Moving Average from Data Stream

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

For example,

```
MovingAverage m = new MovingAverage(3);
m.next(1) = 1
m.next(10) = (1 + 10) / 2
m.next(3) = (1 + 10 + 3) / 3
m.next(5) = (10 + 3 + 5) / 3
```

Solution 1

```
import collections

class MovingAverage(object):

    def __init__(self, size):
        """
        Initialize your data structure here.
        :type size: int
        """
        self.queue = collections.deque(maxlen=size)


    def next(self, val):
        """
        :type val: int
        :rtype: float
        """
        queue = self.queue
        queue.append(val)
        return float(sum(queue))/len(queue)



# Your MovingAverage object will be instantiated and called as such:
# obj = MovingAverage(size)
# param_1 = obj.next(val)
```

written by microleo720 original link here

Solution 2

```
public class MovingAverage {

    Deque<Integer> dq;
    int size;
    int sum;
    public MovingAverage(int size) {
        dq = new LinkedList<>();
        this.size = size;
        this.sum = 0;
    }

    public double next(int val) {
        if (dq.size() < size) {
            sum += val;
            dq.addLast(val);
            return (double) (sum / dq.size());
        } else {
            int temp = dq.pollFirst();
            sum -= temp;
            dq.addLast(val);
            sum += val;
            return (double) (sum / size);
        }
    }

}
```

written by publicstatic2 original link here

Solution 3

The idea is to keep the sum so far and update the sum just by replacing the oldest number with the new entry.

```
public class MovingAverage {
    private int [] window;
    private int n, insert;
    private long sum;

    /** Initialize your data structure here. */
    public MovingAverage(int size) {
        window = new int[size];
        insert = 0;
        sum = 0;
    }

    public double next(int val) {
        if (n < window.length)  n++;
        sum -= window[insert];
        sum += val;
        window[insert] = val;
        insert = (insert + 1) % window.length;

        return (double)sum / n;
    }
}
```

written by sculd original link here

From Leetcoder.