346. Moving Average from Data Stream

Easy 1066 104 Add to List Share

Given a stream of integers and a window size, calculate the moving average of all integers in the sliding window.

Implement the `MovingAverage` class:

- `MovingAverage(int size)` Initializes the object with the size of the window `size`.
- `double next(int val)` Returns the moving average of the last `size` values of the stream.

Example 1:

```
Input
["MovingAverage", "next", "next", "next", "next"]
[[3], [1], [10], [3], [5]]
Output
[null, 1.0, 5.5, 4.66667, 6.0]

Explanation
MovingAverage movingAverage = new MovingAverage(3);
movingAverage.next(1); // return 1.0 = 1 / 1
movingAverage.next(10); // return 5.5 = (1 + 10) / 2
movingAverage.next(3); // return 4.66667 = (1 + 10 + 3) / 3
movingAverage.next(5); // return 6.0 = (10 + 3 + 5) / 3
```

Constraints:

- $1 <= size <= 1000$
- $-10^5 <= val <= 10^5$
- At most 10^4 calls will be made to `next`.

Accepted 216,656 | Submissions 288,693

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 15 Spotify | 7 Amazon | 5 Microsoft | 2

Related Topics

Array Design Queue Data Stream

Java Autocomplete

```java
class MovingAverage {
  int size, windowSum = 0, count = 0;
  Deque queue = new ArrayDeque<Integer>();

  public MovingAverage(int size) {
    this.size = size;
  }

  public double next(int val) {
    ++count;
    // calculate the new sum by shifting the window
    queue.add(val);
    int tail = count > size ? (int)queue.poll() : 0;

    windowSum = windowSum - tail + val;

    return windowSum * 1.0 / Math.min(size, count);
  }
}
```