LeetCode 第 295 号问题：数据流的中位数

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步博客：https://www.algomooc.com*

题目来源于 LeetCode 上第 295 号问题：数据流的中位数。难度级别为 Hard，目前通过率为 33.5% 。

题目描述

中位数是有序列表中间的数。如果列表长度是偶数，中位数则是中间两个数的平均值。

例如，

[2,3,4] 的中位数是 3

[2,3] 的中位数是 (2 + 3) / 2 = 2.5

设计一个支持以下两种操作的数据结构：

- void addNum(int num) - 从数据流中添加一个整数到数据结构中。
- double findMedian() - 返回目前所有元素的中位数。

示例：

```
addNum(1)
addNum(2)
findMedian() -> 1.5
addNum(3)
findMedian() -> 2
```

题目解析

这道题给我们一个数据流，让我们找出中位数。对于数据流这种动态（流动）的数据，如果使用数组存储，那么每次新进来一个数据都进行排序的话，效率很低。

处理动态数据来说一般使用的数据结构是栈、队列、二叉树、堆。

本题中，我们使用 **堆** 这种数据结构。

首先将数据分为两部分，位于 **上边最大堆** 的数据要比 **下边最小堆** 的数据都要小。

为了保证将数据平均分配到两个堆中，在动态的操作的过程中两个堆中数据的数目之差不能超过 1。

为了保证 **最大堆中的所有数据都小于最小堆中的数据**，在操作过程中，新添加进去的数据需要先和最大堆的最大值或者最小堆中的最小值进行比较。

动画描述

代码实现

```
class MedianFinder {
    public PriorityQueue<Integer> minheap, maxheap;
```

```
    public MedianFinder() {
        //维护较大的元素的最小堆
        maxheap = new PriorityQueue<Integer>(Collections.reverseOrder());
        //维护较小元素的最大堆
        minheap = new PriorityQueue<Integer>();
    }

    // Adds a number into the data structure.
    public void addNum(int num) {
        maxheap.add(num);
        minheap.add(maxheap.poll());
        if (maxheap.size() < minheap.size()) {
            maxheap.add(minheap.poll());
        }
    }

    // Returns the median of current data stream
    public double findMedian() {
        if (maxheap.size() == minheap.size()) {
            return (maxheap.peek() + minheap.peek()) * 0.5;
        } else {
            return maxheap.peek();
        }
    }
};
```