**1、海量日志数据，提取出某日访问百度次数最多的那个IP。**

　　首先是这一天，并且是访问百度的日志中的IP取出来，逐个写入到一个大文件中。注意到IP是32位的，最多有个2^32个IP。同样可以采用映射的方法，比如模1000，把整个大文件映射为1000个小文件，再找出每个小文中出现频率最大的IP（可以采用hash\_map进行频率统计，然后再找出频率最大的几个）及相应的频率。然后再在这1000个最大的IP中，找出那个频率最大的IP，即为所求。

　　或者如下阐述：

　　算法思想：分而治之+Hash

　　1.IP地址最多有2^32=4G种取值情况，所以不能完全加载到内存中处理；

　　2.可以考虑采用“分而治之”的思想，按照IP地址的Hash(IP)24值，把海量IP日志分别存储到1024个小文件中。这样，每个小文件最多包含4MB个IP地址；

　　3.对于每一个小文件，可以构建一个IP为key，出现次数为value的Hash map，同时记录当前出现次数最多的那个IP地址；

　　4.可以得到1024个小文件中的出现次数最多的IP，再依据常规的排序算法得到总体上出现次数最多的IP；

**2、搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为1-255字节。**

　　假设目前有一千万个记录（这些查询串的重复度比较高，虽然总数是1千万，但如果除去重复后，不超过3百万个。一个查询串的重复度越高，说明查询它的用户越多，也就是越热门。），请你统计最热门的10个查询串，要求使用的内存不能超过1G。

　　典型的Top K算法，还是在这篇文章里头有所阐述，

　　文中，给出的最终算法是：

　　第一步、先对这批海量数据预处理，在O（N）的时间内用Hash表完成统计（之前写成了排序，特此订正。July、2011.04.27）；

　　第二步、借助堆这个数据结构，找出Top K，时间复杂度为N‘logK。

　　即，借助堆结构，我们可以在log量级的时间内查找和调整/移动。因此，维护一个K(该题目中是10)大小的小根堆，然后遍历300万的Query，分别和根元素进行对比所以，我们最终的时间复杂度是：O（N） + N’\*O（logK），（N为1000万，N’为300万）。ok，更多，详情，请参考原文。

　　或者：采用trie树，关键字域存该查询串出现的次数，没有出现为0。最后用10个元素的最小推来对出现频率进行排序。

**3、有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M。返回频数最高的100个词。**

　　方案：顺序读文件中，对于每个词x，取hash(x)P00，然后按照该值存到5000个小文件（记为x0,x1,…x4999）中。这样每个文件大概是200k左右。

　　如果其中的有的文件超过了1M大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过1M。

　　对每个小文件，统计每个文件中出现的词以及相应的频率（可以采用trie树/hash\_map等），并取出出现频率最大的100个词（可以用含100个结点的最小堆），并把100个词及相应的频率存入文件，这样又得到了5000个文件。下一步就是把这5000个文件进行归并（类似与归并排序）的过程了。

**4、有10个文件，每个文件1G，每个文件的每一行存放的都是用户的query，每个文件的query都可能重复。要求你按照query的频度排序。**

　　还是典型的TOP K算法，解决方案如下：

　　方案1：

　　顺序读取10个文件，按照hash(query)的结果将query写入到另外10个文件（记为）中。这样新生成的文件每个的大小大约也1G（假设hash函数是随机的）。

　　找一台内存在2G左右的机器，依次对用hash\_map(query, query\_count)来统计每个query出现的次数。利用快速/堆/归并排序按照出现次数进行排序。将排序好的query和对应的query\_cout输出到文件中。这样得到了10个排好序的文件（记为）。

　　对这10个文件进行归并排序（内排序与外排序相结合）。

　　方案2：

　　一般query的总量是有限的，只是重复的次数比较多而已，可能对于所有的query，一次性就可以加入到内存了。这样，我们就可以采用trie树/hash\_map等直接来统计每个query出现的次数，然后按出现次数做快速/堆/归并排序就可以了。

　　方案3：

　　与方案1类似，但在做完hash，分成多个文件后，可以交给多个文件来处理，采用分布式的架构来处理（比如MapReduce），最后再进行合并。

**5、 给定a、b两个文件，各存放50亿个url，每个url各占64字节，内存限制是4G，让你找出a、b文件共同的url？**

　　方案1：可以估计每个文件安的大小为5G×64=320G，远远大于内存限制的4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。

　　遍历文件a，对每个url求取hash(url)00，然后根据所取得的值将url分别存储到1000个小文件（记为a0,a1,…,a999）中。这样每个小文件的大约为300M。

　　遍历文件b，采取和a相同的方式将url分别存储到1000小文件（记为b0,b1,…,b999）。这样处理后，所有可能相同的url都在对应的小文件（a0vsb0,a1vsb1,…,a999vsb999）中，不对应的小文件不可能有相同的url。然后我们只要求出1000对小文件中相同的url即可。

　　求每对小文件中相同的url时，可以把其中一个小文件的url存储到hash\_set中。然后遍历另一个小文件的每个url，看其是否在刚才构建的hash\_set中，如果是，那么就是共同的url，存到文件里面就可以了。

　　方案2：如果允许有一定的错误率，可以使用Bloom filter，4G内存大概可以表示340亿bit。将其中一个文件中的url使用Bloom filter映射为这340亿bit，然后挨个读取另外一个文件的url，检查是否与Bloom filter，如果是，那么该url应该是共同的url（注意会有一定的错误率）。

　　Bloom filter日后会在本BLOG内详细阐述。

**6、在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数。**

　　方案1：采用2-Bitmap（每个数分配2bit，00表示不存在，01表示出现一次，10表示多次，11无意义）进行，共需内存2^32 \* 2 bit=1 GB内存，还可以接受。然后扫描这2.5亿个整数，查看Bitmap中相对应位，如果是00变01，01变10，10保持不变。所描完事后，查看bitmap，把对应位是01的整数输出即可。

　　方案2：也可采用与第1题类似的方法，进行划分小文件的方法。然后在小文件中找出不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

**7、腾讯面试题：给40亿个不重复的unsigned int的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那40亿个数当中？**

　　与上第6题类似，我的第一反应时快速排序+二分查找。以下是其它更好的方法：

　　方案1：oo，申请512M的内存，一个bit位代表一个unsigned int值。读入40亿个数，设置相应的bit位，读入要查询的数，查看相应bit位是否为1，为1表示存在，为0表示不存在。

　　方案2：这个问题在《编程珠玑》里有很好的描述，大家可以参考下面的思路，探讨一下：

　　又因为2^32为40亿多，所以给定一个数可能在，也可能不在其中；

　　这里我们把40亿个数中的每一个用32位的二进制来表示

　　假设这40亿个数开始放在一个文件中。

　　然后将这40亿个数分成两类:

　　1.最高位为0

　　2.最高位为1

　　并将这两类分别写入到两个文件中，其中一个文件中数的个数<=20亿，而另一个>=20亿（这相当于折半了）；

　　与要查找的数的最高位比较并接着进入相应的文件再查找

　　再然后把这个文件为又分成两类:

　　1.次最高位为0

　　2.次最高位为1

　　并将这两类分别写入到两个文件中，其中一个文件中数的个数<=10亿，而另一个>=10亿（这相当于折半了）；

　　与要查找的数的次最高位比较并接着进入相应的文件再查找。

　　…….

　　以此类推，就可以找到了,而且时间复杂度为O(logn)，方案2完。

　　附：这里，再简单介绍下，位图方法：

　　使用位图法判断整形数组是否存在重复

　　判断集合中存在重复是常见编程任务之一，当集合中数据量比较大时我们通常希望少进行几次扫描，这时双重循环法就不可取了。

　　位图法比较适合于这种情况，它的做法是按照集合中最大元素max创建一个长度为max+1的新数组，然后再次扫描原数组，遇到几就给新数组的第几位置上1，如遇到5就给新数组的第六个元素置1，这样下次再遇到5想置位时发现新数组的第六个元素已经是1了，这说明这次的数据肯定和以前的数据存在着重复。这种给新数组初始化时置零其后置一的做法类似于位图的处理方法故称位图法。它的运算次数最坏的情况为2N。如果已知数组的最大值即能事先给新数组定长的话效率还能提高一倍。

　　欢迎，有更好的思路，或方法，共同交流。

**8、怎么在海量数据中找出重复次数最多的一个？**

　　方案1：先做hash，然后求模映射为小文件，求出每个小文件中重复次数最多的一个，并记录重复次数。然后找出上一步求出的数据中重复次数最多的一个就是所求（具体参考前面的题）。

**9、上千万或上亿数据（有重复），统计其中出现次数最多的钱N个数据。**

　　方案1：上千万或上亿的数据，现在的机器的内存应该能存下。所以考虑采用hash\_map/搜索二叉树/红黑树等来进行统计次数。然后就是取出前N个出现次数最多的数据了，可以用第2题提到的堆机制完成。

**10、一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前10个词，请给出思想，给出时间复杂度分析。**

　　方案1：这题是考虑时间效率。用trie树统计每个词出现的次数，时间复杂度是O(n\*le)（le表示单词的平准长度）。然后是找出出现最频繁的前10个词，可以用堆来实现，前面的题中已经讲到了，时间复杂度是O(n\*lg10)。所以总的时间复杂度，是O(n\*le)与O(n\*lg10)中较大的哪一个。

　　附、100w个数中找出最大的100个数。

　　方案1：在前面的题中，我们已经提到了，用一个含100个元素的最小堆完成。复杂度为O(100w\*lg100)。

　　方案2：采用快速排序的思想，每次分割之后只考虑比轴大的一部分，知道比轴大的一部分在比100多的时候，采用传统排序算法排序，取前100个。复杂度为O(100w\*100)。

　　方案3：采用局部淘汰法。选取前100个元素，并排序，记为序列L。然后一次扫描剩余的元素x，与排好序的100个元素中最小的元素比，如果比这个最小的要大，那么把这个最小的元素删除，并把x利用插入排序的思想，插入到序列L中。依次循环，知道扫描了所有的元素。复杂度为O(100w\*100)。