一、大数据量去重搜索

1、Bitmap

思想：空间压缩思想

**快速排序：**

假设我们要对0-7内的5个元素(4,7,2,5,3)排序

开辟一个Byte空间



对应位设置为1:



遍历一遍Bit区域，将该位是一的位的编号输出（2，3，4，5，7），这样就达到了排序的目的，时间复杂度O(n)。

运行效率高，不需要进行比较和位移，但不可对重复的数据进行排序和查找。

**快速去重**：

一个数字的状态只有三种，分别为不存在，只有一个，有重复。我们只需要2bits就可以对一个数字的状态进行存储了，假设我们设定一个数字不存在为00，存在一次01，存在两次及其以上为11。遍历一次数据，如果对应的状态位为00，则将其变为01；如果对应的状态位为01，则将其变为11；如果为11，,对应的转态位保持不变。最后，我们将状态位为01的进行统计，就得到了不重复的数字个数，时间复杂度为O(n)。

**快速查找：**

1、首先我们先对所有的数字进行一次遍历，然后将相应的状态位改为1.

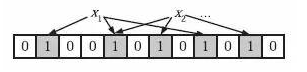
2、查询：由于我们的bitmap采取的是连续存储（整形数组形式，一个数组元素对应32bits），我们采用分桶的思想，一个数组元素可以存储32个状态位，那将待查询的数字除以32，定位到对应的数组元素（桶），然后再求余（%32），可以定位到相应的状态位。

2、Bloom filter（布隆过滤器）

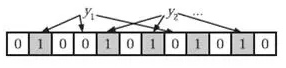
是Bitmap的扩展，使用Bitmap的思想，我们可以将存储空间进行压缩，Bloom Fliter允许在低错误率的场景下，大大地进行空间压缩，是一种拿错误率换取空间的数据结构。

基本思想：

Bloom Filter使用k个相互独立的哈希函数（Hash Function），它们分别将集合中的每个元素映射到{1,…,m}的范围中。对任意一个元素x，第i个哈希函数映射的位置hi(x)就会被置为1（1≤i≤k）。注：如果一个位置多次被置为1，那么只有第一次会起作用，后面几次将没有任何效果。



在判断y是否属于这个集合时，对y应用k次哈希函数，若所有hi(y)的位置都是1（1≤i≤k），就认为y是集合中的元素，否则就认为y不是集合中的元素。



3、堆排序

实现百万级数据去若干数量的最大数字

思路：我们指定获取前M个最大的数字，所以我们构建一个有M个结点的小根堆，遍历百万级数组，用数组去和小根堆的根结点进行比较，如果比根结点大，则替换根结点，然后重建小根堆，维护小根堆的性质，保证根结点在堆中最小，遍历完成后，堆中数字则为数组中最大的数字。

二、题目分析

1、海量日志数据，提取出访问次数最多的IP

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)%1024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。统计最热门的10个查询串

思想：检索的记录很多，但重复率高

1、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

2、然后借助堆这个数据结构，找出TopK,时间复杂度为N`logK,维护一个K大小的小根堆，然后遍历300万的Query,分别和根元素进行对比，我们最终的时间复杂度为O（N）+N`O(logK) N为1000万，N`为300万

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，对于每个词x,取hash(x)%5000,然后按照该值存到5000个小文件。这样每个文件大概是200K左右。

对于每个小文件，