封一

答卷编号（参赛学校填写）：

答卷编号（竞赛组委会填写）：

论文题目： （标明A、B、C、D之一）

B

组 别：(填写研究生、本科生、专科生或中学生**)**

本科生

参赛队员信息(必填)：

|  |  |  |  |
| --- | --- | --- | --- |
|  | **姓 名** | **学 号** | **联系电话** |
| **参赛队员1** | **李沅泽** | **1140310520** | **15114558910** |
| **参赛队员2** | **张翔熙** | **6142910605** | **15645160847** |
| **参赛队员3** | **高琦琦** | **1141900219** | **18245020135** |

参赛学校：哈尔滨工业大学

封二

答卷编号（参赛学校填写）：

答卷编号（竞赛组委会填写）：

评阅情况（学校评阅专家填写）：

学校评阅1.

学校评阅2.

学校评阅3.

评阅情况（联赛评阅专家填写）：

联赛评阅1.

联赛评阅2.

联赛评阅3.

# DNA序列的k-mer index 问题模型

作者：李沅泽 张翔熙 高琦琦

# 摘要：

**关键词：**DNA序列 index trie树 分类算法 k-mer

**1.问题重述**

## 自人类基因组计划实施以来,特别是随着第二代测序技术大规模的使用与推广，现今一次深度测序技术可以产生TB级的序列数据。而在对这些海量数据的分析和应用中，k-mer频次统计是一个非常基础且重要的问题。k-mer频次统计信息可以用来揭示生物各种子序列的分布规律, 它是一种衡量序列相似性的重要工具， 因而其在众多的生物学问题上都有着重要且广泛的应用[1]。

## 部分[DNA](http://baike.baidu.com/view/758.htm" \t "_blank)序列或[基因](http://baike.baidu.com/view/8563.htm" \t "_blank)序列使用一串[字母](http://baike.baidu.com/view/94782.htm)表示的真实的或者假设的携带[基因信息](http://baike.baidu.com/view/5178258.htm)的DNA分子的一级结构。可能的[字母](http://baike.baidu.com/view/94782.htm" \t "_blank)只有A，C，G和T，分别代表组成DNA的四种[核苷酸](http://baike.baidu.com/view/117213.htm" \t "_blank)——[腺嘌呤](http://baike.baidu.com/view/62922.htm)，[胞嘧啶](http://baike.baidu.com/view/461057.htm)，[鸟嘌呤](http://baike.baidu.com/view/125105.htm)，[胸腺嘧啶](http://baike.baidu.com/view/476385.htm)。每个[字母](http://baike.baidu.com/view/94782.htm)代表一种碱基，两个碱基形成一个碱基对，碱基对的配对规律是固定的，即是：A-T,C-G。典型的它们无间隔的排列在一起，例如序列AAAGTCTGAC。任意长度大于4的一串[核苷酸](http://baike.baidu.com/view/117213.htm" \t "_blank)被称做作一个序列，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 。

## 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 。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。(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｝）。

**》》加入研究背景**

**2.问题分析**

问题分为两个子部分：（1）建立索引，（2）使用索引查询 。要连接这两个子问题，就需要一个中间的索引表。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记录。

我们记这1,000,000条DNA的所有出现kMer的位置的集合为Ploc，所有出现过的kMer的集合记为PkMer，则PLoc中的每一个元素x都存在唯一y∈PkMer与之对应，故而两个集合之间建立了一个二元关系。建立索引的过程，实际上就是先把这个二元关系找到，然后再找到它的逆关系-1。而使用索引查询的过程，实际上就是对输入的y∈PkMer，经由-1找到PLoc中的原像。同时，我们还需要找到一种方法表示Ploc与PkMer中的元素。

总之，任何一个满足要求的模型，都需要做到以下几点：

1. 表示Ploc中的元素；
2. 表示PkMer中的元素；
3. 表示并正确建立二元关系；
4. 通过找到逆关系-1并存储。

**3.符号说明**

L：单个DNA序列的长度，即100；

N：DNA序列总个数，即1000000；

k：子列长度，满足1<=k<=100；

Ploc：这N条DNA的所有出现kMer的位置的集合；

PkMer：所有出现过的kMer的集合；

：Ploc与PkMer之间的二元关系；

S：DNA序列；

li：S中第i个子列的起始位置。

BUF\_LEN：每个数据块大小

Ln：在所有序列中出现过的k-mer子列种类数，亦即模型三中键树的叶子总数；

Mk：所有长度为k的ACGT序列的组合数；

MN：在N个长为L的DNA序列中，出现的长度为k的子列的个数；

Node：模型三键树所需的节点个数。

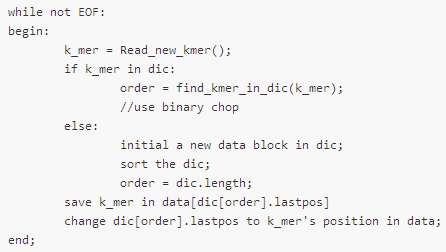
**5.模型建立及求解**

## 建模过程需要实现对DNA序列建立索引，以及搜索过程。现已知数据量较大，为了高效建立索引以及搜索，建立以下三种模型:

1. 模型一——Hash算法/\*好像并没有Hash，直接叫模型一就好\*/

(a).主要代码

次关键字文件 ->　dic 记录所有出现过的不重复K-mer，和最后一次出现在data.txt中的位置  
主关键字文件 -> data 记录所有出现过的K-mer及其位置信息，按照出现顺序排列。



(b).复杂度分析

假设dic文件中将出现最多n条数据，n由k决定。  
n = min{4^k,10^6*(100-k+1)}*

/\*把后面的「最大叶子数」的分析改一改挪到这里来，因为其实它和这里的n是完全一样的东西\*/ *总数据量Len = 10^6*(100-k+1) 经过分析，max(n) 在 **k=14** 处取8.7\*10^7。  
在n确定情况下，因为dic文件数据增长速度和数据有关，假设最坏情况，为加入的前n个k-mer均不相同。  
那么，因为使用希尔排序，因为假定了序列是有序的，所以希尔排序的时间复杂度是 **T(n) = O（n)**。

/\*这个复杂度既没有引文，又没有推导，这里应该加上推导过程\*/  
find\_kmer\_*in\_*dic采用二分算法，时间复杂度是**T(n) = O(log(n))。  
其他操作均为常数时间。 所以每次操作的复杂度可以看成是T(n) = O(n) + O(log(n)) = O(n)**  
因为是最坏情况，所以前n个k-mer均不同，则复杂度应该是  
**T(n) = O(n(n+1)/2) = O(n^2)**

**/\*这里也需要further explanation\*/**  
剩余数据是Len-n, 最坏复杂度为O(n)，则总体算法复杂度  
**T(n) = O(n^2) + (Len-n) \* O(n) = O(n^2)**

2. 模型二

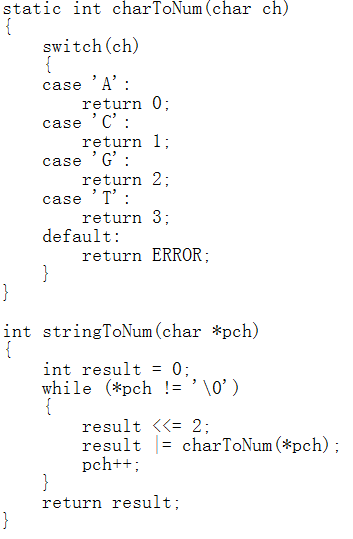
2.1对k-Mer进行编号

#### 用二进制表示每一个碱基:A-->00, C-->01,G-->10,T-->11。对于k-Mer，得到一串01序列，作为这个k-Mer的k-Mer order。如：

#### C T T G A A --> 01 11 10 00 00(480)

#### k-Mer -------> k-Mer order

这个操作，实现了PkMer的表达，完成了问题分析中的第二点。在通过C语言中的位操作可以高效完成：



由循环次数可知，这一步编号操作的时间复杂度为O(k)。

### 2.2把 DNA 序列转换成 k-Mer order 的序列，如：

#### 1326: ACGTTACAGATTTCAGGTTT…

#### 111

#### 1326: ACGTTACAGATTTCAGGTTT…

#### 444

#### 1326: ACGTTACAGATTTCAGGTTT…

#### 753

/\*此处应当加一个插图，说明我们转换之后得到的数据块的存储格式\*/

这一步操作，一方面，通过序列编号和kMer Order在数据块中的出现位置，表示了序列中的一个子列位置，初步完成了问题分析中的第一点；另一方面，建立起了序列中的某一位置与在这一位置上出现的kMer之间的关系，完成了问题分析中的第三点。

此处的时间复杂度为O(N(L-k+1) k)，空间复杂度为O(L-k+1)。

2.3倒排

#### 所谓倒排，就是将之前得到的关系的转换成逆关系-1并存储，实现问题分析中的第四点。

#### 对于每一条DNA得到一个数据块，共有一万N个数据块。每一项数据定义坐标为 (DNA序号 , k-Mer的起始位置) ，这就完全确定了 k-Mer 的一个位置，更明确地完成了问题分析中的第一点。再引入一个常数 BUF\_LEN，表示每一个数据块所包含的数据项个数，在之后的操作中，都将.以一个长度为 BUF\_LEN 的数据块为基本单位。

#### /\*这里应当有一个插图，说明这个数据块的数据结构\*/

#### 将之前得到的数据，按照 k-Mer order 的不同，分别把位置信息装入不同的数组里。在装入的过程中，一旦某个数组装满，就把这个数组的内容放入文件中，作为一个存储起来的数据块；同时，把对应的 k-Mer order计入另一个文件中，作为数据块顺序的记录。两个文件互相参照，就可以根据k-Mer order出现的位置，到存储数据块的文件中读取对应数据。

/\*这里也应该有一个插图，说明这两个文件的具体内容\*/

|  |  |  |
| --- | --- | --- |
| 排序前的索引文件 | 加入一个序号标记 | 计数排序后的结果 |
| 207 | 1，107 | 7，1 |
| 103 | 2，103 | 235，1 |
| 233 | 3，233 | 23，2 |
| 666 | 4，666 | 46，2 |
| 926 | 5，926 | 19，2 |
| 1023 | 6，1023 | 511，3 |
| 1 | 7，1 | 702，4 |

2.4数据块整理

#### 现在得到的数据已经可以进行搜索了。只需要输入 k-Mer，根据前述方法转化为 k-Mer order数字，然后在存储 k-Mer order 的文件(称为索引文件)里找到出现这个序号的位置，然后在存储数据块的文件(称为数据文件)里找对应位置的数据块，即要查的位置信息。然而，这样的搜索需要遍历整个索引文件，在数据量大的时候效率低。一个解决的方法就是对两个文件进行排序处理，排序后可以有较快的算法进行查找定位。

##### 排序后，按照第一个分量从数据文件中读入对应位置的数据块，依次写入到新的数据文件中，就得到了排序后的数据文件。

##### 然后把所有的第二分量存储起来，就得到了新的索引文件。由于对索引文件和数据文件的改动是同步的，所以对于新获得的两个文件，索引文件中每一条 k-Mer order 数据的位置，都对应着数据文件里一个数据块的位置。

##### 按照之前得到的结果，进行查询的过程为：读入待查的 k-Mer → 转换为对应的 k-Mer order →在索引文件中找到这个 k-Mer order 的位置→ 按照这个位置在数据文件中找到对应的数据块。

##### 改进：考虑建立两个数组，一个用来存储每个 k-Mer order 在索引文件中重复的次数，也就是这个 k-Mer 对应了多少个数据块；另一个数组用来储存每个 k-Mer order出现的位置，也就是对应的数据块在数据文件中的起始位置。(O(logn)的查询过程转化为了O(1))

/\*整个这一段可能都需要重新写一下\*/

/\*先略去计数排序的细节，描述如何得到dictionary文件，以及dictionary文件的内容\*/

/\*然后插入计数排序的伪代码，描述如何得到reference文件，以及reference文件的内容，然后分析这一步的复杂度\*/

1.5查询

#### 2.5.1读入要查询的 k-Mer

#### 2.5.2转化为对应的 k-Mer Order

#### 2.5.3在新的索引文件中，查到对应的数据块的数目和起始位置编号

#### 2.5.4在数据文件中，从起始位置开始，读入指定数目的数据块

#### 2.5.5将 (DNA序号, k-Mer起始位置)形式的数据项转化为DNA序号: (起始位置, 终止位置)的格式，写入到文件中，即得到了查询结果。

/\*也要插入复杂度分析\*/

2.6问题与缺陷

#### 2.6.1对数据文件重新排序的过程，在增大数据块大小之后，速度明显提升，但导致倒排过程负担过重

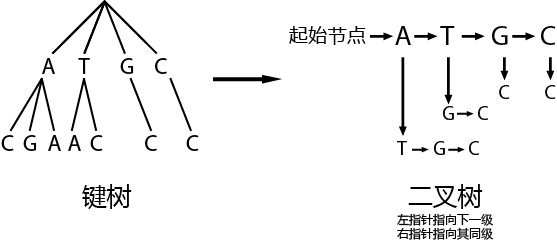
#### 改进：可以增加一个中间文件，记录数据块的索引，从而避免直接对数据文件重排。

#### 2.6.2计算 k-Mer Order时，当k大于14，会出现数据类型溢出的问题。既可以通过改用更大的数据类型来解决，也可以自己另行构建一个数据结构。这个数据结构应该有这样的性质：对于所有出现在 DNA 序列里的 k-Mer，都存在一个编号与之对应；对于相同的 k-Mer，具有相同的编号；这个编号应该在int范围内，并且尽量密集分布。

#### 改进：借助一个不完全的四叉树键树的数据结构来实现。（参考模型三）

3. 模型一三——trie树

3.1树的储存



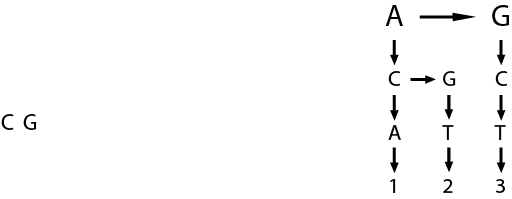
|  |  |  |
| --- | --- | --- |
| int(指向下一级的下标) | char（内容） | int（指向同级的下标） |

节点表示为：

如：tree[3000]

其中，tree[0]不储存元素，表示空节点

|  |  |  |  |
| --- | --- | --- | --- |
| Tree[i] | 下一级 | 内容 | 同级 |
| 0 | 0 | /0 | 0 |
| 1 | 2 | A | 8 |
| 2 | 3 | C | 5 |
| 3 | 4 | A | 0 |
| 4 | 1 | /0 | 0 |
| 5 | 6 | G | 0 |
| 6 | 7 | T | 0 |
| 7 | 2 | /0 | 0 |
| 8 | 9 | G | 0 |
| 9 | 10 | C | 0 |
| 10 | 11 | T | 0 |
| 11 | 3 | /0 | 0 |



ACT = 1

----> AGT = 2

GCT = 3 ……

(剩余步骤参见模型一)

/\*此处应当具体说明一下，建立树的具体算法，和在树上查询的具体算法\*/

2.节点NODE数分析

以ACGTT为例，根 –> A --> C --> G --> T --> T --> /0, 共有k+2层。

(a). 节点数最多的情况

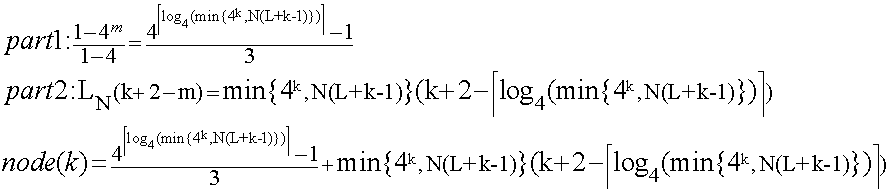
对于一个四叉树，可以分成前后两部分即完全四叉树和不分叉的树，如图：

\* –> \* –> \* –> \* –>………… –> \* –> \* –> \* –> \* –> ………… –> \* –>\*

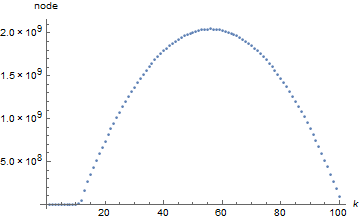
|--------------------->m层<-----------------------||---------------->k+2-m层<----------------|



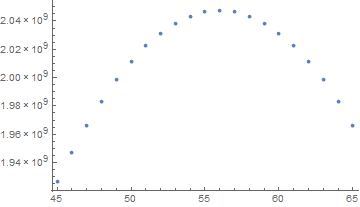
于是，part1和part2的节点个数与总节点个数分别为：



取L=100，N=1,000,000，在Mathematica上可制图如下：



顶点处放大如下：



在k=56时，node达到最大值2047369621。

/\*

注意：4m-1 < Lmax ==> m0 = [log4 Lmax] + 1 (向上取整)

完全四叉树节点数：利用等比数列求和公式，得 (4m-1)/3

不完全四叉树节点数：(K+2-M)/Lmax

因此，总结点数为：node(k) = (4m-1)/3 + (K+2-M)/Lmax (其中，m = m0，Lmax = max{4k,1000000\*(10+k-1)})

\*/

/\*这里应该有一个列表，对所有14~100的k都列出对应的node(k)\*/

(b).期望

E(node) = (4E - 1) / 3 + (k + 2 - E) \* Lmax ，其中，E = [log4 Lmax] + 1 (向上取整)

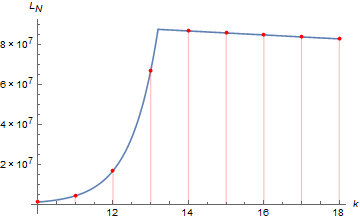
3.叶子个数分析

(a).最大叶子数

/\*这一段应该插入到沅泽的模型一里面，因为他那里最先用到了这个数\*/

已知k-Mer 一共有4k种可能；在100万条记录中k-Mer个数为1,000,000 \* (100 – k + 1)。可知，叶子最大数为：Lmax = maxmin{4k,1000000\*(10+k-1)})

取L和N为100与1,000,000，在Mathematica上绘制图像如下：



可见，当k=14时，Ln取最大值87,000,000（在int类型范围内）。显然，随着N的减小，Ln将更早达到最大值，且最值将更小。

(b).期望

对于下面的分析是对k>=14而言的，对k<14的情况，不一定非要使用键树，所以在此不进行分析。

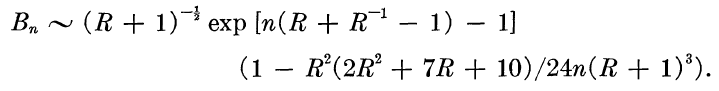
k>=14时，k-Mer的总数为1000000(101-k)条，要叶子个数为L，实际上相当于将这些k-Mer划分成L个集合。而将n个元素划分为m个集合的方法数，为S(n,m)，其中S代表第二类斯特灵数(Stirling Number of the Second Kind)【3】而对n个元素总的划分方法数为贝尔数Bn(Bell Number)，显然有。

C:\Users\Administrator\AppData\Roaming\Tencent\Users\1020563055\QQ\WinTemp\RichOle\VE35K66(~D$L(XBWD[V2A(5.pngC:\Users\Administrator\AppData\Roaming\Tencent\Users\1020563055\QQ\WinTemp\RichOle\[JX$VLR][7AMD@)A3F5KF@Q.png 要求L的期望，实际上就是求S(n,m)对m的概率分布。而又有关系(里面的sigma应该替换成S的表示法)/\*这里的公式应当重新打一下，有不consistent的地方\*/：

其中【2】

令x=0，显然就得到了S(n,m)对m的期望值：Bn+1/Bn-1。

然而，对于较大的n，计算贝尔数的运算量极大，我们采用一个渐进表达式：



/\*这个公式也应该重新打一下，把里面的R换成W(n)\*/

其中，R是方程R exp(R)=n的解，亦即W(n)，其中W为朗伯W函数(Lambert W-Function)。【4】

这是一个非常好的近似估计，在n很小的时候就能达到较高的精度。比如对B5，实际值为52，通过该式的估计值为52.099。

对14~100的k值，算得数值如下:

{14, 5.59913\*10^6},

{15, 5.53864\*10^6},

{16, 5.47812\*10^6},

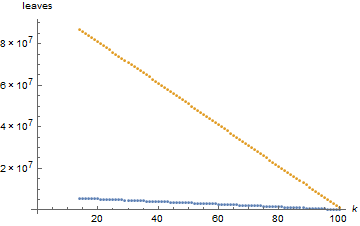
………………

{99, 166362.},

{100, 87846.6}

/\*这里的数据应该改成用列表的形式表示\*/

绘制图形如下：



可见，期望值远远小于估计的最大值，在最大值的6.4%到8.7%之间。而且k与期望叶子数具有良好的线性关系，用最小二乘法拟合得到，，可用于下一步的复杂度分析。

/\*下一步的复杂度分析…应该对后面的二三四步都分析一下的…不要到这里就结束了呀…\*/

5.改进

在建树的过程中，使用了一个储存节点的定长数组。对于一部分k值会超出8G的内存限制。因此提出以下两种方案：

1. 内存没有限制的时候，将该数组长度设为k值对应的最大节点数目。
2. 在8G内存限制下，将该数组长度设定成内存极限的大小。当数组存满时，将内存中的数据全部储存到文件中，重复上述建树过程。在最终搜索k-Mer 时，分别用多个树进行搜索，最后将结果拼接起来。

/\*感觉还是没有太说清楚，需要重新整理表述一下\*/

6.计数排序

！！！！！！！！！我只是没有办法打上，，，回头复制

7．算法复杂度[5]

(a).时间复杂度：O(n + k)

(b).空间复杂度：O(n + k)

注：n约等于N(101 – k)/E(L); k 约等于E(L)

/\*这是应当插入到模型分析里的内容，逻辑上不应该单独分出来。\*/

**5.模型评价及改进**

5.1对k建立索引，对1-k所有k值都能查询

/\*此处应该加一段详细的说明\*/

5．2 8G内存对应100万条数据截取得50万条数据，两个树，分别建立索引

6.参考文献

【2】L. H. Harper. “Stirling Behavior is Asymptotically Normal” Ann. Math. Statist. Volume 38, Number 2 (1967), 410-414)

【3】Abramowitz, M. and Stegun, I. A. (Eds.). "Stirling Numbers of the Second Kind." §24.1.4 in Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing. New York: Dover, pp. 824-825, 1972。

C:\Users\Administrator\AppData\Roaming\Tencent\Users\1020563055\QQ\WinTemp\RichOle\7%[)0LUO6T)4$}5}X8X120P.png

1. Thomas H.Cormen,Charles E.Leiserson,ect.算法导论（原书第三版）.北京：机械出版社pp193-195S

/\*引文格式需要统一\*/