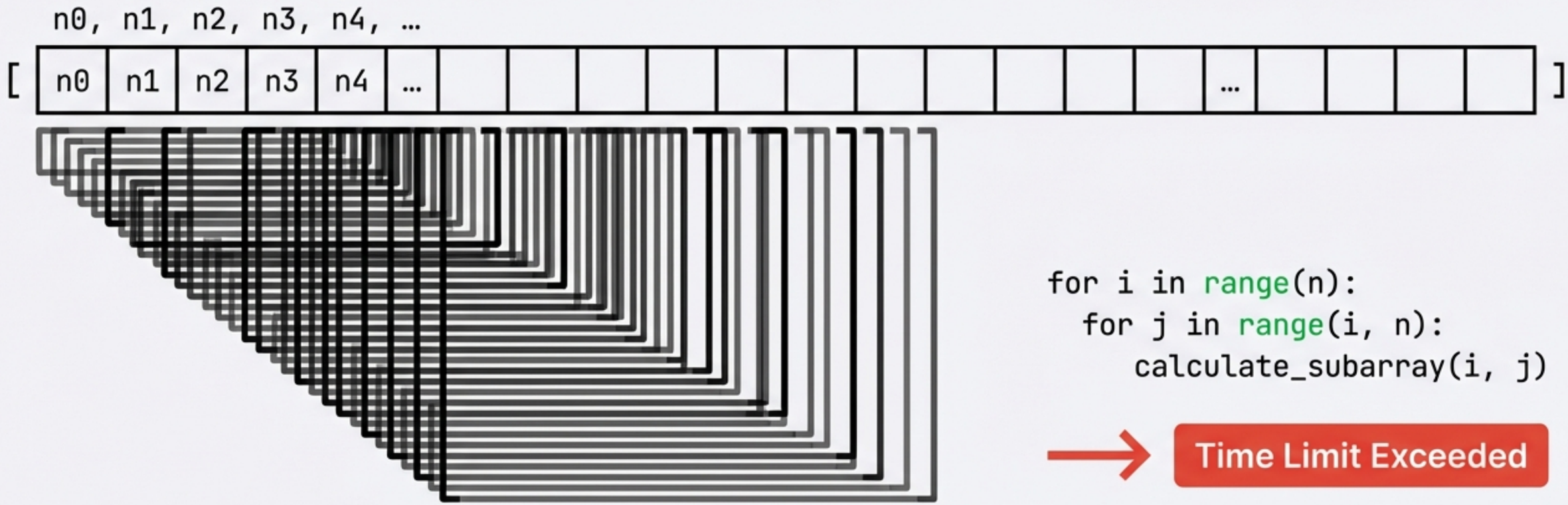
$$6 \times -2 = -12$$

$$-12 \times 4 = -48$$

Magnitude & Sign. A single negative flips the direction immediately. The dip is **fatal** here.

The Naive Solution

Brute Force Approach ($O(N^2)$)

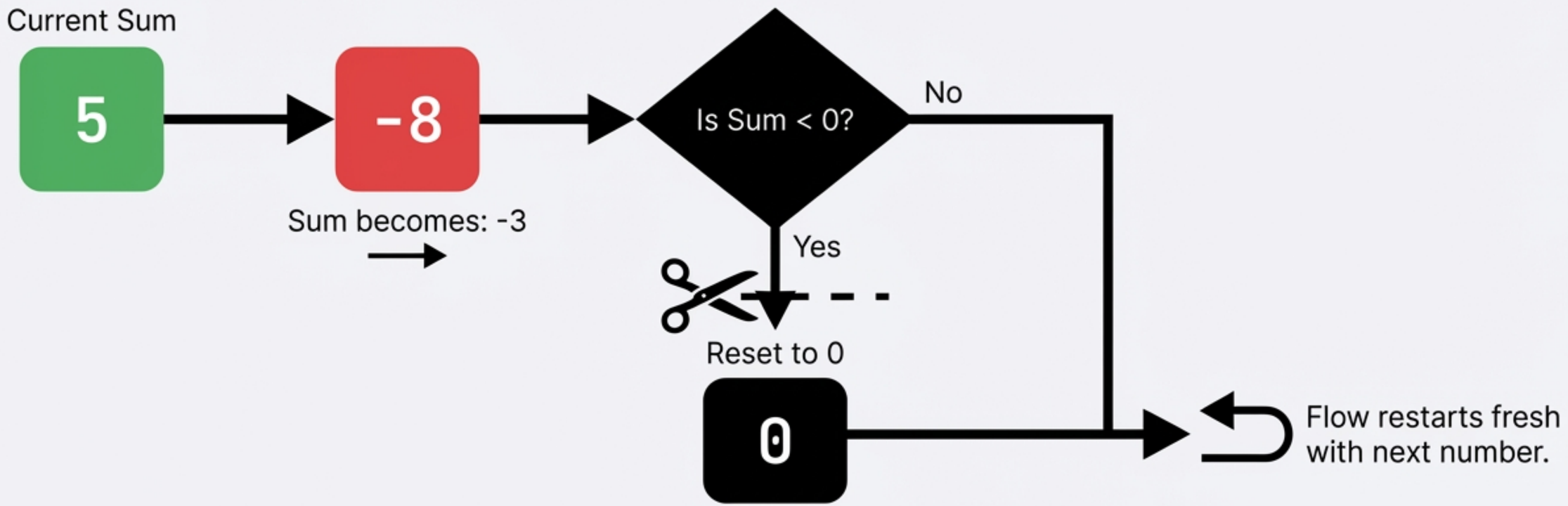



```
for i in range(n):
  for j in range(i, n):
    calculate_subarray(i, j)
```

→ Time Limit Exceeded

Redundant calculations. As N grows, the workload explodes quadratically.

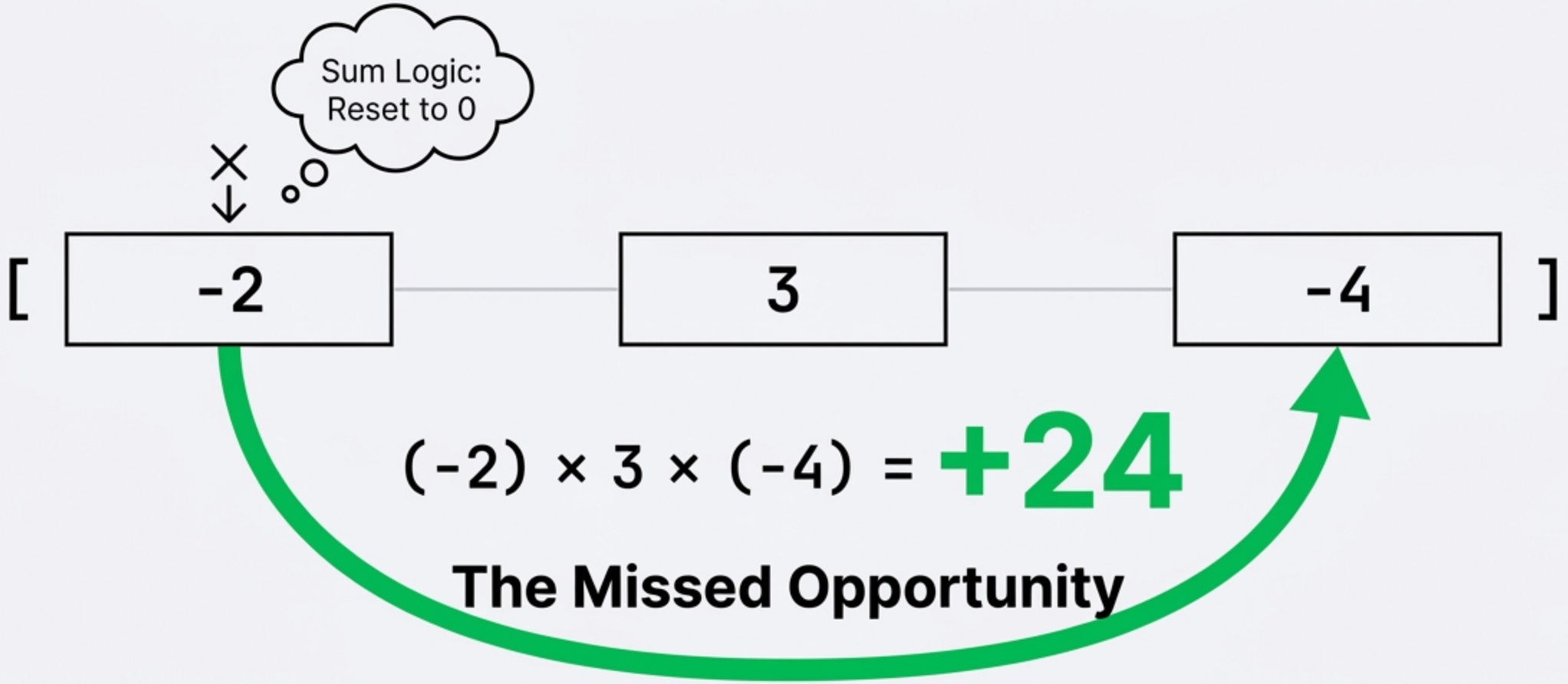
Optimising ‘Sum’: Discarding the Baggage Kadane’s Algorithm (*O*(*N*))




```
current_sum = max(num, current_sum + num)
```

Insight: A negative prefix contributes nothing. Cut the loss and reset.

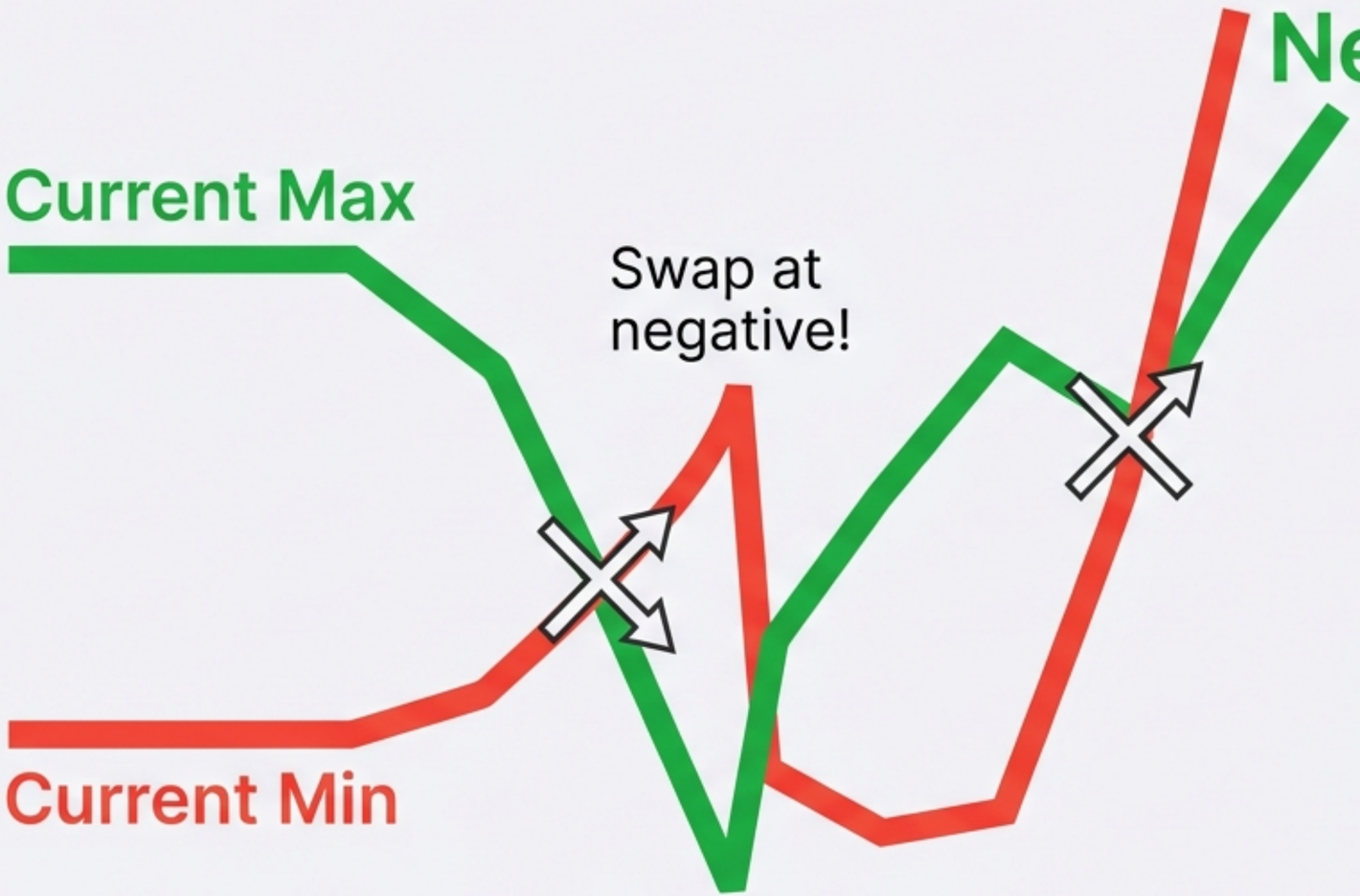
Why Sum Logic Fails for Products



The Twist: In multiplication, a negative number isn't just "baggage". It is potential energy. A future negative number can flip this "bad" value into a "maximum" value.

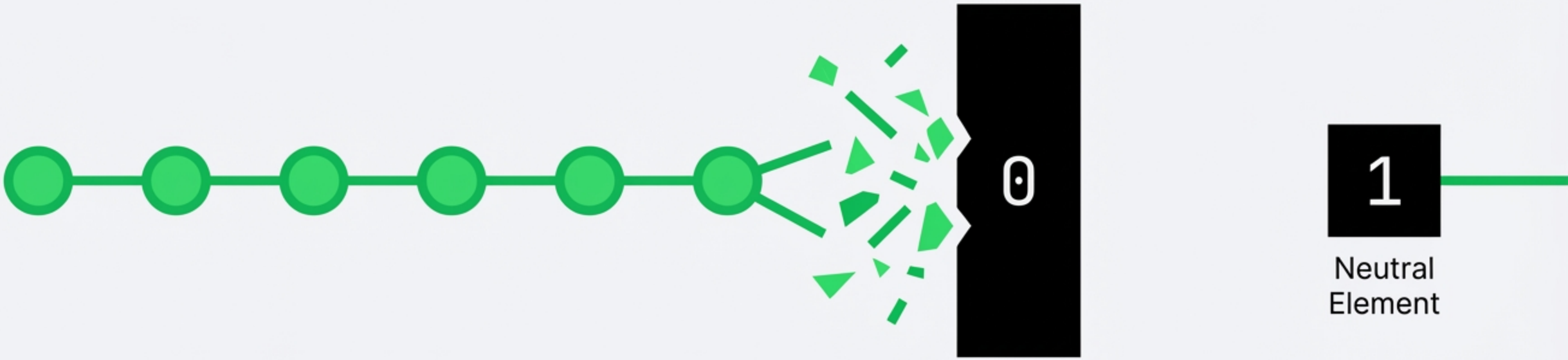
Maximum Product: The Two-Pointer Variation

Tracking Capability, Not Just Value




```
temp_max =  max(n, n * max, n * min)
current_min = min(n, n * max, n * min)
max = temp_max
```

The Streak Killer: Handling Zeros



The Problem: Multiplication by zero collapses all history to `0`.

The Solution: Hard Reset. If `num == 0`, reset `Current Max` and `Current Min` to `1`.

Logic Comparison at a Glance

Max Subarray (Sum)		Max Product Subarray	
Track:	1 Variable: Current Sum	Track:	2 Variables: Current Max & Current Min
Strategy:	Discard negatives immediately.	Strategy:	Store negatives (swapping potential).
Logic:	`if current < 0: current = 0`	Logic:	`if negative: swap(max, min).` `if 0: reset to 1`
Complexity:	Time: O(N) | Space: O(1)	Complexity:	Time: O(N) | Space: O(1)

Both problems optimise from $O(N^2)$ to O(N) using single-pass iteration.