第 33 卷 第 9 期 计 算 机 工 程 2007 年 5 月
Vol.33 No.9 Computer Engineering May 2007
·博士论文· 文章编号：1000—3428(2007)09—0040—03 文献标识码：A 中图分类号：TP311

# k-长 DNA 子序列计数算法研究

王树林[1,2]，王 戟[1]，陈火旺[1]，张鼎兴[1]
（1. 国防科技大学计算机学院，长沙 410073；2. 湖南大学计算机与通信学院，长沙 410082）

**摘 要：**基因组的结构与功能存在密切联系，其功能主要通过 DNA 子序列来表达，因此研究 DNA 序列结构对于生物信息学来说具有重要的意义。该文研究了 k-长 DNA 子序列在 DNA 全序列中出现频数的计数问题，设计并实现了 k-长 DNA 子序列内部计数算法和外部计数算法。该算法通过一个哈希函数把 k-长 DNA 子序列映射为整数关键字从而把 k-长 DNA 子序列出现频数的计数问题转化为整数关键字的重复计数问题，使得能够利用经典 B 树算法来解决 k-长 DNA 子序列的出现频数计数问题。针对所要解决的问题提出 3 种改进措施以进一步提高算法的性能。

**关键词：**k-长 DNA 子序列；DNA 序列；B 树；全基因组

# Research on Counting Algorithm of k-mer Occurrence in DNA Sequence

WANG Shulin[1,2], WANG Ji[1], CHEN Huowang[1], ZHANG Dingxing[1]
(1. School of Computer Science, National University of Defense Technology, Changsha 410073;
2. College of Computer and Communication, Hunan University, Changsha 410082)

**【Abstract】** There is a close relationship between the structures of whole genome and its functions which are expressed by its subsequences. Researching the structure of DNA sequence has a profound meaning to bioinformatics. The problem that all k-mers in whole genome are counted is researched. The internal and external algorithm which counts all k-mers occurrence in DNA sequences is designed and implemented. This algorithm translates the problem of counting all k-mers into the problem of counting integer keys with the help of a hash function which maps a k-mer to an integer, and it applies the classic B-tree algorithm to solve the problem of counting k-mers in DNA sequence. It proposes three measures to further improve the efficiency of the algorithm according to the feature of the counting problem.

**【Key words】** k-mer; DNA sequence; B-tree; Whole genome

生物序列主要是指 DNA、RNA 和蛋白质序列，其中 DNA 序列是生物遗传信息的存储和控制中心，对它们的研究有望揭示生命的奥秘，因此研究 DNA 序列的结构模式与功能之间的关系是目前生命科学中一个至关重要的基本问题。生物序列测序技术的进步为快速获得生物全基因组序列提供了有利的技术保证。目前测出的全基因组序列呈爆炸式增长，对目前计算机的处理能力提出了严峻挑战，同时也促进了计算机科学的发展。

生物学家们想知道 DNA 全序列中哪些子序列是频繁出现的，出现多少次；哪些子序列是不出现的，出现最频繁的子序列是什么；不同 DNA 序列中频繁出现的子序列是否存在相似特征；哪些子序列的出现是稀少的；是否存在标识种属的子序列等，对这些问题的研究有助于生物序列功能和进化模型的研究。因此研究 DNA 全序列中所有子序列出现频数的计数问题具有重要的生物学意义。鉴于此，本文设计并研究了 DNA 全序列中所有 k-长 DNA 子序列出现频数的内部和外部计数算法。虽然这个问题初看起来是一个简单的问题，但由于单个 DNA 全序列的长度从几百万碱基对到几千亿碱基对不等，因此这直接导致该计数算法的实现难度。

## 1 相关工作

Ferragina和Grossy结合B树和Patricia树的特点提出一种利用外存来处理字符串匹配的字符串树SB-Tree(String B-Tree)的算法，其优点是能够处理不受限长度的关键字符串并具有强大的字符串搜索能力[3]。Jeong-Hyeon Choi等对SB-tree数据结构进行改进提出一种能够快速有效分析k-长DNA子序列的SSB-tree(Static SB-tree)，并开发了一个软件平台SequeX，它具有强有力的数据查询、文件管理和序列可视化以及k-长DNA子序列计数和检索等功能。郝柏林院士等于2000 年提出一种基于k-长DNA子序列出现频数的能产生 2D 分形图像的序列可视化方法[1,2]，实际上这也是一种统计子序列出现频数的计数方法，但它只能统计k值较小时的k-长DNA子序列的出现频数。受此方法的启发，设计了一个基于B树的快速统计k-长DNA子序列在DNA全序列中出现频数的计数算法。

## 2 描述符号与问题定义

DNA 序列是由 4 种核苷酸构成的线性序列，这 4 种核苷酸可以看成字母表集合 $\Sigma$，则 $\Sigma=\{a, c, t, g\}$。长度为 $n$ 的 DNA 序列记为 S[1…n]，称为 n-长序列，则 $S[1\ldots n]=S[1]\ldots S[j]\ldots S[n]$，其中 $S[j]\in\Sigma, 1\le j\le n$。所有不同 n-长序列构成序列空间 $\sum^n$，则 $|\sum^n|=4^n$。序列 S[1…n]的长度为 k 的子序列称之为 k-长

**基金项目：**国家自然科学基金资助项目(60233020)
**作者简介：**王树林(1966－)，男，博士生，主研方向：生物信息学，软件工程和复杂系统；王 戟，博士、教授、博导；陈火旺，教授、博导、中国工程院院士；张鼎兴，副教授
**收稿日期：**2006-06-27 **E-mail：**jt_slwang@hnu.cn

DNA 子序列，记为 $S[j+1\ldots j+k], 0\le j\le n-k$，且 $k\le n$。序列 S[1…n]的所有 k-长 DNA 子序列构成多重集 $C^k$，则 $|C^k|=n-k+1$。

一个 DNA 全序列长度记为 $L$，从生物学可知 $L$ 的取值范围约在 $O(10^3)$ 到 $O(10^{11})$ 量级之间。通常 $k<<L$，所以认为 DNA 全序列 S[1…n]的 $C^k$ 中的元素个数约为 $L$。假设地球上所有物种的 DNA 全序列构成一个集合，记为Ω，则 $\Omega=\{S^i[1\ldots L^i]\,|\,i\in N\}$，其中 $i$ 唯一标示一个物种，$L^i$ 表示物种 $i$ 的 DNA 全序列长度。

**定义 1** 采用宽度为k的滑动窗口 $W_k$ 沿DNA全序列 $S^i[1\ldots L^i]$ 从 5' 端以步长为1个碱基字母的方式依次移动到 3' 端，则可以把该DNA全序列分解成由k-长DNA子序列构成的多重集合 $C_i^k$，即 $C_i^k=\{S^i[j+1\ldots j+k]\,|\,0\le j\le L^i-k\}$，且 $\left|C_i^k\right|=L^i-k+1$。一个k-长DNA子序列在 $C_i^k$ 中出现的次数称为该k-长DNA子序列在 $S^i[1\ldots L^i]$ 中的出现频数 $f_m$，下标 $m\in N$ 唯一标示该k-长DNA子序列，显然 $f_m\ge 0$，$f_m=0$ 表示标示为 $m$ 的k-长DNA子序列不在 $S^i[1\ldots L^i]$ 中出现。

**定义 2** 映射 $M:\Sigma\mapsto Binary$ 按下式定义，其中 *Binary* 表示只有两位的二进制集合。

$$M(x)=\begin{cases}00 & x=c\\ 01 & x=t\\ 10 & x=a\\ 11 & x=g\end{cases}$$

**定义 3** 哈希函数 $h:\sum^k\mapsto N$ 的定义为

$$h(S[1\ldots k])=\sum_{j=1}^{k}4^{k-j}M(S[j])$$

其中，S[$j$]∈Σ, S[1…k]∈$\Sigma^k$。称h(S[1…k])为S[1…k]的关键字$m$，由 $C_i^k$ 经$h$映射成的关键字多重集记为 $K_i^k$。

**定义 4** 对定义 3 中的哈希函数 $h$ 进行改造可把 k-长 DNA 子序列映射为二维空间中的点(x, y)。

$$\begin{cases}x=\sum_{j=1}^{k}2^{k-j}low(M(S[j])), & S[j]\in\Sigma\\ y=\sum_{i=1}^{k}2^{k-j}high(M(S[j])), & S[j]\in\Sigma\end{cases}$$

其中，函数 low 表示取二进制的低一位，函数 high 表示取二进制高一位。

**定义 5** 所要解决的问题可形式化地定义为：对于 $\forall i$，把 $S^i[1\ldots L^i]$ 按滑动窗口 $W_k$ 分解为 k-长 DNA 子序列多重集合 $C_i^k$，再按定义 3 把 $C_i^k$ 映射为 $K_i^k$。则计算 $C_i^k$ 中所有不同 k-长 DNA 子序列的出现频数 $f_m$ 就转化为计算 $K_i^k$ 中所有不同关键字的出现频数 $f_m$。由关键字和其出现频数组成的二元组构成频数集 $F$，则

$$F=\{(m,f_m)\,|\,m=h(S^i[j+1\ldots j+k],0\le j\le L^i-k)\}$$

$F$中的元素不可能相同，因为相同关键字已经合并，并计算了其出现频数$f_m$。

**定义 6** 与排序算法的分类类似，k-长 DNA 子序列计数算法根据其实现过程是否借助于外存来完成可分为两类：内部计数算法和外部计数算法。内部计数算法是指算法实现的整个过程全部在内存中进行，而外部计数算法的实现需要把数据的一部分存储在内存，另一部分存储在外存。

## 3 内部计数算法

### 3.1 算法思想

内部计数算法的思想描述如下：

Input：给定一个物种 $i$ 的序列 $S^i[1\ldots L^i]$ 及其滑动窗口宽度 $k$。

Output：k-长 DNA 子序列出现频数集合

$F=\{(m,f_m)\,|\,m=h(S^i[j+1\ldots j+k],0\le j\le L^i-k)\}$

$F=\phi$；/*置 k-长 DNA 子序列出现频数集合为空集*/

将 $W_k$ 置于 $S^i[1\ldots L^i]$ 起始 5' 端。

$j$=0；

While $j\ne L^i-k$ Do

Begin

$m=h(S^i[j+1\ldots j+k]$；

/*按定义 3 把 $W_k$ 中的 k-长 DNA 子序列映射为一个整数*/

If $(m,*)\notin F$ then

/*在集合 $F$ 中查找关键字 $m$ 是否出现*/

$f_m=1$，并把 $(m,f_m)$ 加入到出现频数集合 $F$ 中；

else

对 $F$ 中的元素 $(m,f_m)$ 中的 $f_m$ 加 1；

$j$=$j$+1 /*把 $W_k$ 移动一个碱基*/

End;

输出子序列出现频数集 $F$；

### 3.2 算法性能分析

对上述算法的时间与空间复杂度的分析取决于算法实现时所采用的数据结构。如果采用二维数组 $2^k\times 2^k$ 来表示子序列出现频数集合 $F$，采用定义 4 映射的两个关键字作为数组下标，其元素值为频数，则算法的时间复杂度为 O(L)，L 为 DNA 序列的长度，空间复杂度为 $O(4^k)$，k 为子序列长度。如果采用单向链表来表示 $F$，则算法的时间复杂度为 $O(\frac{1}{2}L^2)$，空间复杂度为 O(3L)。

更好的方法是采用二叉平衡树、B 树等数据结构来存储 $F$，算法总体性能会更好一些。但在 $k$ 值较大和 $L$ 值也很大时，算法的性能是非常低的，主要原因是 k-长 DNA 子序列关键字不能完全存放在内存中，运行的大部分时间用在频繁的内外存交换上。例如在具有 256MB 内存的机器上运行该算法，处理全长 $L$ 为 5 231 428bp 的 Escherichia coli DNA 序列，则当 $k\le 12$ 时，算法的效率是很高的，但当 $k\ge 13$ 时，算法效率骤然下降以致无法运行获得结果。因此内部计数算法只适合于解决 k 值不大时的 k-长 DNA 子序列的出现频数计数问题。设计高效计数算法的关键是解决内外存交换所引起的算法效率不高的问题。

## 4 外部计数算法

### 4.1 算法实现

1970 年，R.Bayer 和 E.McCreight 提出了一种适用于外查找的树，它是一种平衡的多叉树，称为 B 树，一棵 m 阶的 B 树满足下列条件：(1)树中每个结点至多有 m 个孩子；(2)除根结点和叶子结点外，其它每个结点至少有 m/2 个孩子；(3)若根结点不是叶子结点，则至少有 2 个孩子；(4)所有叶子结点都出现在同一层，叶子结点不包含任何关键字信息；(5)有 k 个孩子的非终端结点恰好包含有 k-1 个关键字。对 B 树节点数据结构稍加改造，定义如图 1 所示，其中 $k_j(1\le j\le n)$ 表示子序列关键字域，且满足 $k_i<k_j(1\le i<j\le n)$，$f_{k_j}(1\le j\le n)$ 表示关键字 $k_j$ 的出现频数域，$p_j(0\le j\le n)$ 表示孩子节点的地址域。在往 B 树中插入关键字 $m$ 时，首先在 B 树中查找关键字 $m$，如没有找到则插入，否则使相应关键字 $m$ 的频数域 $f_m$ 加 1，因此 B 树中没有重复关键字。

| $p_0$ | $k_1\ f_{k_1}\ p_1$ | …… | $k_j\ f_{k_j}\ p_1$ | …… | $k_n\ f_{k_n}\ p_n$ |
|---|---|---|---|---|---|

**图 1 B 树节点的数据结构**

例如，假设物种 $i$ 的 DNA 序列 $S^i[1\ldots L^i]$ = 5'-ggtctctctggttagaccagatctgagcctgggagctctc-3'，这里 $L^i$=40，取 $W_k$ 的宽度 k=4，则该 DNA 序列经哈希函数映射成为多重关键字集 $K_i^4$ 为：{244,209,68,17,68,17,71,31,125,245,214,91,110,184,224,130,11,46,185,228,145,71,30,123,236,176,193,7,31,127,254,251,236,177,196,17,68}，构造 5 阶 B 树如图 2 所示。

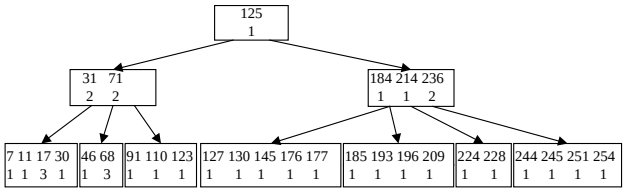


**图 2 5 阶 B 树结构**

外部计数算法思想与内部计数算法思想类似，简要描述如下：

$B=\phi$；/*创建一棵空 B 树*/

将滑动窗口 $W_k$ 置于序列 $S^i[1\ldots L_i]$ 起始 5' 端。

j=0;

While $j \neq L_i - k$ Do

Begin

$m = h(S^i[j+1\ldots j+k]$；

If $m \notin B$ then /*在 B 树中查找关键字 $m$*/

Begin

插入关键字 $m$，并 $f_m = 1$；

检查是否分裂节点；如果是则分裂；

End

else

相应关键字 $m$ 的频数 $f_m = f_m + 1$；

j=j+1

End;

采用宽度或深度优先遍历 B 树，输出出现频数集 $F$；

### 4.2 算法性能分析

构建 B 树的过程主要涉及关键字的查找、插入和节点分裂；关键字查找又分两步：(1)根据待查找关键字定位节点；(2)在该节点内部进行二分查找。在这些操作中，只有节点定位是需要访问磁盘的，而其它所有操作都是在内存中进行的，故只考虑最费时的节点定位操作的时间复杂度。

实际上 B 树中最多包含 L 个关键字，根据定义，m 阶 B 树的高度 $h \le \log_{\lfloor m/2 \rfloor}((L+1)/2)+1$。则算法最多访问 $\sum_{j=1}^{L}(\log_{\lfloor m/2 \rfloor}((j+1)/2)+1)$ 次磁盘，这个和式不会超过 $L(\log_{\lfloor m/2 \rfloor}((L+1)/2)+1)$。

### 4.3 外部计数算法改进

改善外部计数算法性能的关键是提高磁盘访问的效率，尽可能减少无谓的磁盘访问次数。分析外部计数算法可知，B 树是一次性批量插入关键字生成的，不涉及关键字的删除和修改等问题。针对这一特点，提出下面 3 种改进措施以提高算法效率。

(1)溢出思想：Bayer 和 McCreight 提出的溢出思想能够尽可能地避免频繁地分裂节点，从而提高插入效率。

(2)采用B*树：Knuth在 1973 年提出的B*树是B树的变形[6]。B*树至少利用了每个节点中 2/3 的可用空间，而不像B树那样只利用了节点一半的空间，使得树的高度降低，查找过程加快。

(3)磁盘缓冲：虽然磁盘缓冲策略是解决磁盘扇区频繁读写的有效方法，但其效率取决于磁盘读写的局部性。由于 DNA 序列的随机性，因此磁盘读写的局部性不一定能满足。但是如果采取分批预读措施，把预读的部分 DNA 序列映射为关键字序列后存储在数组中再排序，后再批量插入到 B 树中，从而增加了磁盘读写的局部性，因而能够提高计数算法的效率。当查找或插入算法发出“虚拟读”命令后，仅当需要的页不在内存中时，它才被翻译成真正的读磁盘命令。缓冲区的页替换策略可以采用最近最少使用方案。

## 5 结束语

由于 DNA 序列在长期的进化过程中形成了其独特的生命元件之间复杂的相互作用关系以及 DNA 序列数据量的异常庞大，致使生物信息学中的许多问题都是对目前计算机处理能力的挑战，需要针对生物序列的特点设计性能卓越的算法来解决。本文设计并实现了 k-长 DNA 子序列在 DNA 全序列中出现频数的内部和外部计数算法，较好地解决了 k-长 DNA 子序列出现频数的计数问题。对这个问题的研究能解决许多生物信息学问题，为生物学家分析 DNA 子序列分布、研究序列进化提供重要帮助。我们利用这个算法研究了 k-长 DNA 子序列的频数分布、k-长 DNA 子序列的最大出现频数与 k 值的关系等生物信息学问题，发现了一些新的规律。

## 参考文献


1 Hao Bailin. Fractals from Genomes——Exact Solutions of a Biology-inspired Problem[Z]. 2000. http://power.itp.ac.cn/~hao/scitlist.htm.

2 Hao Bailin, Lee H C, Zhang Shuyu. Fractals Related to Long DNA Sequences and Complete Genomes[Z]. 2000. http://power.itp.ac.cn/~hao/scitlist.htm.

3 Ferragina P, Grossi R. The String B-tree: A new Data Structure for String Search in External Memory and Its Application[J]. Journal of ACM, 1999, 46(2): 236-280.

4 Delcher A L, Kasif S, Fleischmann R D. Alignment of Whole Genomes[J]. Nucleic Acids Res., 1999, 27(11): 2369-2376.

5 Choi J H, Hwangue C. Analysis of Common k-mers for Whole Genome Sequences Using SSB-tree[Z]. 2002. http://www.jsbi.org/journal/GI13.html.

6 Donald E K. The Art of Computer Programming[M]. Addison Wesley Longman, 1998.

7 Kunihiko S, Tetsuo S. Indexing Huge Genome Sequences for Solving Various Problems[Z]. 2001. http://www.jsbi.org/journal/GI12.html.


（上接第 39 页）

## 参考文献


1 NetGEO. A NetGeo White Paper[Z]. 2005. http://www.netgeo.com/.

2 David M, Ram P. Where in the World is netgeo.caida.org[Z]. 2000. http://netgeo.caida.org/.

3 Ram P, Evi N G. A Graphical Traceroute Tool[Z]. 1999. http://www.caida.org/tools/visualization/gtrace.

4 VisualRoute[Z]. 2006. http://www.visualware.com/visualroute/index.html.

5 Meyer D. University of Oregon Route Views Project[Z]. 2004. http://www. antc.uoregon.edu/route-views/.

6 Getty Thesaurus of Geographic Names[Z]. 2000. http://www.getty.edu/research/conducting_research/vocabularies/tgn/index.html.