基于字符和 0/1 码的 DNA 压缩模式匹配算法*

张丽霞[a], 张义青[a], 林丕源[a], 刘吉平[b]
(华南农业大学 a. 计算机科学与工程系; b. 动物科学学院, 广州 510642)

摘　要: 分析对 DNA 序列数据进行压缩和压缩模式匹配的重要性, 采用 0/1 编码的非自适应算法进行压缩, 提出两类压缩模式匹配思路, 设计实现了四种算法, 并进行了性能比较。

关键词: DNA 序列; 数据压缩; 压缩模式匹配
中图分类号: TP301　　**文献标志码**: A　　**文章编号**: 1001-3695(2007)09-0022-03

DNA compressed pattern matching algorithms based on character and 0/1 coding

ZHANG Li-xia[a], ZHANG Yi-qing[a], LIN Pi-yuan[a], LIU Ji-ping[b]
(a. Dept. of Computer Science & Engineering, b. College of Animal Science, South China Agriculture University, Guangzhou 510642, China)

Abstract: This paper analyzed the importance of compression and compressed pattern matching for DNA sequence data. Coding was used in no-adaptive-compression algorithm, proposed two schemes of compressed pattern matching, designed and implemented four algorithms, compared their performances.

Key words: DNA sequence; data compression; compressed pattern matching

随着生物序列数据的日益增加, 数据占用的存储空间也日益增大。如何在有效的存储空间存储和使用生物序列数据, 成为计算机专家和生物学家必须面临的问题。就 DNA 序列数据的应用而言, 一方面需要考虑数据的存储效率, 另一方面也要考虑数据的检索速度。考虑数据的存储效率, 需要将 DNA 序列数据压缩后再进行存储, 这样可以在有限的存储设备上存储更多信息。由于 DNA 序列数据的特殊性, 即 DNA 序列数据由 A、C、G、T 四个字母组成, 并且 DNA 序列长度可达上百万个碱基对, 使用传统的数据压缩算法并不理想, 需要研究专门针对 DNA 序列数据的压缩算法。考虑数据的检索速度, 一方面要设计好的检索算法, 另一方面要考虑如何使用压缩后的数据。通常使用压缩数据的方法是先对数据解压缩, 再在解压缩数据上进行检索。采用这种方式, 在计算机上实际使用压缩数据时, 并不是真正意义上地减少空间, 也不能改善检索速度。若能够在压缩数据上直接进行检索, 既提高了检索速度, 又大大节省了使用时的存储空间, 有助于将 DNA 序列数据迁移到普通性能的工作平台(如笔记本电脑或 PDA) 上, 使得生物学家可以随时随地开展研究工作。

在压缩数据上直接进行信息检索的实质是对压缩数据直接进行模式匹配, 称为压缩模式匹配[1]。这已引起计算机及相关学科研究人员的关注。本文将介绍一类专门针对 DNA 序列数据的压缩算法和压缩模式匹配算法。

1　DNA 压缩算法

从编码的角度可以将压缩算法分成两类, 即非适应性编码的压缩算法和适应性编码的压缩算法[2]。所谓非适应性编码, 是指相同的字符得到的压缩编码一样; 适应性编码, 是指相同的字符可能会有不一样的压缩编码。

考虑到 DNA 序列数据仅由 A、C、G、T 四个字母组成, 采用一种非自适应编码方法来压缩 DNA 文本串和模式串。用两个二进制位 00、01、11、10(称为编码单元) 来编码字符 A、C、G、T, 即 A 编码成 00, C 编码成 01, G 编码成 11, T 编码成 10[3]。例如 DNA 序列 GACCGTCT 可编码成 11 00 01 01 11 10 01 10。

1.1　压缩 DNA 文本串

使用上述编码机制, 用两位二进制数来表示一个八位字符, 因此长度为 m 字节的 DNA 文本串 T 可以压缩成 $\lceil m/4 \rceil$ 字节的 T'。但是如果 $m \bmod 4 \neq 0$, T 的最后一个字节仅包含($m \bmod 4$) 个编码单元, 即($m \bmod 4$) ×2 位。例如 DNA 序列文本串 TACCGT 被编码成 10 00 01 01 11 10, 最后一个字节仅包含两个编码单元, 即四位。后面进行模式匹配是以字节为单位读入编码流的, 因此不足 1 Byte 的需要进行补位, 通常用 x 来补位(xx 称为不需关注的编码单元)。

使用这种压缩机制, 一个 DNA 序列文本串 T 可以被表示成 $\langle T', A\rangle$, 其中 T 是压缩后的文本串; A 是一个非负整数, 表示将 T 用 x 补位后, 所生成的不需要关注的编码单元个数。显然, A 的取值只可能是$\{0, 1, 2, 3\}$。上例用 x 补位后的编码变成 10 00 01 01 11 10 xx xx, 则可表示成 $\langle T', 2\rangle$。其中 T'= 10 00 01 01 11 10 xx xx。

这种简单的编码压缩方法, 可以节省约 75% 的存储空间。

1.2　压缩 DNA 模式串

本文的目标是在压缩后的 DNA 序列数据上直接进行检

收稿日期: 2006-06-15; 修返日期: 2006-09-05　　基金项目: 国家自然科学基金资助项目(60573043); 华南农业大学新学科扶持基金资助项目(4900-K03208, 5600-K04098)

作者简介: 张丽霞(1977-), 女, 讲师, 主要研究方向为生物信息学、人工智能; 张义青(1976-), 女, 讲师, 主要研究方向为生物信息学、数据库应用; 林丕源(1963-), 男, CCF 高级会员, 教授, 主要研究方向为生物信息学、信息安全、软件构件技术(pylin4882@21cn.com); 刘吉平(1968-), 男, 副教授, 硕导, 主要研究方向为昆虫病理学、生物信息学.

索,因此需要对模式串也进行相应的压缩处理。如果按照处理 DNA 文本串的方式,即如果需要补位,在最后一个字节进行补位,匹配时可能会出现漏配的情况。为了避免这种错误发生,在对压缩模式串进行补位时,其位置可以是开头字节,也可以是结束字节,或前后均有。用 P 表示模式串,则压缩后的格式可以表示成 $\langle P', A_1, A_2\rangle$。其中: P 是压缩后的模式串; A_1 是非负整数,表示 P 的第一个字节中无须关注的编码单元个数; A_2 也是非负整数,表示 P 的最后一个字节中关注的编码单元个数。A_1 和 A_2 的取值范围仍然只可能是{0, 1, 2, 3}。

这样,模式串"ACTGA"压缩后的形式有 $\langle P_1', 0, 3\rangle$, $\langle P_2', 1, 2\rangle$, $\langle P_3', 2, 1\rangle$, $\langle P_4', 3, 0\rangle$四种。其中: P_1' = 00 01 10 11 00 *xx xx xx*; P_2' = *xx* 00 01 10 11 00 *xx xx*; P_3' = *xx xx* 00 01 10 11 00 *xx*; P_4' = *xx xx xx* 00 01 10 11 00。

2　DNA 压缩模式匹配算法

对应上述 DNA 压缩算法,有两类压缩模式匹配思路:

a) 将压缩串以字节为单位,转换成一个新的字符,例如 01010010 转换成十进制是 82, ASCII 码 82 对应的字符是 R,则将 01010010 转换成 R。模式串进行相同的处理,用转换后的新字符串进行匹配。这种机制的本质是字符与字符之间的匹配,因此可以对一些经典的模式匹配算法稍作改变,来进行匹配工作。

b) 保持压缩串的二进制码形式,以一个字节为单位,直接对二进制码进行匹配。这种机制本质是 0/1 码之间的匹配,可以直接使用一些数学运算来进行匹配工作。

2.1　基于字符的 DNA 压缩模式匹配算法

KMP 算法是模式匹配领域应用比较广泛的算法之一,可以在 $O(m+n)$ 的时间数量级上完成串的模式匹配操作(m 和 n 分别是模式串的长度和文本串的长度)[4]。下面以 KMP 算法为例,来说明基于字符的压缩模式匹配思路。

2.1.1　KMP 模式匹配算法

算法基本思想[4]如下:

a) 根据模式串计算 next 函数值。

```
void get-next( Sstring P, int next[ ] ) {
//求模式串 P 的 next 函数值并存入数组 next
i = 1; next[ 1] = 0; j = 0;
While( i < P[ 0] ) {
   If( j = = 0 || P[ i] = = P[ j] ) { + + i; + + j; next[ i] = j}
   Else j = next[ j] ;
}
}
```

b) 将文本串 T 和模式串 P 左对齐, 逐个字符进行比较。当模式串的第 j 个字符与文本串的第 i 个字符不匹配时, 计算 next [j], 令 j = next [j], 即模式串右移 next [j] 大小,继续匹配。

c) 匹配结束时,若 j 值大于 P 的长度,则匹配成功,匹配成功的位置是(i - P 的长度) ;否则,匹配失败。

2.1.2　基于 KMP 的 DNA 压缩模式匹配算法

将 KMP 模式匹配算法应用到压缩的 DNA 数据中,可以设计对应的 DNA 压缩模式匹配算法,称为 DNA-KMP。该算法的基本思路是: 读入压缩后的 DNA 文本串和模式串,每 8 位作为一个单元,计算出其 ASCII 码值所对应的字符,这样匹配时的字母表就由原来的 4 个字符变成了 256 个字符(ASCII 码个数)。

压缩后的文本串格式为 $\langle T', A\rangle$,模式串格式为 $\langle P', A_1, A_2\rangle$,在文本的末尾字节和模式的开头及末尾字节都可能存在补位符。因此,将压缩后的文本串格式重新设置为 $\langle T_f, T_b, A\rangle$。其中: T_b 表示原 T 的最后一个字节; T_f 表示剩余部分,则补位符仅存在于 T_b 中。将压缩后的模式串重新设置为 $\langle P_f, P_m, P_b, A_1, A_2\rangle$。其中: P_f 表示原 P 中的首字节; P_b 表示原 P 中的末字节; P_m 表示剩余部分,则补位符仅存在于 P_f 和 P_b 中。匹配过程分为两步:

a) 运用 KMP 算法匹配 P_{m_i}和 T_f,根据对模式串的压缩机制,这里会有四个 P_m,分别与 T_f 进行匹配。如果某个 P_{m_i}与 T_f 匹配成功,则转 b) ;否则,说明整个模式串都不会在文本串中出现。

b) 取 a) 中匹配成功的 P_m 所相关的 P_f 和 P_b,将 P_f 和 P_b 分别与 $T_f[i]$ ($0 \le i \le n'-2$) 进行比较,或将 P_b 与 T_b 进行比较。若能够匹配,则找到了模式串在文本串中的一次出现;否则,根据 KMP 算法的跳跃函数,将模式串移动相应距离后,继续进行匹配工作。

同理,可以将其他经典模式匹配算法,如 BM[5]、Horspool[6] 等算法应用到 DNA 压缩模式匹配中,分别实现对应的 DNA 压缩模式匹配算法 DNA-BM 和 DNA-Horspool。

2.2　基于 0/1 码的 DNA 压缩模式匹配算法

对压缩后的文本串和模式串的 0/1 码形式直接进行匹配,以字节为单位,将文本串和模式串对应作 0、1 的异或操作。为了实现方便,这里对前述压缩算法的补位进行调整,对于文本串和模式串都只补在最后一个字节,匹配过程中通过四次移位模式串来避免漏配。该算法称为 DNA-match。

2.2.1　预处理模式串

为了在匹配过程遇到不匹配情况时,使得模式串能够尽量多地移动,避免重复匹配操作,需要事先对模式串进行预处理,定义一个跳跃表。

8 位 0/1 组合的取值为 0 ~255。因此跳跃表定义为整型数组 step[256] 。将模式串每字节转换成一个十进制整数作为数组下标,每个数组元素的初始值均为 $m'/8$。m 为模式串(压缩后) 的长度,赋值为针对尾字节的偏移。例如:

10101000	01111100	10001011	11000000	01111111
168	124	139	192	127
4	3	2	1	0

step[168] = 4, step[124] = 3, step[139] = 2, step[192] = 1, step[127] = 0

这样,当匹配失败时,在模式串中寻找与不匹配处文本串值相同的字节,即 step 数组的下标;然后根据数组值,移动模式串,使相同的字节对齐,使得模式串的移动尽可能大,而不会有错漏。

2.2.2　DNA-match 压缩模式匹配算法

以字节为单位,将压缩后的文本串与模式串对齐,对应的 0/1 码进行异或运算。根据异或运算的本质, 0^1 = 1, 1^0 = 1, 1^1 = 0, 0^0 = 0。算法过程如下:

a) 将压缩后的文本串与模式串对齐,从模式串的最右码

字开始,从右至左逐位进行异或操作。若结果为 0,说明匹配成功,直接转 b);否则,以最右对齐的不匹配单元为准,查找跳跃表,根据跳跃值将模式串整体右移,继续匹配,直至文本末端。

b) 将模式串整体右移两位,与文本串对齐,即将模式串第一个单元的第一个码字对齐文本串第一个单元的第三个码字;之后重复 a) 的过程。

c) 将模式串继续整体右移两位,与文本串对齐,即将模式串的第一个单元的第一个码字对齐文本串第一个单元的第五个码字;之后重复 a)。

d) 将模式串继续整体右移两位,与文本串对齐,即将模式串的第一个单元的第一个码字对齐文本串第一个单元的第七个码字;之后重复 a)。

经过上述四个步骤,能够找到所有的匹配位置。将模式串右移两位,是为了弥补匹配过程中是以字节为单位,而且压缩时仅在模式串的最后一个字节补位,所可能引起的漏配情况。

3 算法比较

DNA-KMP、DNA-BM、DNA-Horspool 的实验结果对比如表 1、2 所示。KMP、BM 和 Horspool 指对于文本串和模式串不压缩而进行模式匹配。DNA-KMP、DNA-BM 和 DNA-Horspool 指采用前述方法压缩后直接进行模式匹配。

表 1 比较次数对比

文本,模式	BM	DNA-BM	Horspool	DNA-Horspool	KMP	DNA-KMP
1M,100	62 192	348	94 960	344	334 436	4 167
50M,8	4 519 865	444 911	5 222 136	444 907	12 712 225	471 041
50M,200	3 101 612	12 614	6 864 503	12 609	16 645 353	201 636
50M,3 000	1 432 452	146 653	4 306 952	147 112	17 002 561	333 794

表 2 运行时间对比 ms

文本,模式	BM	DNA-BM	Horspool	DNA-Horspool	KMP	DNA-KMP
1M,100	0.016	0.001	0.062	0.047	0.015	0.016
50M,8	0.625	2.047	0.344	1.25	1.047	0.703
50M,200	0.468	0.719	0.453	1.688	3.859	0.656
50M,3 000	0.156	0.031	0.219	0.031	1.078	0.046

由表 1、2 的对比数据可知:

a) 当文本串和模式串都比较小时,压缩后直接匹配与普通的匹配相比,没有明显的优势,但当文本串和模式串都较大时,前者优势明显,其比较次数和比较时间都小很多。

b) DNA-BM、DNA-Horspool、DNA-KMP 三种算法性能差不多;DNA-BM 稍好一些。

DNA 序列数据通常都比较庞大,因此压缩后直接进行模式匹配是一种更好的策略。

四种算法及其运行时间的比较如表 3 所示。其中文本串 50 M。文本串、模式串均随机生成。为了方便比较,将这四种算法集成为一个小系统。这里的运行时间包括输入、输出、压缩、匹配等全部时间(单位为 s)。

表 3 算法比较 s

模式长度	DNA-match	DNA-BM	DNA-Horspool	DNA-KMP
16	3.185	3.435	3.255	4.045
800	2.834	2.894	2.884	4.066
6 400	2.824	2.884	2.884	4.156
12 800	2.824	2.914	2.915	4.045
102 400	2.874	3.044	3.025	4.136

从表 3 可以看出,DNA-match 算法的性能很好,比其他几种算法都稍优。

4 结束语

本文分析了 DNA 序列数据压缩及直接模式匹配的重要性和必要性,并根据 DNA 序列数据的特点,采用非自适应编码方式进行压缩。在此基础上,研究了两类直接模式匹配思路,设计实现了四种直接模式匹配算法,并集成为一个系统,进行了性能比较。在有大量 DNA 序列数据存在的今天,研究针对 DNA 序列数据的压缩和直接模式匹配算法有十分重要的意义和应用前景。笔者将继续致力于此方面的研究,争取取得更大的突破。

参考文献:

[1] MIR A, BENSON G. Efficient two-dimensional compressed matching[C]//Proc of the 2nd IEEE Data Compression Conference. 1992:279-288.

[2] AMIR A, BENSON G, FARACH M. Let sleeping files lie: pattern matching in Z-compressed files[J]. Journal of Computer and System Sciences, 1996, 52(2):299-307.

[3] CHEN X, KWONG S, LI M. A compression algorithm for DNA sequences and its applications in genome comparison[C]//Proc of the 10th Workshop on Genome Informatics (GIW '99). Tokyo: [s. n.], 1999:51-61.

[4] KNUTH D E, MORRIS J H, PRATT J V B. Fast pattern matching in strings[J]. SIAM J Computing, 1977, 6(2):323-350.

[5] BOYER R S, MORRE J S. A fast string searching algorithm[J]. COMMACM, 1977, 20(10):762-772.

[6] LEVITIN A. Introduction to the design and analysis of algorithms [M]. Pearson: Addison-Wesley, 2004:22-45.

[7] 林毅申, 林丕源. 基于 Web services 的生物信息解决方案[J]. 计算机应用研究, 2005, 22(6):157-158, 164.

[8] 邢仲璟, 林丕源, 林毅申. 基于 Bioperl 的生物二次数据库建立及应用[J]. 计算机系统应用, 2004(11):58-60.

下期要目

电子政务用户群细体分及关系模型研究
发展安全公钥密码系统的新方法研究
图像的多尺度几何分析概述
群体专家协作研究综述
高能物理网格数据管理关键技术研究
无线传感器网络操作系统关键技术研究
主题网络爬虫研究综述
基于 GQM 的软件体系结构适应性度量方法研究
模幂与点乘 m_ary 算法中窗口大小的最优化估计
基于预备工作集的最小序列优化算法
决策表属性集分解的等价性研究
基于 SOM 的离散数据挖掘集成框架研究
紧耦合分布式系统中的故障过滤算法
基于策略的对象存储文件系统管理模型研究
新型非线性信道判决反馈自适应均衡器
基于栅格地图的机器人覆盖路径规划研究
一种引入通信的多移动机器人编队方法
自适应实时资源管理技术研究