海量数据处理面试

海量日志数据，提取出某日访问百度次数最多的那个IP

- 分而治之：数据量太大，内存受限。把大文件划分为小文件（取模映射）
- 哈希统计：HashMap<IP,count>
- 取最大值

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

由上面第1题，我们知道，数据大则划为小的，但如果数据规模比较小，能一次性装入内存呢?比如这第2题，虽然有一千万个Query，但是由于重复度比较高，因此事实上只有300万的Query，每个Query255Byte，因此我们可以考虑把他们都放进内存中去，而现在只是需要一个合适的数据结构，在这里，Hash Table绝对是我们优先的选择。所以我们摒弃分而治之/hash映射的方法，直接上hash统计，然后排序。So，

- hash统计：先对这批海量数据预处理(维护一个Key为Query字串，Value为该Query出现次数的HashTable，即hash_map(Query，Value)，每次读取一个Query，如果该字串不在Table中，那么加入该字串，并且将Value值设为1；如果该字串在Table中，那么将该字串的计数加一即可。最终我们在O(N)的时间复杂度内用Hash表完成了统计；
- 堆排序：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）。

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

- 分而治之/hash映射：顺序读文件中，对于每个词x，取hash(x)%5000，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。
- hash统计：对每个小文件，采用trie树/hash_map等统计每个文件中出现的词以及相应的频率。
- 堆/归并排序：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。最后就是把这5000个文件进行归并（类似于归并排序）的过程了。

海量数据分布在100台电脑中，想个办法高效统计出这批数据的TOP10

- 堆排序：在每台电脑上求出TOP10，可以采用包含10个元素的堆完成（TOP10小，用最大堆，TOP10大，用最小堆）。比如求TOP10大，我们首先取前10个元素调整成最小堆，如果发现，然后扫描后面的数据，并与堆顶元素比较，如果比堆顶元素大，那么用该元素替换堆顶，然后再调整为最小堆。最后堆中的元素就是TOP10大。
- 求出每台电脑上的TOP10后，然后把这100台电脑上的TOP10组合起来，共1000个数据，再利用上面类似的方法求出TOP10就可以了。

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

直接上：

- hash映射：顺序读取10个文件，按照hash(query)%10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）。
- hash统计：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。注：hash_map(query,query_count)是用来统计每个query的出现次数，不是存储他们的值，出现一次，则count+1。
- 堆/快速/归并排序：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）。

除此之外，此题还有以下两个方法：
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。

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

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。

- 分而治之/hash映射：遍历文件a，对每个url求取$\text{hash}(url)\%1000$，然后根据所取得的值将url分别存储到1000个小文件（记为$a_0, a_2, \cdots, a_{999}$）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小文件中（记为$b_0, b_1, \cdots, b_{999}$）。这样处理后，所有可能相同的url都在对应的小文件（a_0 vs b_0, a_1 vs $b_1, \ldots, a_0$ vs b_{999}）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。
- hash统计：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。

怎么在海量数据中找出重复次数最多的一个?

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）。

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。

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，可以用堆机制完成。

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。

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。

2.5亿个整数中找出不重复的整数的个数，内存空间不足以容纳这2.5亿个整数

方案一：整数个数为2^32，内存中每一位表示一个整数，需要约500M空间。

方案二（判断某数是否存在于2.5亿数中）：先外部排序，2.5亿个整数可能有连续，构造数据结构[x,y]表示从x到y之间的数均存在，再二分查找

5亿个int找它们的中位数

这个例子比上面那个更明显。首先我们将int划分为2^16个区域，然后读取数据统计落到各个区域里的数的个数，之后我们根据统计结果就可以判断中位数落到那个区域，同时知道这个区域中的第几大数刚好是中位数。然后第二次扫描我们只统计落在这个区域中的那些数就可以了。
实际上，如果不是int是int64，我们可以经过3次这样的划分即可降低到可以接受的程度。即可以先将int64分成2^24个区域，然后确定区域的第几大数，在将该区域分成2^20个子区域，然后确定是子区域的第几大数，然后子区域里的数的个数只有2^20，就可以直接利用direct addr table进行统计了。