而不会产生溢出，所以哈希表的 key 需要占用 4B，value 也是 4B。那么哈希表的一条记录(key,value)需要占用 8B，当哈希表记录数为 2 亿个时，需要至少 1.6GB 的内存。

但如果 20 亿个数中不同的数超过 2 亿种，最极端的情况是 20 亿个数都不同，那么在哈希表中可能需要产生 20 亿条记录，这样内存会不够用，所以一次性用哈希表统计 20 亿个数的办法是有很大风险的。

解决办法是把包含 20 亿个数的大文件用哈希函数分成 16 个小文件，根据哈希函数的性质，同一种数不可能被哈希到不同的小文件上，同时每个小文件中不同的数一定不会大于 2 亿种，假设哈希函数足够好。然后对每一个小文件用哈希表来统计其中每种数出现的次数，这样我们就得到了 16 个小文件中各自出现次数最多的数，还有各自的次数统计。接下来只要选出这 16 个小文件各自的第一名中谁出现的次数最多即可。

把一个大的集合通过哈希函数分配到多台机器中，或者分配到多个文件里，这种技巧是处理大数据面试题时最常用的技巧之一。但是到底分配到多少台机器、分配到多少文件，在解题时一定要确定下来。可能是在与面试官沟通的过程中由面试官指定，也可能是根据具体的限制来确定，比如本题确定分成 16 个文件，就是根据内存限制 2GB 的条件来确定的。

40 亿个非负整数中找到没出现的数

【题目】

32 位无符号整数的范围是 0～4294967295，现在有一个正好包含 40 亿个无符号整数的文件，所以在整个范围中必然有没出现过的数。可以使用最多 1GB 的内存，怎么找到所有没出现过的数？

进阶：内存限制为 10MB，但是只用找到一个没出现过的数即可。

【难度】

尉 ★★☆☆

【解答】

原问题。如果用哈希表来保存出现过的数，那么如果 40 亿个数都不同，则哈希表的记录数为 40 亿条，存一个 32 位整数需要 4B，所以最差情况下需要 40 亿×4B=160 亿字节，大约需要 16GB 的空间，这是不符合要求的。

哈希表需要占用很多空间，我们可以使用 bit map 的方式来表示数出现的情况。具体地说，是申请一个长度为 4294967295 的 bit 类型的数组 bitArr，bitArr 上的每个位置只可以表示 0 或 1 状态。8 个 bit 为 1B，所以长度为 4294967295 的 bit 类型的数组占用 500MB 空间。

怎么使用这个 bitArr 数组呢？就是遍历这 40 亿个无符号数，例如，遇到 7000，就把 bitArr[7000]设置为 1。遇到所有的数时，就把 bitArr 相应位置的值设置为 1。

遍历完成后，再依次遍历 bitArr，哪个位置上的值没被设置为 1，哪个数就不在 40 亿个数中。例如，发现 bitArr[8001]==0，那么 8001 就是没出现过的数，遍历完 bitArr 之后，所有没出现的数就都找出来了。

进阶问题。现在只有 10MB 的内存，但也只要求找到其中一个没出现过的数即可。首先，0~4294967295 这个范围是可以平均分成 64 个区间的，每个区间是 67108864 个数，例如：第 0 区间（0~67108863）、第 1 区间（67108864~134217728）、第 i 区间（67108864×i ~ 67108864×(i+1)-1），……，第 63 区间（4227858432~4294967295）。因为一共只有 40 亿个数，所以，如果统计落在每一个区间上的数有多少，肯定有至少一个区间上的计数少于 67108864。利用这一点可以找出其中一个没出现过的数。具体过程为：

第一次遍历时，先申请长度为 64 的整型数组 countArr[0..63]，countArr[i]用来统计区间 i 上的数有多少。遍历 40 亿个数，根据当前数是多少来决定哪一个区间上的计数增加。例如，如果当前数是 3422552090，3422552090/67108864=51，所以第 51 区间上的计数增加 countArr[51]++。遍历完 40 亿个数之后，遍历 countArr，必然会有某一个位置上的值(countArr[i])小于 67108864，表示第 i 区间上至少有一个数没出现过。我们肯定会至少找到一个这样的区间。此时使用的内存就是 countArr 的大小（64×4B），是非常小的。

假设我们找到第 37 区间上的计数小于 67108864，以下为第二次遍历的过程：

1. 申请长度为 67108864 的 bit map，这占用大约 8MB 的空间，记为 bitArr[0..67108863]；

2. 再遍历一次 40 亿个数，此时的遍历只关注落在第 37 区间上的数，记为 num（num/67108864==37），其他区间的数全部忽略。

3. 如果步骤 2 的 num 在第 37 区间上，将 bitArr[num - 67108864*37]的值设置为 1，也就是只做第 37 区间上的数的 bitArr 映射。

4. 遍历完 40 亿个数之后，在 bitArr 上必然存在没被设置成 1 的位置，假设第 i 个位置上的值没设置成 1，那么 67108864×37+i 这个数就是一个没出现过的数。

总结一下进阶的解法：

1. 根据 10MB 的内存限制，确定统计区间的大小，就是第二次遍历时的 bitArr 大小。

2. 利用区间计数的方式，找到那个计数不足的区间，这个区间上肯定有没出现的数。

3．对这个区间上的数做 bit map 映射，再遍历 bit map，找到一个没出现的数即可。

找到 100 亿个 URL 中重复的 URL 以及搜索词汇的 top *K* 问题

【题目】

有一个包含 100 亿个 URL 的大文件，假设每个 URL 占用 64B，请找出其中所有重复的 URL。

【补充题目】

某搜索公司一天的用户搜索词汇是海量的（百亿数据量），请设计一种求出每天最热 top 100 词汇的可行办法。

【难度】

士 ★☆☆☆

【解答】

原问题的解法使用解决大数据问题的一种常规方法：把大文件通过哈希函数分配到机器，或者通过哈希函数把大文件拆成小文件。一直进行这种划分，直到划分的结果满足资源限制的要求。首先，你要向面试官询问在资源上的限制有哪些，包括内存、计算时间等要求。在明确了限制要求之后，可以将每条 URL 通过哈希函数分配到若干机器或者拆分成若干小文件，这里的“若干”由具体的资源限制来计算出精确的数量。

例如，将 100 亿字节的大文件通过哈希函数分配到 100 台机器上，然后每一台机器分别统计分给自己的 URL 中是否有重复的 URL，同时哈希函数的性质决定了同一条 URL 不可能分给不同的机器；或者在单机上将大文件通过哈希函数拆成 1000 个小文件，对每一个小文件再利用哈希表遍历，找出重复的 URL；或者在分给机器或拆完文件之后，进行排序，排序过后再看是否有重复的 URL 出现。总之，牢记一点，很多大数据问题都离不开分流，要么是哈希函数把大文件的内容分配给不同的机器，要么是哈希函数把大文件拆成小文件，然后处理每一个小数量的集合。

补充问题最开始还是用哈希分流的思路来处理，把包含百亿数据量的词汇文件分流到