0295. 数据流的中位数

ITCharge 大约 1 分钟

- 标签：设计、双指针、数据流、排序、堆（优先队列）
- 难度：困难

题目链接

- 0295. 数据流的中位数 - 力扣

题目大意

要求：设计一个支持一下两种操作的数组结构：

- `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。

解题思路

使用一个大顶堆 `queMax` 记录大于中位数的数，使用一个小顶堆 `queMin` 小于中位数的数。

- 当添加元素数量为偶数：`queMin` 和 `queMax` 中元素数量相同，则中位数为它们队头的平均值。
- 当添加元素数量为奇数：`queMin` 中的数比 `queMax` 多一个，此时中位数为 `queMin` 的队头。

为了满足上述条件，在进行 `addNum` 操作时，我们应当分情况处理：

- `num > max{queMin}`：此时 `num` 大于中位数，将该数添加到大顶堆 `queMax` 中。新的中位数将大于原来的中位数，所以可能需要将 `queMax` 中的最小数移动到 `queMin` 中。
- `num ≤ max{queMin}`：此时 `num` 小于中位数，将该数添加到小顶堆 `queMin` 中。新的中位数将小于等于原来的中位数，所以可能需要将 `queMin` 中最大数移动到 `queMax` 中。

代码

```py
import heapq

class MedianFinder:

    def __init__(self):
        """
        initialize your data structure here.
        """
        self.queMin = list()
        self.queMax = list()


    def addNum(self, num: int) -> None:
        if not self.queMin or num < -self.queMin[0]:
            heapq.heappush(self.queMin, -num)
            if len(self.queMax) + 1 < len(self.queMin):
                heapq.heappush(s    queMax, -heapq.heappop(self.queMin))
        else:
            heapq.heappush(self.queMax, num)
            if len(self.queMax) > len(self.queMin):
                heapq.heappush(self.queMin, -heapq.heappop(self.queMax))


    def findMedian(self) -> float:
        if len(self.queMin) > len(self.queMax):
            return -self.queMin[0]
        return (-self.queMin[0] + self.queMax[0]) / 2
```