1. [从4300000000个32位整数的顺序文件中找出重复的两个数](http://blog.csdn.net/f81892461/article/details/9121639)。顺序文件，不允许随机访问。

方法一：

    解答：Binary Search，但不是对文件内容折半，而是对搜索范围折半。由于4.3G>32位的整数空间，根据鸽笼原理，肯定会有重复的整数。搜索范围从所有的32位正整数开始（全部当成unsigned int，简化问题），即[0, 2^32)，中间值即为2^31。然后遍历文件，如果小于2^31的整数个数大于2^31，则调整搜索范围为[0, 2^31]，反之亦然；然后再对整个文件再遍历一遍，直到得到最后的结果。这样一共会有logn次的搜索，每次过n个整数（每次都是完全遍历），总体的复杂度为o(nlogn)。

方法二:

可以采用Bit-Map，可以申请537500000个char型数组，数组中每个位对应4300000000个整数中的一个数，刚开始时，都所有的位都置0，如果有存在相对应的数，那么对应的位就置一。

问题又出来了，如何才能表示至少包含两次的整数呢？

这时要表示至少包含两次的整数，仅用一位来表示是不够的。那么用两位呢？00表示没有数据，01表示存在一个，10表示存在两个，11表示存在两个以上，over啦。

1. 给定一亿个数字，怎样找到最大的1000个数？要求效率尽可能高，空间使用尽可能少，给出思路即可。

小顶堆。

通过对海量数据的挖掘能有效地揭示用户的行为模式，加深对用户需求的理解，提取用户的集体智慧，从而为研发人员决策提供依据，提升产品用户体验，进而占领市场。

本文内容总结自《Java程序员面试笔试宝典》。

海量处理的基本方法：

1.     Hash法

构造hash表和处理冲突的方法和上面介绍的一样。

在处理海量数据的过程中，使用Hash方法一般可以快速存取、统计某些数据，将大量数据进行分类，例如提取某日访问网站次数最多的ip。

2.     BitMap法

BitMapde(位图)法的基本原理是使用数组来表示某些元素是否存在。

BitMapde(位图)法的时间复杂度为O(n)，比一般的排序都快，但它是以空间换时间，而且又一些限制，即数据状态不是很多，例如排序前集合大小最好已知，而且集合中元素的最大重复次数必须已知。

3.     Bloom Filter 法（适用于对低错误率可以容忍的场合）

Bloom Filter是一种空间效率和时间效率都很高的随机数据结构，可用来检测一个元素是否属于一个集合。其基本原理是位数组与Hash函数的联合使用。首先，Bloom Filter是一个包含m位的位数组，数组的每一位都初始化为0；其次定义k个不同的hash函数，每个函数都可以将集合中的元素映射到位数组中的某一位，这样k个hash函数就将某个元素映射到了位数组中的k个位。查询的时候，根据k个hash函数可以得到数组中的k个位，判断这个k个位是否全为1，若是则说明该元素可不存在。插入的时候，把k个hash函数就将某个元素映射到了位数组中的k个位全部置为1。

Bloom Filter的难点是如何根据输入元素个数n确定m和hash函数。

特点：确定不存在的元素一定不存在，确定存在的数据不一定存在。

优点：具有很好的空间效率和时间效率，它的插入和查询时间都是常数，另外它不保存元素本身，具有良好的安全性。

缺点：牺牲了正确率，当它判断一个元素已存在于集合中时可能不正确。因为可能该元素对应的k个1位可能是被其他多个元素导致的。

4.     数据库优化法

互联网上的数据一般都是被存储到数据库中，如何从数据库中存储的海量信息中提取对自己有用的信息就涉及到了数据的查询技术等相关内容。

常见的数据库优化方法有数据分区、索引、缓存机制、分批处理、优化查询语句、使用采样数据进行数据挖掘等。

5.     倒排索引法

倒排索引是目前搜索引擎公司对搜索引擎最常用的存储方式，也是搜索引擎的核心内容。按照关键字建立索引，关键字指向了包含它的文档。优点是在处理复杂的多关键字时，可在倒排表中完成查询的并、交等逻辑运算，得到结果后再对结果进行存取，这样把记录的查询转换为地址集合的运算，不必对每个记录随机存取，从而提高查找速度。

6.     外排序法

外排序是相对于内部排序而言的，它是大文件的排序，由于内存限制，不能一下子把所有的待排内容都读到内存中进行排序，需要在内存和外部存储器之间进行多次数据交换才能达到对整个文件进行排序的目的。

一般采用归并排序等方式进行外部排序，首先生成若干个子文件，分别对这些子文件进行排序，然后对这些子文件进行多次归并，使得有序的归并段主键扩大，最后在外存上形成整个文件的单一归并段。

外排序适用于大数据的排序以及去重，但外排序的缺陷是回消耗大量的IO，效率不高。

7.     Trie树

Trie树是一种用于快速字符串检索的多叉树结果，原理是利用字符串的公共前缀来减少空间开销。经常被搜索引擎系统用于文件词频统计。优点是：最大限度地减少无谓的字符串比较，查询效率比散列表高。适合用于数据量大，重复多，但是数据种类小可以放入内存的情况。

但 是当系统中存在大量字符串且字符串基本没有公共前缀，则相应的Trie树会非常消耗内存。

8.     堆

|  |  |
| --- | --- |
| 最大堆 | 求前k小 |
| 最小堆 | 求前k大 |
| 双堆 | 中位数 |

  比如用最小堆求前k个最大的元素：

先从待找的n个数字中找出前k个建成小顶堆，然后依次读后面的元素并与小顶堆的堆顶进行比较，如果当前元素小或相等（这个相等是自己加的，需要进一步验证，应该是正确的，因为此时的堆顶元素是前k个里面最小的。如果遇到相等的即使该元素在最后前k个里面，也可以不要那个元素因为之前的已经够k个了，不需要再进行替换了），则继续读后面的元素；如果当前的元素大，则用当前元素替换堆顶元素，然后调整小顶堆。

9.     双层桶法

双层桶不是一种数据结构，而是一种算法思想，类似于分治思想。因为元素范围很大，不能利用直接寻址表，所以通过多次划分，逐步确定范围，然后最后在一个可以接受的范围内进行。

桶排序一般适用于寻找第k个大的数，寻找中位数、寻找不重复或重复的数字。

如：1）在一个文件中有10G个整数，乱序排列，要求找出中位数，内存限制是2GB；

2）现在有一个0~30000的随机数生成器。请根据这个随机数生成器，设计一个抽奖范围是0~350000彩票中奖号码列表，其中要包含20000个中奖号码。

10. MapReduce法

MapReduce是云计算的核心技术之一，是一种简化并行计算的分布式编程模型，主要目的是为了大型集群的系统能在大数据集上进行并行工作，并用于大规模数据的并行运算。

海量数据处理的最大难题在于数据规模巨大，使得传统处理方式面临计算能力不和存储能力不足的瓶颈问题，而基于Hadoop可以非常轻松和方便完成处理海量数据的分布式并行程序，并运行与大规模集群上