The Shifting Sands of Time

Story

In the heart of the Infinite Desert, there lived a Guardian who protected the Time Orb. The Orb's power came from the perfect ordering of the countless grains of sand within it, based on their **Age**. When this order was disturbed over time, the Guardian's duty was to re-sort the grains. The Guardian performs an ancient ritual: starting from the second grain of sand in the pile, they take each grain into their palm one by one.

The Guardian drops the grain in their palm (the 'key') into the already sorted pile of sand that precedes it. The grain trickles down through the grains that are older (have a greater **Age**) than itself and settles into its correct place. However, during this process, the young grain in the palm interacts with every older grain it passes and displaces, creating a "temporal fluctuation." The magnitude of the fluctuation created is equal to the **difference between the Ages** of the two grains.

The Guardian's task is to calculate the total temporal fluctuation that spreads into the universe once all the grains have settled.

Task (Problem Definition)

You are given a single initial array of N integers, representing the 'Age' of each grain. Your goal is to sort this array in ascending order of 'Age' values using the **in-place Insertion Sort** algorithm.

During the algorithm, when placing a currently considered grain ('key') into its correct position within the sorted subarray, every older grain (with a larger 'Age' value) is **shifted one position to the right**. Each shift operation generates a cost. This cost is the difference between the **value of the 'key' grain** and the **value of the grain being shifted** at that moment (shifted_value − key_value). The total cost is the sum of all costs incurred during these operations.

- **Required output**
  - The **Total Temporal Fluctuation** generated when the sorting process is complete.

Constraints

- $1 \leq N \leq 5,000$
- $1 \leq \text{Age}_i \leq 10^9$ (integer)

Example Scenario

Scenario 1 $N = 4$
Initial Array: $[20, 5, 15, 8]$
Expected Total Fluctuation: `39`.
Explanation (Insertion Sort Steps):

1. **Initial:** 'Array = [20, 5, 15, 8]', 'Cost = 0'.

2. **i=1: Processing ‘key = 5‘ grain.**
   - The age of ‘Array[0]‘ (20) > age of ‘key‘ (5). Space needs to be made.
   - The ‘20‘ grain is **shifted to the right**. Cost: ‘0 + (20 - 5) = 15‘.
   - ‘key‘ is placed in the created space.
   - Result: ‘Array = [5, 20, 15, 8]‘.
3. **i=2: Processing ‘key = 15‘ grain.**
   - The age of ‘Array[1]‘ (20) > age of ‘key‘ (15). Space needs to be made.
   - The ‘20‘ grain is **shifted to the right**. Cost: ‘15 + (20 - 15) = 20‘.
   - The age of ‘Array[0]‘ (5) < age of ‘key‘ (15). Scan stops.
   - ‘key‘ is placed in the created space.
   - Result: ‘Array = [5, 15, 20, 8]‘.
4. **i=3: Processing ‘key = 8‘ grain.**
   - The age of ‘Array[2]‘ (20) > age of ‘key‘ (8). Space needs to be made.
   - The ‘20‘ grain is **shifted to the right**. Cost: ‘20 + (20 - 8) = 32‘.
   - The age of ‘Array[1]‘ (15) > age of ‘key‘ (8). Space needs to be made.
   - The ‘15‘ grain is **shifted to the right**. Cost: ‘32 + (15 - 8) = 39‘.
   - The age of ‘Array[0]‘ (5) < age of ‘key‘ (8). Scan stops.
   - ‘key‘ is placed in the created space.
   - Result: ‘Array = [5, 8, 15, 20]‘.
5. **Sorting complete. Final Total Cost: 39**.