问题重述：

=======================以下是废话======================

这个问题来自 DNA序列的k-mer index问题。

给定一个DNA序列，这个系列只含有4个字母ATCG，如 *S =*“CTGTACTGTAT”。给定一个整数值*k*，从*S*的第一个位置开始，取一连续*k*个字母的短串，称之为*k-*mer（如*k*= 5，则此短串为CTGTA）， 然后从*S*的第二个位置， 取另一*k-*mer（如*k*= 5，则此短串为TGTAC），这样直至*S*的末端，就得一个集合，包含全部*k-*mer 。 如对序列S来说，所有5-mer为

｛CTGTA，TGTAC，GTACT，TACTG，ACTGT，TGTAT｝

通常这些k-mer需一种数据索引方法，可被后面的操作快速访问。例如，对5-mer来说，当查询CTGTA，通过这种数据索引方法，可返回其在DNA序列S中的位置为｛1，6｝。

**问题**

现在以文件形式给定 100万个 DNA序列，序列编号为1-1000000，每个基因序列长度为100 。

（1）要求对给定k， 给出并实现一种数据索引方法，可返回任意一个k-mer所在的DNA序列编号和相应序列中出现的位置。每次建立索引，只需支持一个k值即可，不需要支持全部k值。

（2）要求索引一旦建立，查询速度尽量快，所用内存尽量小。

（3）给出建立索引所用的计算复杂度，和空间复杂度分析。

（4）给出使用索引查询的计算复杂度，和空间复杂度分析。

（5）假设内存限制为8G，分析所设计索引方法所能支持的最大k值和相应数据查询效率。

（6）按重要性由高到低排列，将依据以下几点，来评价索引方法性能

* 索引查询速度
* 索引内存使用
* 8G内存下，所能支持的k值范围
* 建立索引时间

=======================我是分割线=======================

问题分析：

显而易见，问题分为两个子部分：建立索引(I)，使用索引查询 (II)。要连接这两个子问题，就需要一个中间的索引表Index。建立索引，就是将原始的DNA序列集，转换为索引表。而使用索引查询的过程，就是用指定的k-mer，在索引表中寻找对应的值。首先要考虑的关键问题，是索引表所包含的信息，以及索引表的数据结构。

不管什么形式的索引表，必须包含的基本信息是，包含每一个 k-mer 所对应的序列编号的集合。而注意到 k-mer 共有4^K 种不同的情况，所以索引表必定包含了对应的4K个序列数表Ks。我们又知道，对于每一条长为L的序列S都存在(L-K+1)个子列，最坏的情况下可能分属 (L-K+1) 个 Ks 中，最大的空间占用长度为 (L-K+1)\* N \* elementSize，最少占用空间不会少于N \* elementSize。

而倘若要求索引表同时包含子列出现位置，则每条DNA序列都会产生(L-K+1)条记录，空间占用将恒为(L-K+1)\* N \* elementSize。可见所需空间最多扩大(L-K+1)倍。然而，建立索引时记录位置并不需要额外计算，不会带来时间复杂度的负担。但如果不记录位置，则需要在索引查询后再进行一次字符串查找，即使使用KMP一类的高效算法，实际上也会带来o(m+n)的额外负担(应当认识到，其实只需要对Ks中的序列进行子列查找即可)。根据题目中查询速度尽量快的要求，我们认为后一种选择更加合理。

于是通过分析我们确定，索引表的key为k-mer序列，value包含序列编号和每个序列内的出现位置，形式为(S1, l1),( S1, l2),.. (Sn, l1)的一个链表、定长数组或散列表(即Hash表)。注意到查询时K值是一定的，所以我们只需要记录子列开始的位置即可，结束位置可有li+K运算得到。

确定了索引表的信息内容之后，就要开始考虑索引表的数据结构。索引表必需的功能有：1.建立一个新索引表; 2.输入k-mer字符串，输出对应的value表的位置或头结点。其中，建立索引表的性能是次要因素，索引表查询的效率是最优先考虑的问题。比如目前的trie树可以在O(n)的时间内完成查询，可能能够完成要求。

确认索引表的数据结构之后，子问题(II)便已经自然解决。只要输入给定的k-mer串，通过索引表得到value表的位置，然后输出value表即可。

现在来着重考虑(I)的问题。显而易见，我们需要对每一条S进行单独的分析并将结果写入索引表(Index)中，注意到这是一个高度独立化的过程，可以进行进一步的并行优化。我们现在先关注单条S的索引建立过程。

建立单条索引时，根据前面的分析，我们需要遍历Si的所有共(L+1-K)条K子列，分别作为k-mer，通过Index索引表找到对应的value表，然后将S的序号i以及子列的起始位置li存储入value表中，相当于进行了(L-K+1)次检索和写入的操作。

综上，我们需要实现的算法包括:

1.空索引表Index的建立；

2.空value表的建立；

3.通过Index的查询对应的value表位置或头指针；

4.value表的写入；

5.value表的输出

6.遍历DNA序列S获得(L+1-K)个k-mer子列；

我们容易发现，需要且仅需要完成以上操作，便可以实现从建立索引到通过索引查询的整个过程。建立索引过程为: 1 -> 2 -> 6 -> 3 -> 4，通过索引查询的过程为: 3->5。

基本假设：

1.内存能够容纳至少1个完整的序列记录；

2.外存能够储存索引表所需的空间；

符号说明：

L：单个DNA序列的长度，即100；

K：子列长度，满足1<=k<=100；

Ks：包含每一个 k-mer 所对应的序列编号的集合；

N：DNA序列总个数，即1000000；

S：DNA序列；

Index：索引表；

elementSize：存储S的序号和子列位置所用的数据类型的空间大小；

li：S中第i个子列的起始位置。

模型建立与求解：

然而现在并没有。

模型评价与改进：

事实上，我们的算法还可以有别的实现方式。例如，可以将k-mer作为待检索子串，将所有序列顺次连接作为主串，使用AC自动机或者KMP等算法可以在O(4k(K+L\*N）)时间内进行求解。然而这种算法的缺点在于：

1.高度单线程，不能够很好地并行优化；

2.不同序列的连接处需要特殊的排除处理，实现比较复杂；

3.单次KMP只能够解决一个k-mer的匹配问题，用时并没有优势。

另外，我们获得的检索表，可以稍作处理后用于其他的K值。

记建立检索表时的K值为K0，新的K值为K’，则有以下讨论：

当K’ < K0，我们可以就以检索的k-mer构建包含k-mer且字长为K0的串共(K0-K’+1)\*4(K0-K’)­个，其中k-mer起始于串的第i位。那么，我们只需要对每一个串利用先前构件好的检索表来搜索，然后将标记的位置增加(i-1)即可。

当K’>K0，我们先产生k-mer的字长为K0的子串共(K0-K’+1)个，然后对各个子串分别进行检索，先对序列数取交集(由于检索表数据是有序的，求交运算速度很快)，然后在获得的交集中，查找各子串对应位置偏移量分别增一的case，进行输出，即为所求。

应当注意到，当k|3时，检索k-mer的序列具有实在的生物学意义，即假定所有序列均为内显子情况下，根据密码子确定的氨基酸序列。

//现在我们写的这个东西，是raw material

//最后我们会reorganize出来一个东西，作为最终的论文

//郑明毅曾经表示，raw data永远是越多越好

//这么加注释好二…