# 2. 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。

方案 1：

s 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）。

s 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 ）。

s 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。