# 从 5 亿个数中找出中位数

## 题目标签

学习时长：20分钟

题目难度：中等

知识点标签：双堆法、分治法、大数据量处理

## 题目描述

从 5 亿个数中找出中位数。

数据排序后，位置在最中间的数就是中位数。当样本数为奇数时，中位数为 第 `(N+1)/2` 个数；当样本数为偶数时，中位数为 第 `N/2` 个数与第 `1+N/2` 个数的均值。

## 1. 解答思路

如果这道题没有内存大小限制，则可以把所有数读到内存中排序后找出中位数。但是最好的排序算法的时间复杂度都为 O(NlogN)。这里使用其他方法。

### 方法一：双堆法

维护两个堆，一个大顶堆，一个小顶堆。大顶堆中最大的数**小于等于**小顶堆中最小的数；保证这两个堆中的元素个数的差不超过 1。

若数据总数为**偶数**，当这两个堆建好之后，**中位数就是这两个堆顶元素的平均值。**当数据总数为**奇数**时，根据两个堆的大小，**中位数一定在数据多的堆的堆顶。**

``

```
class MedianFinder {

    private PriorityQueue<Integer> maxHeap;
    private PriorityQueue<Integer> minHeap;

    /** initialize your data structure here. */
    public MedianFinder() {
        maxHeap = new PriorityQueue<>(Comparator.reverseOrder());
        minHeap = new PriorityQueue<>(Integer::compareTo);
    }

    public void addNum(int num) {
        if (maxHeap.isEmpty() || maxHeap.peek() > num) {
            maxHeap.offer(num);
        } else {
            minHeap.offer(num);
```

```
        }

        int size1 = maxHeap.size();
        int size2 = minHeap.size();
        if (size1 - size2 > 1) {
            minHeap.offer(maxHeap.poll());
        } else if (size2 - size1 > 1) {
            maxHeap.offer(minHeap.poll());
        }
    }

    public double findMedian() {
        int size1 = maxHeap.size();
        int size2 = minHeap.size();

        return size1 == size2
            ? (maxHeap.peek() + minHeap.peek()) * 1.0 / 2
            : (size1 > size2 ? maxHeap.peek() : minHeap.peek());
    }
}
```

以上这种方法，需要把所有数据都加载到内存中。当数据量很大时，就不能这样了，因此，这种方法**适用于数据量较小的情况。**5 亿个数，每个数字占用 4B，总共需要 2G 内存。如果可用内存不足 2G，就不能使用这种方法了，下面介绍另一种方法。

## 方法二：分治法

分治法的思想是把一个大的问题逐渐转换为规模较小的问题来求解。

对于这道题，顺序读取这 5 亿个数字，对于读取到的数字 num，如果它对应的二进制中最高位为，则把这个数字写到 f1 中，否则写入 f0 中。通过这一步，可以把这 5 亿个数划分为两部分，而且 f0 中的数都大于 f1 中的数（最高位是符号位）。

划分之后，可以非常容易地知道中位数是在 f0 还是 f1 中。假设 f1 中有 1 亿个数，那么中位数一定在 f0 中，且是在 f0 中，从小到大排列的第 1.5 亿个数与它后面的一个数的平均值。

> **提示**，5 亿数的中位数是第 2.5 亿与右边相邻一个数求平均值。若 f1 有一亿个数，那么中位数就是 f0 中从第 1.5 亿个数开始的两个数求得的平均值。

对于 f0 可以用次高位的二进制继续将文件一分为二，如此划分下去，直到划分后的文件可以被加载到内存中，把数据加载到内存中以后直接排序，找出中位数。

> **注意**，当数据总数为偶数，如果划分后两个文件中的数据有相同个数，那么中位数就是数据较小的文件中的最大值与数据较大的文件中的最小值的平均值。