

## **295. Find Median from Data Stream**

The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value, and the median is the mean of the two middle values.

- For example, for arr = [2,3,4], the median is 3.
- For example, for arr = [2,3], the median is  $(2 + 3) / 2 = 2.5$ .

*Implement the MedianFinder class:*

- MedianFinder() initializes the MedianFinder object.
- void addNum(int num) adds the integer num from the data stream to the data structure.
- double findMedian() returns the median of all elements so far. Answers within 10<sup>-5</sup> of the actual answer will be accepted.

### **Example 1:**

#### **Input**

- ["MedianFinder", "addNum", "addNum", "findMedian", "addNum", "findMedian"]
- [[], [1], [2], [], [3], []]

#### **Output**

- [null, null, null, 1.5, null, 2.0]

#### **Explanation**

- `MedianFinder medianFinder = new MedianFinder();`
- `medianFinder.addNum(1);` // arr = [1]
- `medianFinder.addNum(2);` // arr = [1, 2]
- `medianFinder.findMedian();` // return 1.5 (i.e.,  $(1 + 2) / 2$ )
- `medianFinder.addNum(3);` // arr[1, 2, 3]
- `medianFinder.findMedian();` // return 2.0

### **Constraints:**

- $-10^5 \leq \text{num} \leq 10^5$
- There will be at least one element in the data structure before calling findMedian.
- At most  $5 * 10^4$  calls will be made to addNum and findMedian.

**Follow up:**

- If all integer numbers from the stream are in the range  $[0, 100]$ , how would you optimize your solution?
- If 99% of all integer numbers from the stream are in the range  $[0, 100]$ , how would you optimize your solution?