分而治之（高频问题）

1.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/hash映射

首先把这一天访问百度日志的所有IP提取出来，然后逐个写入到一个大文件中，接着采用映射的方法，比如%1000，把整个大文件映射为1000个小文件

hash_map统计

当大文件转化成了小文件，那么我们便可以采用hash_map(ip,value)来分别对1000个小文件中的IP进行频率统计，再找出每个小文件中出现频率最大的IP

堆/快速排序

统计出1000个频率最大的IP后，依据各自频率的大小进行排序(可采取堆排序)，找出那个频率最大的IP，即为所求

小根堆寻找前K个最大数

初始化一个大小为K的小根堆（堆顶是第K大，第1大~第K-1大也在堆里）

遍历输入的数据，对于每个数据元素

1.如果堆未满（即元素数量小于K），则直接将元素加入堆中

2.如果堆已满，则比较当前元素与堆顶元素（即当前堆中的最小值，也就是第K大的数）

2.1如果当前元素大于堆顶元素，则弹出堆顶元素，将当前元素加入堆中。

2.2如果当前元素不大于堆顶元素，则不做任何操作。

3.遍历结束后，堆中剩余的K个元素就是前K个最大的数

2.寻找热门查询，300万个查询字符串中统计最热门的10个查询

原题

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，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

假设去重后只有300万的Query，而每个Query最多255B，则最多占用内存为3M*0.25K=0.75G<1G，可以全部放入内存

hash_map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hash_map，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，并将Value值设为1

如果该字串在Table中，那么将该字串的计数加1即可

最终在O(N)的时间复杂度内用hashmap完成了统计

堆排序

借助堆这个数据结构，找出TopK，时间复杂度为N'·O（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），其中，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

分而治之/hash映射

顺序读取文件，对于每个词x，取hash(x)%5000**（相同的单词会进入到同一个文件中）**，然后把该值存到5000个小文件中

这样每个文件大概是200k左右

如果其中有的小文件超过了1M大小，还可以按照类似的方法继续往下分**（比如a_0,a_1...）**，直到分解得到的小文件的大小都不超过1M

hash map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)后

这100个词所在的文件中的所有词，再重新放到5000个小文件中（一次就去除了4900个小文件的数据）

- **7在a文件（567） 8在b文件（128），前2大其实都在a文件中；只能保证：不在这100个词所在的文件 一定不是高频的前100个；所以不能只看这100个词**

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

4.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

如果同一个数据元素只出现在某一台机器中

堆排序

在每台电脑上求出TOP 10，可以采用包含10个元素的堆完成

组合归并

求出每台电脑上的TOP 10后，然后把这100台电脑上的TOP 10组合起来，共1000个数据，再利用上面类似的方法求出TOP 10就可以了

如果同一个元素重复出现在不同的电脑中

第一台的数据分布及各自出现频率为：a(50)，b(50)，c(49)，d(49)，e(0)，f(0)

- 其中，括号里的数字代表某个数据出现的频率，如a(50)表示a出现了50次。

第二台的数据分布及各自出现频率为：a(0)，b(0)，c(49)，d(49)，e(50)，f(50)

遍历一遍所有数据，重新hash取摸

使得同一个元素只出现在单独的一台电脑中

然后采用上面所说的方法，统计每台电脑中各个元素的出现次数找出TOP 10，继而组合100台电脑上的TOP 10，找出最终的TOP 10

5.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

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a0,a1..a9)中，可以保证相同query进入同一文件，这时候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 map统计

找一台内存在2G左右的机器（为了保证同一个query只会在一个文件中出现）

依次用hash_map(query, query_count)来统计每个query出现的次数

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q0,q1,...,q9)。最后，对这10个文件进行归并排序(内排序与外排序相结合)

6.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?

估计每个文件大小为5Gx64=320G，远远大于内存限制的4G

分而治之/hash映射

遍历文件a，对每个url求取 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小文件中(记为b1-b999)。这样处理后，所有可能相同的url都在对应的小文件(即a1 对应 b1，a2 对应 b2..，a999 对应 b999)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 set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(hash_set自带去重，去重后，整体字符串可能是装的下内存的)。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

7.100万个数中找出最大的100个数

用一个含100个元素的最小堆完成

举一反三

1.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

2.上千万或上亿数据(有重复)，统计其中出现次数最多的前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

用一个含N个元素的最小堆完成

3.一个文本文件，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0(n·le)(le表示单词的平均长度)

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0(n·lg10)

总的时间复杂度，是0(n·le)与0(n·lg10)中较大的哪一个

- **trie树适合前缀相关的扩展问题，哈希表统计频率+堆更优**

4.1000万字符串，其中有些是重复的，需要把重复的全部去掉，保留没有重复的字符串。请怎么设计和实现?

用trie树比较合适，hash_map也行

也可以先hash成小文件分开处理再综合

5.一个文本文件，找出前10个经常出现的词，但这次文件比较长，说是上亿行或十亿行，总之无法一次读入内存，问最优解

首先根据用hash并求模，将文件分解为多个小文件，对于单个文件利用上题的方法求出每个文件件中10个最常出现的词

然后再进行归并处理，找出最终的10个最常出现的词

外排序

https://zhuanlan.zhihu.com/p/343986766外部排序归并算法

在内存外面的排序，因为当要处理的数据量很大，而不能一次装入内存时，此时只能放在读写较慢的外存储器(通常是硬盘)上

在排序阶段，先读入能放在内存中的数据量，将其排序输出到一个临时文件，依此进行，将待排序数据组织为多个有序的临时文件

而后在归并阶段将这些临时文件组合为一个大的有序文件，也即排序结果

1.给10^7个数据量的磁盘文件排序

原题

输入：给定一个文件，里面最多含有n个不重复的正整数(也就是说可能含有少于n个不重复正整数)，且其中每个数都小于等于n，n=10^7

输出：得到按从小到大升序排列的包含所有输入的整数的列表。 条件:最多有大约1MB的内存空间可用，但磁盘空间足够。且要求运行时间在5分钟以下，10秒为最佳结果

位图+多路归并

如果用位图方案的话，需要约10^7/8/1024≈1.22MB

可分为2块(k=2，1趟反正占用的内存只有1.22/2M)，1~4999999和5000000~9999999

位图：先遍历一趟，首先排序处理1~4999999之间的整数(用5000000/8=625000个字的存储空间来排序0~4999999之间的整数)

多路归并：然后再第二趟，对5000001~1000000之间的整数进行排序处理

举一反三

1.已知某个文件内包含一些电话号码，每个号码为8位数字，统计不同号码的个数。8位最多99 999 999，大概需要99m个bit，大概10几m字节的内存即可

初始化1 亿bit 大小的的bitmap，循环读文件的号码，将号码对应的bitmap的位置标记为 1，比如bitmap[66666666]=1，如果后面遇到相同的号码，还是标记为 1，这样重复的号码就会被去重了

最后统计 bitmap 中 bit 为1的个数，就代表不同号码的个数了

多层划分

本质上还是分而治之的思想，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

1.　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

整数个数为2^32，也就是我们可以将这2^32个数，划分为2^8个区域(比如用单个文件代表一个区域)，将数据分离到不同的区域

然后不同的区域在利用bitmap就可以直接解决了。也就是说只要有足够的磁盘空间，就可以很方便的解决。

- **抽屉原理/鸽巢原理：如有n+1个元素放到n个集合中去，其中必定有一个集合里至少有两个元素**

2.　5亿个int找它们的中位数

首先将int划分为2^16个区域，每个区域包含一部分整数，对个每个区域的个数进行统计

然后读取数据统计落到各个区域里的数的个数

之后根据统计结果就可以判断中位数落到哪个区域，同时知道这个区域中的第几大数刚好是中位数

然后第二次扫描只统计落在这个区域中的那些数就可以了，最后就能找到中位数了

位图（bitmap）

要求

输入数据限制在相对较小的范围内

数据没有重复

其中的每条记录都是单一的整数，没有任何其它与之关联的数据

方案

将所有的位都置为0，从而将集合初始化为空

通过读入文件中的每个整数来建立集合，将每个对应的位都置为1

检验每一位，如果该位为1，就输出对应的整数（产生有序的输出文件）

- **布隆过滤器可以看做是对位图的扩展**

1.　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

同多层划分的题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2.5 亿*2bit ≈60 MB内存

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

扫描完事后，查看bitmap，把对应位是01的整数输出即可

2.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?

可以用位图/Bitmap的方法，申请512M的内存，一个bit位代表一个unsigned int值

512M=512·1024·1024=536870912字节=536870912字节·8=4294967296bit≈42.9亿个值

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

字典树（Trie树）

介绍

Trie树，即字典树，又称单词查找树或键树，是一种树形结构

典型应用是用于统计和排序大量的字符串(但不仅限于字符串)，所以经常被搜索引擎系统用于文本词频统计

优点是最大限度地减少无谓的字符串比较，查询效率比较高

核心思想是空间换时间，利用字符串的公共前缀来降低查询时间的开销以达到提高效率的目的

基本性质

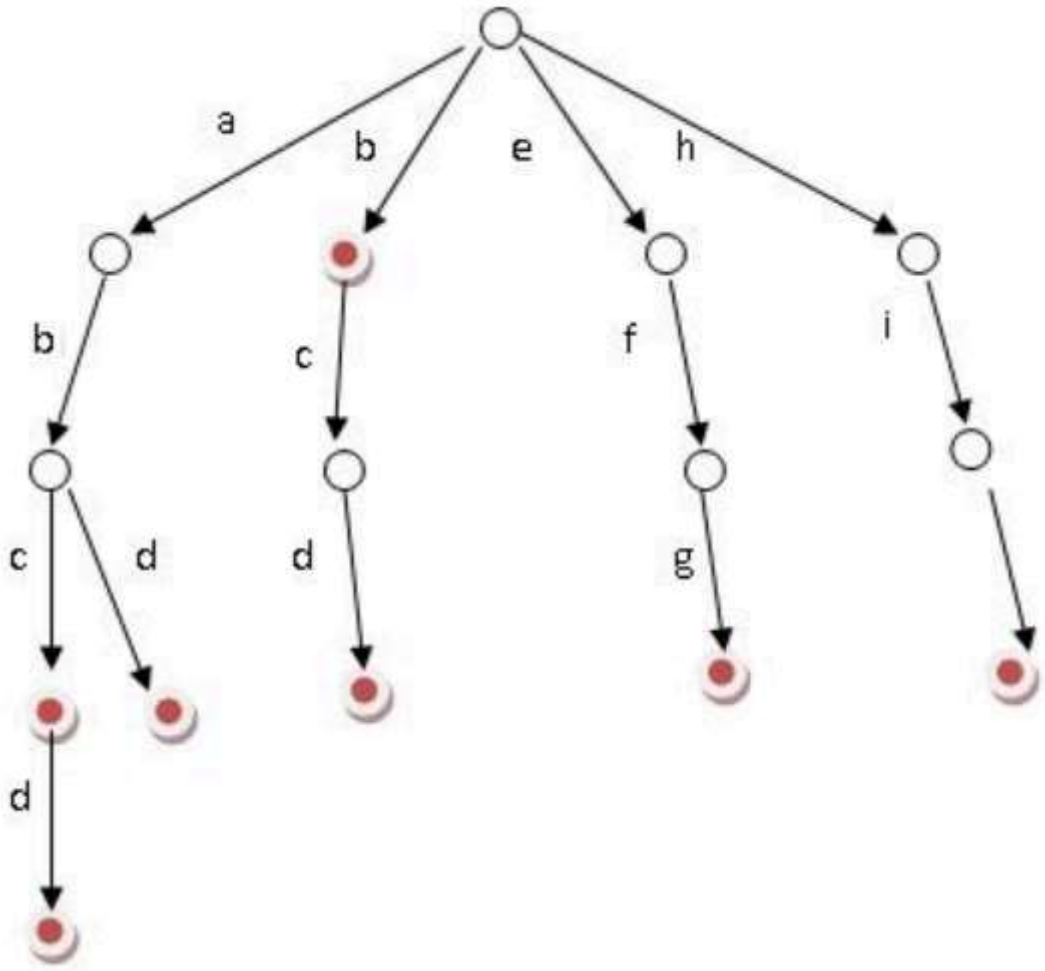
根节点不包含字符，除根节点外每一个节点都只包含一个字符

从根节点到某一节点，路径上经过的字符连接起来，为该节点对应的字符串

每个节点的所有子节点包含的字符都不相同

树的构建

现在有b，abc，abd，bcd，abcd，efg，hii这6个单词，可以构建一棵如下图所示的树



对于每一个节点，从根遍历到他的过程就是一个单词，如果这个节点被标记为红色，就表示这个单词存在，否则不存在

对于一个单词，只要顺着他从根走到对应的节点，再看这个节点是否被标记为红色就可以知道它是否出现过了，把这个节点标记为红色，就相当于插入了这个单词。

这样一来，查询和插入可以一起完成，所用时间仅仅为单词长度之和

trie树每一层的节点数是26^i级别的

为了节省空间，还可以用动态链表或者用数组来模拟动态，而空间的花费，不会超过单词数与单词长度的乘积

查询

比如要查找bcd，顺着路径b->bc->bcd就找到了

1.一个文本文件，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

同分而治之的举一反三的题3

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0(n·le)(le表示单词的平均长度)

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0(n·lg10)

总的时间复杂度，是0(n·le)与0(n·lg10)中较大的哪一个

2.寻找热门查询

原题

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，这些查询串的重复读比较高，虽然总数是1千万，但是如果去除重复和，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

同分而治之的的题2

利用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

倒排索引

一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射，常被应用于搜索引擎和关键字查询的问题中

要被索引的文本

```
T0 = "it is what it is'
T1 = "what is it"
T2 = "it is a banana"
```

反向文件索引

```
"a":        {2}//在T2出现
"banana":   {2}//在T2出现
"is":       {0,1, 2}//在T0、T1、T2出现
"it":       {0,1, 2}//在T0、T1、T2出现
"what" :    {0,1}//在T0、T1出现
```

正向索引是文档指向了它包含的那些单词

反向索引是单词指向了包含它的文档

1.文档检索系统，查询那些文件包含了某单词，比如常见的学术论文的关键字搜索

建倒排索引