硕士学位论文

基于自索引结构的高通量基因组重测序数据压缩算法

HIGH-THROUGHPUT GENOME RESEQUENCING DATA COMPRESSION ALGORITHM BASED ON SELF-INDEX STRUCTURE

荣河江

哈尔滨工业大学

2018 年 6 月

国内图书分类号：TP39　　学校代码：10213
国际图书分类号：004.9　　密级：公开

工学硕士学位论文

基于自索引结构的高通量基因组重测序数据压缩算法

硕士研究生：荣河江
导　　师：王亚东教授
申请学位：工学硕士
学　　科：计算机科学与技术
所在单位：计算机科学与技术学院
答辩日期：2018 年 6 月
授予学位单位：哈尔滨工业大学

Classified Index: TP39
U.D.C:004.9

Dissertation for the Master Degree in Engineering

HIGH-THROUGHPUT GENOME RESEQUENCING DATA COMPRESSION ALGORITHM BASED ON SELF-INDEX STRUCTURE

Candidate: Rong Hejiang
Supervisor: Prof.Wang Yadong
Academic Degree Applied for: Master of Engineering
Speciality: Computer Science and Technology
Affiliation: School of Computer Science and Technology
Date of Defence: June, 2018
Degree-Conferring-Institution: Harbin Institute of Technology

摘 要

测序技术的进步，使得人们对基因组测序的兴趣日益增加。早期测序技术需要几年的时间来捕获 30 亿个核苷酸的基因组，目前新一代测序技术在数天内就可以对 220 亿个核苷酸的基因组进行测序。在测序速度提升的同时，测序成本也直线下降。基因组测序在个性化医疗和公共健康中日益发挥着重要的作用。越来越多的基因组测序数据在不断产生，这些数据需要进行有效的存储、传输和分析。如何解决高速增长的数据与有限的存储空间的矛盾，成为重要的研究课题。DNA 数据压缩为解决问题提供了一种有效思路。但由于 DNA 数据自身的特点，传统的压缩方法难以达到很好的压缩效果。

本文针对上述问题，在前两章调研了现有的高通量数据压缩技术，并对相关的压缩算法原理和以及面临的挑战进行分析，最后提出了改进的高通量数据压缩算法。本论文做了如下几件工作：

（1） 调研了高通量测序数据集的存储格式，以及现有的压缩算法。分析了测序数据的生物特性，同时通过分析表明，对质量分数的有损压缩，在提高压缩性能的同时，在下游分析中还能保持较好（有时甚至更优）的性能。

（2）在基于参考基因组进行差异化压缩编码的方案基础上，采用垂直方向的编码方式，同时对质量数采用稀疏化处理和均值处理相结合的方式，获得较好的有损压缩性能，实验表明压缩效果更优。

（3）针对数据需要随机解压缩和快速检索的需求，在分析自索引压缩技术原理的基础上，提出基于 PBWT 数据结构的自索引压缩技术，实验表明，自索引技术的引入，在随机解压缩上有较好的性能。

本文在基于参考基因组的压缩算法基础上，提出了基于自索引结构的随机解压缩算法，在压缩效率上有一定的优势，同时可以满足局部检索和解压缩的需求。这在一定程度上可以缓解海量高通量数据的存储和传输压力，为后续相关研究提供经验和借鉴。

关键词：DNA 数据压缩；基于参考基因组；自索引；垂直压缩

Abstract

There has been growing interest ingenome sequencing, driven by advancements in the sequencing technology. Althoughearly sequencing technologies required several years to capture a 3 billionnucleotide genome, genomes as large as 22 billion nucleotides are now beingsequenced within days using next-generation sequencing technologies. Assequencing speeds increase, the cost of sequencing has plummeted. As sequencingspeeds increase, the cost of sequencing has plummeted. Genome sequencing playsan important role in personalized medicine and public health. More and moregenomic sequencing data is constantly being generated, and these data need tobe stored, transmitted and analyzed. How to solve the contradiction betweenhigh-speed growth data and limited storage space has become an importantresearch topic. DNA data compression provides an effective way to solve problems.However, due to the characteristics of the DNA data itself, the traditionalcompression method is difficult to achieve a good compression effect.

In view of the above issues, theprevious two chapters investigated the current status of high-throughput data compression and analyzed theprinciples and challenges of the related compression algorithms. Finally, animproved high-throughputdata compression algorithm was proposed. The maincontribution of this study lie in:

(1) Researched the storage formatof high-throughput datasets and existing compression algorithms. The biologicalcharacteristics of the sequencing data were analyzed. At the same time, the analysisshowed that the lossy compression of mass fractions can maintain better(sometimes even better) performance in downstream analysis while improvingcompression performance.

(2) On the basis of the scheme ofdifferential compression coding based on reference genomes, a vertical codingmethod is adopted. At the same time, a combination of sparseness processing andmean processing is used for mass data to obtain better lossy compressionperformance. Indicates better compression.

(3) For the data needs of randomdecompression and fast retrieval requirements, based on the analysis of theprinciple of self-index compression technology, a self-indexing compressiontechnology based on PBWT data structure is proposed. Experiments show that theintroduction of self-indexing technology in the random decompression have better performance.

Based on the reference genome-basedcompression algorithm, this paper proposes a random decompression algorithmbased on self-index structure, which has certain advantages in compressionefficiency, and can meet the requirements of local retrieval and decompression.This can relieve the storage and transmission pressure of

massivehigh-throughput data to a certain extent, providing experience and lessons forsubsequent research.

Keywords: DNA sequence compression, Reference-based compression, self-index, Vertical Encoding

目 录

第 1 章 绪　论

1.1 课题研究背景

测序已经成为在生物研究中广泛应用的基本技术。获取不同生物体的基因遗传信息，能够帮助我们提高对周围有机世界的理解。DNA 双螺旋结构被解析以后，激发了人类对健康的探索以及对基因的研究。Sanger 测序方法的推广对生命科学研究产生了巨大的影响。高通量测序技术的发展，使经典分子生物科学家对基因组学的认识提升到一个新的水平，对临床和基因组学的研究产生了深远的影响。高通量测序技术(High-throughput sequencing，HTS)[1]又称为“下一代”测序（next-generation sequencing，NGS）技术，一次可以对几十万到几百万条核苷酸分子进行序列测定，是当前最为广泛应用的测序技术。与传统使用的 Sanger 测序法相比，高通量测序数据已更低的成本提供更快的操作，并且仅需要更加少量的 DNA 样本进行测序[2]。 测序数据在生物学、医学、遗传科学等诸多领域都具有重要研究价值。基因组测序的最终目的是分析数据，测序技术的高速发展改变探索生命蓝图的方式。推动了包括基因组学、生物信息学等学科的创立和发展，也为生物产业发展、基因诊断和基因治疗产业创造了巨大商机。

自 1977 年，Science 的两篇文章描述了第一种用于确定 DNA 片段中化学碱基顺序的方法[3]。从那时起，全基因组以及单个区域基因的测序已经成为现代生物学的一个焦点，并彻底改变了遗传学领域。四十年的时间，DNA 测序技术得到惊人速度的发展。伴随着更加便宜和更加便利的测序数据的普及，DNA 测序相关的应用在不断涌现，DNA 测序技术的应用需求也随之提升。研究人员对 DNA 序列数据有着极大的需求，遗传学家希望为生物的每个发育阶段，为健康和疾病个体提供 DNA 序列，希望能够得到全面的基因表达模式。甚至考古学家也希望通过 DNA 测序技术重建人类祖先的基因流动。分类学家、生态学、微生物学家等在寻求和分析生态系统的基因组。高通量测序技术，由于其高效的设计，特别是在所需的劳动力和试剂方面，由于几个供应商之间的有效竞争，使得自这些技术引入以来，测序成本稳步下降。图 1-1 显示了近些年基因测序成本的变化曲线图。

高通量测序技术的诞生是基因组学研究领域一个里程碑意义的事情，帮助基因组学理论得以快速实践和大规模应用转化，极大地推进疾病诊断、生物制药、生物化工等领域研究和应用进程。许多大型项目，例如千人基因组计划（1000

genome project）[4]、国际癌症基因组计划（international Cancer genome project）[5]、孟德尔遗传疾病计划（1000 Mendelian Disorders Project）[6]等项目，均采用高通量测序技术(High-throughput sequencing，HTS)，产生了海量 DNA 测序数据。NGS 技术的不断发展，使得人们甚至可以在一条 read 上读出整条基因组序列。根据 Verritas Genomics 的数据，人类基因组测序的成本已经下降到 1000 美元/人[2],预计未来几年测序成本将进一步下降。

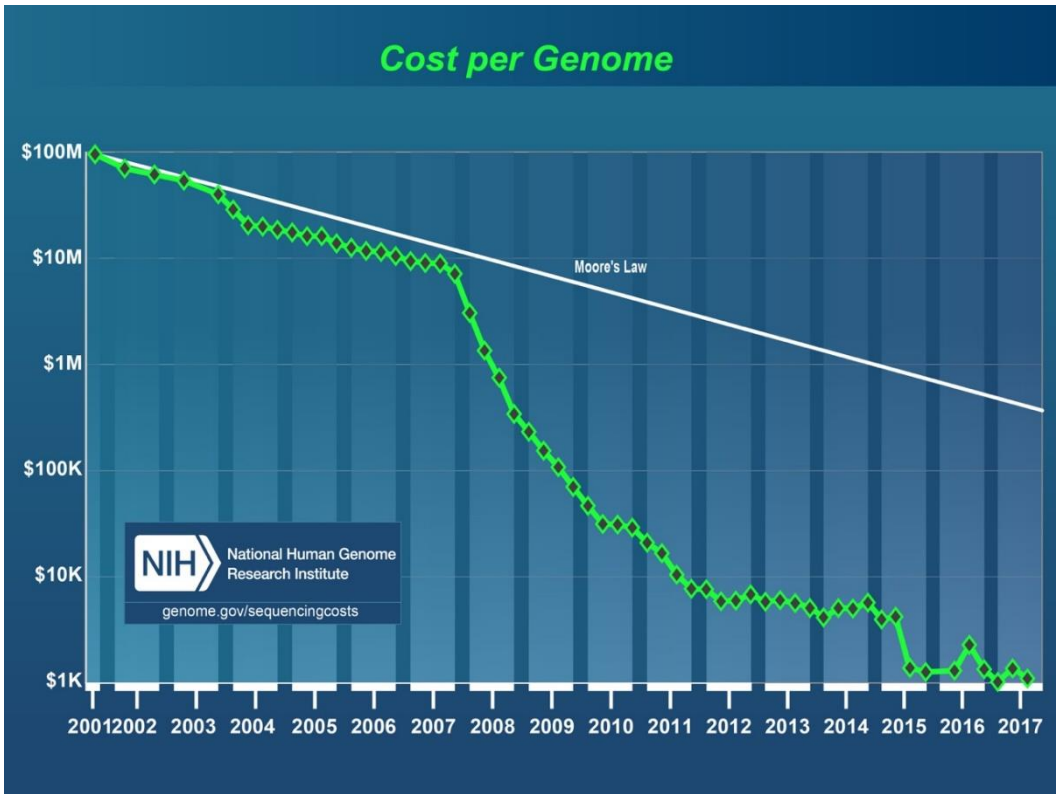


图 1-1 基因测序成本变化
（数据来源于 https://www.genome.gov/sequencingcosts/）

由于测序技术的发展和测序成本的降低，海量的测序数据呈现爆炸式增长。例如，千人基因组计划在前 6 个月中生成的测序数据超过了 NCBI GenBank 数据库 21 年期间累计的序列数据。按照目前的增长速度（测序数据大约每七个月翻一倍），每年将产生超过 1 亿兆字节的测序数据，到 2025 年接近泽（zettabytes）字节[7]。根据摩尔定律，计算机和计算和存储性能每隔 18-24 个月会提高一倍，显然 DNA 测序数据增长速度已经远远超过了硬件性能的增长速度。

1.2 研究的目的和意义

由于测序技术的进步，人类对基因测序的兴趣日益增加。早期测序技术需要几年的时间来捕获 30 亿个核苷酸的基因组。但目前正在使用新一代测序技术在数天内对大至 220 亿个核苷酸的基因组进行测序。此外，在过去的 15 年里，整个人类基因组测序的成本已从数十亿美元降至仅 1000 美元。面对不断增长的测序数据，可用的处理方案主要包括：（1）增加存储空间；（2）根据数据的重要程度，丢弃掉一部分重要程度较低的数据；（3）对数据进行压缩存储。

根据 Kryder 定律[8]，硬盘的存储密度大约每年翻一番。硬盘的成本也在呈现指数级别下降。但是我们生活在一个大数据时代，数据的产生远远超过硬盘存储技术的发展。例如，欧洲核子研究中心的大型强子对撞机在完全活跃是每年将产生大约 15PB 的数据[9]。在许多领域，如生态科学，神经生物学和临床医学方面，正在以前所未有的速度收集数据。采用增加存储空间的方式，项目的存储预算将随着数据量的增大逐步提高，在数据激增的情况下，这不是一个可持续的解决方案。因此单纯增加存储空间的方案难以适应当前的数据的飞速增长。

测序数据是独一无二的，因为样本不能用于重新测序，同时有机体和生态系统是不断变化的。通过物理方法，长期存储 DNA 样本本身也是一项非常复杂的操作，保存成本极高。很多样本都是难以或者不可再生的（例如癌症样本和其他人类样本），这些数据进行电子存档可能是永久可用的唯一可行解决方案。即使样本可用再生，对于临床样本来说，获得它们的成本也相当可观。而且用于处理和分析数据的工具会随着时间的推移而改善，因此未来可能会重新审视和重新分析数据。

在上述三种解决方案中，压缩是最具吸引力的。一方面，数据的采集是高度分散的，这需要很大的带宽来通过网络传输和访问这些大量的信息。这种情况下需要先进高效的大规模生物数据集的压缩方法，这不仅可以缓解存储需求，还可以促进这些数据的交换和传播。另一方面，压缩可用长期存储数据，无损压缩可以保证不会丢失实际信息，满足日后对该数据的使用需求。数据压缩这项工作至关重要，因为数据的存储和获取正在成为主要的瓶颈。测序数据有自身的特点和规律，分子信息之间有密切的联系，存在大量的信息冗余。这些数据特征也是进行数据压缩的基础。

压缩理论在香农[10]（Claude Shannon）创建信息论的背景下产生的。压缩是必要的，压缩过程一方面促进对数据信息结构的理解，还可以通过分析数据产生的过程，理解数据的内在机理。压缩技术的初始意图是捕获规律性和冗余性，

最终暴露出确认或发现信息的属性。特别是在基因系统中，生物体可以视为与环境沟通的主体，并存储适应环境所需的信息。无论内容还是形式，这些存储的信息都是在进化的过程中形成的，并通过 DNA 分子从一代传播到下一代，基因为了进化的目的有时会重复自己。生命的复杂性意味着即使简单的单细胞生物，DNA 分子中包含的信息量也非常大，需要有效的存储，在保持数据稳健性的同时，去除掉冗余信息。

基因组学的研究发展给如何对生物数据进行有效分析带来了新的挑战。面对高速增长的测序数据，如何以更低的成本存储数据，在满足相关研究的同时有效减少海量数据带来的存储和传输压力，成为促进 DNA 研究发展的重要因素之一。数据压缩为解决 DNA 序列传输和存储提供了一个新思路。DNA 数据压缩的研究已经成为人们关注的热点之一。如何在有限的存储资源下，有效急剧膨胀的 DNA 数据，是目前科学家面临的新问题。

1.3 国内外研究现状

由于 DNA 序列功能的复杂性，人类尚未完全掌握，这就需要在压缩的过程中保证数据的完整性可靠性。DNA 数据有其自身的特点，只包含四种碱基符号{A，C，G，T}。若将其看成 4 种碱基的随机字符串，则每个碱基需要 2bits 的存储空间。传统的压缩工具如 Gzip、bzip2、7-zip 等，一般用于压缩文本，图像，视频等格式的数据，这些通用压缩工具都未能充分考虑数据的某些生物特征，如重复内容，与现有序列的关系，小写字符的长度，回文字符串等，没有达到很好的压缩效果。

1993 年出现专门针对 DNA 数据的压缩算法。Crumbach[11]等提出针对 DNA 序列的互补回文结构的压缩算法 BioCompress，重点针对 DNA 序列中的直接重复和互补回文进行压缩，相比传统的通用压缩算法，有很大的优势，开启了 DNA 序列压缩的研究之门。1996 年，Rivals[12]等人提出 Cfact 压缩算法，该算法首次对没有压缩的重复片段进行编码，解析阶段使用后缀树结构来获得最大重复子串。1999 年，Chen Xin 等人提出 GenCompress[13] 压缩算法。该算法是基于字段算法的改进，定义两个序列之间的对称距离，来测量两个 DNA 序列之间的“相关性”，对近似匹配片段（Approximate matching fragment）进行编码。2000 年，Matsumoto[14]等提出 CTW+LZ 的数据压缩算法。这是另一种基于上下文树加权方法的算法，它结合了 CTW 和 LZ-77 类型的方法，对于长的精确、重复序列有 LZ77 算法编码，对于短重复序列由 CTW 编码。2005 年，Behzadi 等人提出 DNAPack[15]算法，该算法对 DNA 中不同数据片段采用不同的压缩方法。对于非重复区域采用二阶算数编码和上下文树加权算法进行混合

编码，对于数字部分，采用斐波那契编码和移位编码方法，对于近似重复部分采用汉明距离进行编码。2006 年，Srinivasa[16]等基于 DNACompress 算法，进行改进，提出 DNADP 算法，对于非重复片段，采用动态规划编码方式进行压缩。

上述针对 DNA 的压缩算法，均未考虑 DNA 数据构成的特点和相似性，未充分考虑 DNA 数据所具有的生物学特征和生物学含义。仅仅根据 DNA 序列中碱基的分布规律来进行压缩，压缩性能有所限制。2010 年，纪震等提出 BioLZMA[17] 压缩算法。该算法考虑到了生物学特性，并根据生物学特征和和生物学含义，将片段切分重组为四个集合：外显子、内含子、RNA 片段和剩余序列集合，根据不同集合内片段的特点，分别进行压缩。2011 年，朱泽轩等提出 POMA 算法[18]，该算法提出自适应粒子群优化模型算法（POMA），利用综合学习粒子群优化（CLPSO）和基于自使用智能单粒子优化器（AdplSPO）的局部搜索协同作用。2013 年，周家锐等提出 BioLMA-2[19]算法，该算法是基于生物学特征和 Memetic 优化的非对称压缩算法。算法通过合理调配全局优化过程与局部搜索策略，可在较少的计算资源内，获得更佳的寻优结果。Gaëtan Benoit[20]在 2015 年提出基于参考序列构建 de Bruijn[21]图，同时使用 de Bruijn 图执行质量分数的有损变换。并将结果存储在 Bloom 过滤器中。Ayad E. Korial[22]在 2017 年提出一种称为 A2 的 DNA 数据压缩算法。该算法由四个阶段构成替代模型。第一阶段使用改进的游程编码的方法，在第二和第三阶段映射模型用于格式化数据并使用 Burrows-Wheeler 进行变换，并结合改进的 Lempel-Ziv（LZ77）字典方法进行编码。

以上这些算法在数据规模较小时可以取得不错的压缩效果，但是在高通量测序技术中，DNA 测序是针对全基因组进行大规模测序，产生的数据量极其大，使用上述压缩算法的压缩效率有限。根据高通量测序方式的不同，可分为重测序(resequencing)和从头测序(de novo sequencing)两种不同的测序方法，产生了重测序 DNA 序列压缩和从头测序 DNA 数据压缩两种不同的压缩算法。

从头测序不依赖外部的参考基因组，一种方法是通过识别冗余的 DNA 序列，然后利用通用的压缩算法进行压缩处理，代表性的有 Beetl、SRComp、SCALCE 和 ORCOM。Beetl[23]算法利用 Burrows-Wheeler 算法识别冗余，SRComp[24]算法首先使用快速字符串排序算法 burstsort 按字典顺序对读取序列进行排序，使相同的字符串聚集在一起，然后使用基于 Elias omega 的整数编码对已排序的短读序列进行编码。 SCALCE[25]算法利用局部一致性对短读序列进行排序，ORCOM[26]算法利用并行的 Minimizers 算法压缩短读序列的重叠区域（overlap）。另一种基于短读拼接的方法，代表性的有 Quip[27]算法。Quip 算法

对未映射的短读序列采用马尔可夫链模型进行压缩。从头测序的方法不依赖于外部参考基因组，具有良好的完备性，但仍受限于拼接技术本身。

重测序是指利用高通量测序技术对已经取得完整基因组序列的物种的不同个体进行测序重。重测序技术已经测序完成的完整基因组作为参考样本，通过 NGS 技术产生 DNA 短读（reads）覆盖目标基因组，利用短读（reads）与参考基因组的映射获得目标基因组的数据。2009 年，Christley[28]等提出基于参考基因组的 DNAzip 算法。通过待压缩基因组与参考基因组进行映射，结合单核苷酸多态性 SNP（Single nucleotide polymorphism）数据以及多个核苷酸插入、缺失数据，将 James Waston 基因组压缩至 4Mbit。Brandon 等同年提出 BWB[29]算法，同样基于参考基因组，同时结合 Golomb，Elias，Huffman 等编码方式，对人类线粒体基因组进行压缩，压缩比达到 300:1。Carl Kingsford[30]等在 2015 年提出一种将基于参考基因组的方法优点与从头编码结合在一起的路径编码方式，利用 de Bruijn 图存储路径和依赖于上下文的算术编码之间建立联系，压缩效率有一定提升，同时在参考基因组与编码序列匹配度不高的情况下，仍能保持较好的压缩性能。基于参考基因组的压缩算法有较好的压缩比，但不具有自完备性，需要依赖参考基因组数据，如若发生参考基因组的缺失将直接影响压缩数据的使用。表 1-1 列出了重测序数据压缩的代表算法。

实际上，根据下游数据处理的要求，NGS 数据中会包含不同类型的信息，这些信息或多或少是有用的。每个 DNA 都附带有与测序技术和质量控制相关的附属信息。在某些情况下，可以通过丢弃部分或者降低这些信息来提高压缩比，这些有损处理，可以被接受的。Rodrigo[31]等在 2014 年提出对质量分数的有损压缩方法，量化了有损压缩对下游应用的影响，引入保真度标准，用以评估压缩性能和损失之间的权衡关系。Greg Malysa[32]等在 2015 年提出对质量分数的有损压缩算法 QVZ，该算法具有更好的速率失真性能，同时允许用户定义任何准凸最小化失真函数。

索引技术广泛应用与在信息检索，信息过滤。在信息量不断增大的情形下，利用索引技术可以进行高效的搜索。压缩可以节省数据的存储空间和传输时间，但是压缩后的数据，最重要的用途就是下游进行数据，在数据分析的过程中，就涉及到数据的查找。压缩文件本身很大，若根据需求只需要获得其中的某一条数据，采用完全解压，生成的原始文件不但占据了巨大的存储空间，而且浪费了大量的时间。如果能够让 DNA 序列在压缩状态下就能直接进行检索，或者在局部区域进行解压工作将节省大量的空间和时间。早期的索引通过引入新的数据结构，加速数据查找过程，索引本身也要占据大量的存储空间。在数据压缩中使用到的索引技术主要包括：Burrows-Wheeler 索引[33]，后缀数组索引，LZ

索引。

表 1-1 重测序数据压缩代表算法

算法	输入数据	编码	压缩参考基因组	技术特点
BWB	参考基因组 目标基因组	Colomb Elias Huffman	是	集合 Golomb、Elias 或 Huffman 编码方式对人类线粒体基因组进行压缩，压缩比超过 300：1
Cramtools	短读 参考基因组	Huffman Golomb	否	基于参考基因组，可使用有损 DNA 短读数据压缩
GRS	参考基因组 目标基因组	Huffman	否	在不依赖外部数据的情况下，将人类基因组压缩至 18.8Mbit
Quip	短读 参考基因组	Arithmetic delta	否	对未映射短读采用马尔可夫链模型进行处理
NGC	短读 参考基因组	Run-length Golomb	否	对 DNA 短读和之狼分数机械能压缩，支持有损和无损

Burrow-wheeler 索引，通过 BWT 变换的基础上，使用 FM-index 这种新的数据结构用于搜索和索引，提升了在压缩序列上的检索效率。FM-index 的数据结构包括 BWT 变换后的文本压缩表示和用于计算列排序位置查询的辅助结构，通过从 BWT 中获取一段文本的索引,然后使用向后查找算法找出所有的匹配。改进后的 FM-index 对空间的消耗也大大降低了，例如使用后缀数组，减少后缀树的空间损耗。Kref[34]等提出 LZ-End 索引，该索引使用 LZ77 算法解析输入字符串，不仅支持随机访问文本，同时还支持索引模式匹配，处理高度重复的序列数据效率较高。

1.4 论文结构

根据间根据 DNA 数据的特点，本文主要针对高通量 DNA 测序数据的压缩方法和理论及应用展开研究。论文的主要内容安排如下：

（1）压缩技术的理论基础，介绍 DNA 数据的特点及存储格式，分析现有的压缩技术，详细介绍了数据压缩的实现细节，通过数据对比，将现有的 DNA 压缩工具的优缺点进行分析，为后续高通量数据压缩算法的提出提供铺垫。

（2）采用基于参考基因组的差异化编码方式，提出了 RVZip 压缩算法。分析了垂直方向进行压缩的优势所在，并详细介绍了对于质量数的处理策略。最后通过实验，分析 RVZip 压缩算法的性能。

（3）分析索引技术在数据压缩中的应用，然后在此基础上，提出了一种基于自索引技术的压缩 SRVZip 压缩算法，并对应用 SRVZip 算法进行局部解压缩进行了详细说明。最后通过实验，实现自索引结构的局部随机解压缩算法。

（4）总结全文的主要内容及创新点，并对论文所研究领域的发展前景进行展望，提出了未来的研究方向。

第 2 章 DNA 测序数据压缩技术

2.1 引言

DNA（Deoxyribonucleic acid, 脱氧核苷酸）是存储生物遗传指令信息的双螺旋链状聚合物。DNA 是一种生物大分子，是遗传指令的组成部分，能够引导生物发育和生命机能的运作。DNA 的主要功能是信息存储，被喻为“蓝图”或“配方”[16]。DNA 序列中含有遗传信息的片段成为基因。作为生物生存、延续和发展的重要物质，DNA 广泛应用于生物学、医学等诸多重要领域的研究。DNA 信息可用于濒危生物物种的保护，基因序列信息可用于疾病的预测和治疗。对 DNA 序列进行测定和分析，成为重要的研究项目。在 DNA 序列数据的爆炸式增长，数据的存储成本日益突出的背景下，如何有效解决存储资源和急剧膨胀的 DNA 序列数据的不匹配的冲突，成为新的重要研究课题。通过采用有效的压缩编码方式，用较小的存储空间获得较大的 DNA 数据存储，是解决冲突的有效途径。由于 DNA 序列自身特点，传统的字符串、图像压缩算法，难以取得很好的压缩效果。针对 DNA 序列特点进行专门设计的压缩算法可以获得更好的压缩性能。因此在研究 DNA 测序数据压缩算法之前，先分析 DNA 测序数据的特点和存储格式以及压缩原理，有利于提高压缩效果。

2.2 DNA 数据特点

DNA（脱氧核糖核酸）是一种长链聚合物,由核苷酸重复排列组成。脱氧核糖核酸长链上的碱基以氢键相互吸引，维持双螺旋形态。这些碱基可分为两大类，嘌呤和嘧啶。脱氧核糖核酸的碱基分别是腺嘌呤（A），胞嘧啶(C)，鸟嘌呤和胸腺嘧啶(T)[35]。DNA 双链上的碱基通过氢键链接，形成碱基对。碱基对的组成符合一定规律，即嘌呤与嘧啶配对，并且腺嘌呤（A）只能与胸腺嘧啶（T）配对，鸟嘌呤（G）只能与胞嘧啶（C）配对。两股脱氧核糖核酸长链会以右旋方式交互缠绕成双螺旋结构，A 和 T 碱基结合在一起，同样 C 和 G 碱基结合在一起，这些配对被称为“补充”。传统上，这两股脱氧核苷酸长链是以相反的方向阅读的。例如：如果一条链包含序列“GATACCA”，则另一条链（向后读取时）将包含“TGGTATC”。图 2-1 显示了 DNA 的结构。

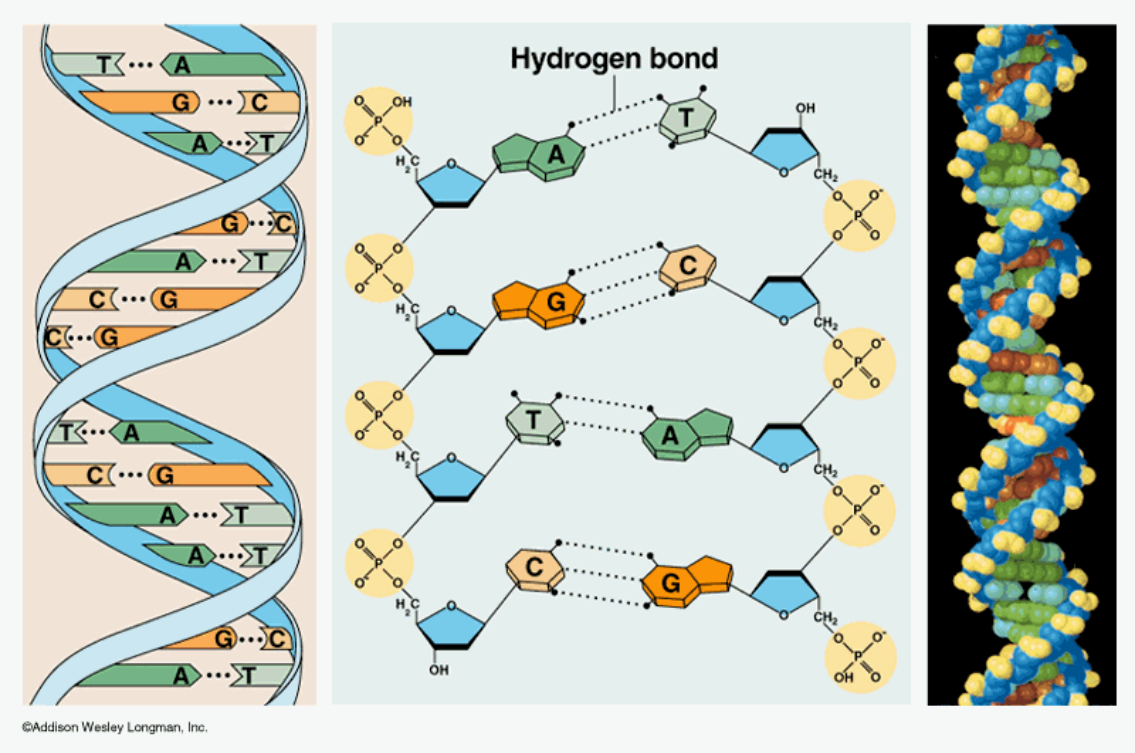


图 2-1 DNA 和双螺旋结构

两条链都可以通过仅检测一条链来唯一确定。所以 DNA 可以表示为由{A，C，G，T}字符组成的一维序列。生物 DNA 的某些部分包含基因，而这些基因包含被翻译成蛋白质的 DNA。一定数量的基因可以产生更多的数量的蛋白质。通过将螺旋结构分开成两部分来复制 DNA，分离成两条链。因为一条新的互补链可以由一条单链形成，每条单独的链可以形成一个与原来相同的新螺旋。但是，复制错误会导致随机突变，新螺旋与原始的螺旋稍有不同。当复制 DNA 时，新的延伸可能不一定与原始方向相同。这导致序列中的反向互补重复，这也是一般的文本压缩算法没有利用到的信息。DNA 和蛋白质序列表现出与其他类型数据不同的特性。虽然针对文本和图像文件的压缩算法利用小的重复模式和上下文相似性来实现压缩，但这些方法不能成功应用于 DNA 和蛋白质。

DNA 序列和其他数据之间的另一个区别是重复区域的性质不同。处理上述不能精确匹配之外，DNA 和蛋白质中的重复通常与文本中的重复相比要长得多。文本中某些单词可能会已非常高的频率出现，但是单词长度一般都较短，而基因序列中的重复模式通常要长得多，且不太频繁。

由以上特点可以看出，DNA 数据与普通的数据相比，有自身的组成特性。通用的压缩技术难以充分利用其特性进行有效压缩。与普通数据相比，DNA 数据具有如下一些特点：

2.2.1 构成特点

从组成形式上看，DNA 数据由{A，C，G，T}四个字符组成，可以将 DNA 序列看成是固定字符组成的超长字符串。若直接采用出传统的字符串压缩技术，则忽略了数据的内在构成特点，即腺嘌呤（A）与胸腺嘧啶（T）相匹配，鸟嘌呤（G）与胞嘧啶相匹配（C）的匹配原则。基于碱基互补原则，DNA 分子的双链结构中，仅包含了一条碱基序列的信息，这是进行序列压缩的基础。

2.2.2 功能特点

DNA 承载着生物遗传指令信息。序列的不同部分功能不同，其重要性也不同。例如，DNA 分子按功能分为基因（Gene）和基因组片段（Intergenic DNA）。其中基因是具有遗产效应的片段，存储遗传所需必要信息。而基因组片段是无遗传效应的片段，发挥着序列结构构造的功能。基因中的作为直接表达为蛋白自的外显子（Exon）部分，在序列碱基中的占比不到 2%。不表达为蛋白质的内含子（Intron）片段和 DNA 序列中的重复部分，占比达到 32%。若能根据 DNA 序列不同片段的特定和重要性，分别从生物学功能特性出发，进行针对性分析和压缩处理，将有利于提高压缩算法的性能。

2.2.3 信息特点

DNA 中碱基的排列并非完全随机的。是符合一定的生物学分布规律的。例如，在 DNA 编码区，三碱基周期性出现。富含基因的 DNA 片段中，G-C 碱基对出现的概率较高[36]，在基因匮乏去，则 A-T 碱基对的出现频率较高。由此可以看出，DNA 序列中的碱基排列规则是符合一定的概率分布规则的。

2.2.4 相似特点

DNA 序列高度相似性的特点，说明数据之间存在较高的冗余，这是进行压缩处理的基础。相似性主要体现在如下方面：

1. 不同物种间的 DNA 具有较高的序列相似性，生物分类中分类接近的物种间 DNA 相似度更高。同源物种的 DNA 数据相似度非常高。

2. 相同个体的 DNA 序列中存在着大量重复。DNA 序列中大量存在直接重复（Direct repeat）、反转重复（Inverted repeat）、镜像重复（Mirror repeat）和互补回文（Complementary palindromes）结构。图 2-2 显示了基因序列中的重复模式，许多重要的片段也会在基因序列中不断重复。

图 2-2 基因序列中的重复模式

2.2.5 变异特点

DNA 序列不仅具有高度相似性，同时在复制和表达的过程中序列也会发生变异（Mutaiton）和序列错误（Damage），造成碱基被随机插入、置换、位移或丢失。正是这些突变形成了丰富多样的物种。在生物个体中，DNA 序列也在不断地发生着突变。由于变异的存在，使得 DNA 序列的生物规律性有所降低，给序列压缩造成了一定的困难。图 2-3 显示了两条 DNA 序列的变异示意图。

GCCGGCTGAGTTTACTCCGCCTGTTCATGC
CGGTAAAGAGTTTACTCCCCCGGTTCAGGC

图 2-3 两条 DNA 序列的变异

2.3 高通量 DNA 数据的存储格式

高通量测序技术的不断发展，使得测序速度不断提高，测序成本快速下降，测序数据在不断产生，大量的数据需要处理和存储。这些数据既有直接通过测序获得的原始数据，也有通过软件分析后得到的结果文件。在不同数据处理阶段产生的数据，有不同的存储格式定义。如原始测序数据产生 FASTQ 格式数据，与参考序列对齐匹配后产生的 SAM/BAM 格式数据，数据拼接产生的 FASTA 格式数据等。

2.3.1 FASTQ 格式

FASTQ[37]格式是一种基于文本的，用于存储生物序列及其对应碱基质量的文件格式，序列与质量分数均用单个 ASCII 码来表示。该格式最初由桑格研究所开发出来，最初目的是将 FASTA 序列与质量数集成在一起，便于发布和共享，目前已经成功高通量测序结果的保存标准。

原始的测序数据主要有短读序列（reads）和质量分数（quality）组成。每条序列由 4 行字符表示。第一行是标题行：必须以“@”开头，后面跟着唯一的序列 ID 标识符及可选的序列描述内容，标识符与描述字符以空格分隔。第二行是核苷酸序列，表示碱基数据，由{A，C，G，T，N}构成，其中 N 表示不明确的碱基，即{A，C，G，T}中的任何一种。第三行：必须以”+“开头，”+“后面可以再次加上序列的标识及描述信息，或者没有信息，充当分隔符。第四行是：碱基的质量分数。每个字符对应第二行相应位置上碱基或氨基酸的质量（两行必须包含相同数量的符号），表示每个碱基字符在对应位置上测序的可信度。质量分数的得分在 FASTQ 文件中用 ASCII 字符表示。图 2-4 显示了一个 FASTQ 文件的例子。

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

图 2-4 FASTQ 格式

在 FASTQ 文件中，质量数是一个重要的值，它是一个碱基的错误率的对数值。在测序仪进行测序的时候，会根据荧光信号的强弱给出一个参考的测序错误概率（error probility），然后根据荧光强度的范围和分辨率得出碱基的 p 值。通过如下的 Phred 质量分数计算公式，得到每个碱基测序的准确性和可靠性。

$$Q = -log_{10}^{p} \tag{2-1}$$

对于每个碱基的质量编码表示，不同的软件有不同的编码方案。Sanger 采用的 Phred 范围值从 0 到 92，对应 ASCII 码的 33 到 126。对于测序数据，通常的质量分数得分小于 60。Phred 质量分数与错误率对应关系如表 2-1 所示：

存在于原始测序数据中的 reads 数量取决于覆盖范围（即基因组的特定核苷酸的预测测序次数）。例如，用 200 个覆盖率对人类基因组进行测序（illumina

技术）将产生大约 60 亿条 reads（假设的 reads 平均长度为 100）。因此生成的 FASTQ 文件非常大，通常大约为几百 GB 或者更大。

表 2-1 质量分数与错误率对应关系

SCORE	BASE CALL	ACCURACY
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10000	99.99%
50	1 in 100000	99.999%

2.3.2 SAM/BAM 格式

一旦生成了原始测序数据（FASTQ 格式存储），下一步通常就是对齐数据到参考序列上。简而言之，对齐过程是通过比较 reads 的序列和参考序列来为 FASTQ 文件中的每个 reads 确定其在参考基因序列中相应的位置（即使不存在这样的区域）。映射（Mapping）算法将试图找到参考序列中与 reads 匹配的（希望唯一）位置，同时容忍一定数量的不匹配情况。对于每条 reads，对齐过程都会提供 reads 映射到参考序列中的位置，以及不匹配的信息。该信息以标准 SAM 格式存储，其中还包括原始 reads 和质量数。这些文件会被下游应用程序使用，且非常大，通常在 TB 级或更大。

SAM[38]是由 Sanger 定制的，以 TAB 为分隔符的序列比对格式标准。主要应用于测序序列 Mapping 到基因组上的结果表示，也可以用于表示任意多重比对结果。SAM 分为注释信息（header section）和比对结果部分（alignment section）。图 2-5 显示了 SAM 格式的注释信息的示例。

```
@HD     VN:1.0   SO:coordinate
@SQ     SN:1     LN:249250621     AS:NCBI37
    UR:file:/data/local/ref/GATK/human_g1k_v37.fasta
    M5:1b22b98cdeb4a9304cb5d48026a85128
@RG     ID:UM0098:1PL:ILLUMINA     PU:HWUSI-EAS1707-615LHAAXX-L001
    LB:80     DT:2010-05-05T20:00:00-0400  SM:SD37743 C        N:UMCORE
@PG     ID:GATK TableRecalibration     VN:1.0.3471
```

图 2-5 SAM 格式注释信息

注释信息非必要，以”@”开头，用不同的 tag 表示不同的信息。其中”@HD”为符合标准的版本、比对序列的排序顺序说明，”@SQ”为参考序列的说明，”@RG”为比对上的序列的说明,”@PG”为使用的程序的说明，”@CO”为任意的说明。

比对结果部分（alignment section），每一行表示一个片段（segment）的比对信息。包括 11 个必须字段（mandatory fields）和一个可选字段，字段间以 tag 分割。11 个必须字段如下：

表 2-1 SAM 必须字段

字段名称	字段含义说明
QNAME	碱基或碱基对的名称
FLAG	位标识，匹配标记
RNAME	参考序列的名称
POS	开始匹配的位置
MAPQ	匹配质量
CIGAR	碱基匹配上的碱基（匹配类型 MIDNSHP）
RNEXT	下一个片段比对上的参考序列的名称
PNEXT	下一个片段比对上的位置
TLEN	模板的长度
SEQ	序列片段上的碱基序列
QUAL	序列的质量信息，格式同 FASTQ

图 2-6 示比对结果的示意图，其中第一条比对信息对应 QNAME 为“HWI-ST170:265:5:44:14178:183344#0”，FLAG 为“145”， RNAME 为“1”，POS 为“62421”，MAPQ 为“37”，CIGAR 为“63M1I35M”，以 tab 键分割，依次按序即可获得 SAM 必须字段内容。

HWI-ST170:265:5:44:14178:183344#0 145 1 62421 37 63M1I35M 18 56843949 0
CCTGTATACATAGTAATCAAAGTGTACCACTGGTCGGTGTTTGTGTTCAGGCCCCTGT
TGGGTAATGTGCATGTGAAGACCTCAGGTGGTATAGTTTTG
CEE?@F@BE@GGEGFBHHEDEEEDEEBEDHHBGHGGFHHDFHHHGGGGFFFEEEHFHF
GFHHHHHFHHHFHHHHGHGHEHHHHHHHHHFHHHHHHHHHH RG:Z:DU23M01_Duroc
XT:A:U NM:i:4 SM:i:37 AM:i:37 X0:i:1 X1:i:0 XM:i:3 XO:i:1 XG:i:1 MD:Z:20T22C1A52

图 2-6 SAM 比对结果信息

2.3.3 FASTA 格式

FASTA[39]格式是一种用于记录核苷酸序列的文本格式，其中的核酸（DNA/RNA）或蛋白质氨基酸（Amino Acid sequence）均以单个字母编码呈现。FASTA 简明的格式降低了序列操作和分析的难度。FASTA 格式中的一条完整序列，包含开头的单行描述行和多行序列数据。FASTA 包含标识行和碱基两部分数据，FASTA 格式首先以“>”开头，接着是一行序列描述信息。换行后是碱基信息。碱基部分仅包含{A,C,G,T,N}五种字符，不包含质量信息。图 2-7 显示了 FASTA 格式信息示例。

```
;LCBO - Prolactin precursor - Bovine
; a sample sequence in FASTA format
MDSKGSSQKGSRLLLLLVVSNLLLCQGVVSTPVCPNGPGNCQVSLRDLFDRAVMVSHYIHDLSS
EMFNEFDKRYAQGKGFITMALNSCHTSSLPTPEDKEQAQQTHHEVLMSLILGLLRSWNDPLYHL
VTEVRGMKGAPDAILSRAIEIEEENKRLLEGMEMIFGQVIPGAKETEPYPVWSGLPSLQTKDED
ARYSAFYNLLHCLRRDSSKIDTYLKLLNCRIIYNNNC*

>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken
ADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTID
FPEFLTMMARKMKDTDSEEEIREAFRVFDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA
DIDGDGQVNYEEFVQMMTAK*

>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

图 2-7 FASTA 格式信息

2.4 DNA 序列压缩算法

理解生物序列的含义具有广泛的应用，从药物合成到基因筛选和工程。序列的结构是理解它的重要基础。如果一个序列具有另一个序列所共享的特定属性，那么它们可能以某种方式相互关联，或者适用于一个序列的知识可能对另一个序列也有用。压缩可以帮助显示序列的结构以及它与其他序列的关系。压缩是进行基因组比较和研究基因组各种性质的重要工具。编码生命的 DNA 序列应该是可压缩的。众所周知，高等真核生物中的 DNA 序列含有许多串联重复序列，必需基因（如 rRNA）具有许多拷贝。还有证据表明，基因为了进化目的有

时会重复自己。所有这些事实都得出结论，DNA 序列应该是可压缩的。然而，压缩 DNA 序列并不是一件容易的事。

压缩过程由两个阶段组成：建模和编码。编码阶段用于产生比特序列，如哈夫曼编码（huffman）和算数编码。因此，建模在其中起到了关键作用，因为良好的模型意味着良好的压缩。一个良好的模型具有如下特征：1. 基于生物知识。2. 简单，参数少；3.每一个符号都具有信息量 4.有效的算法，实现良好压缩。

下面对 DNA 压缩方法的原理和相关代表算法进行介绍：

2.4.1 基于统计思想的 NDA 数据压缩算法

基于统计思想的压缩算法，首先对数据中的字符分布进行统计，计算出不同字符出现的概率，然后进行相关优化编码。由于 DNA 序列的并非随机分布的字符串，其中含有生物学信息，例如 DNA 序列上不同位置片段间的互补关系，基因变异或交叉等的敏感性程度。因此通过统计归纳碱基对的排列规律，归纳序列数据的具体生物学含义，估计碱基符号出现的概率模型，预测构成突变发生的位置，就能更加有效地进行碱基序列的压缩。

Allison[40]等学者在 1998 年提出 ARM 算法，这是最早的单纯统计类压缩算法。该算法对所有子序列的产生来源进行统计分析，由此子序列的概率和代表序列本身的概率。1999 年 Loewenstern 和 Yianilos 提出 CDNA 算法[41]，算法基于领域知识：DNA 序列包含更接近重复的序列，而不是随机的序列。算法将统计压缩与近似重复算法相结合，得到每个符号的概率分布，并将每个近似匹配应用于前面的子序列，该子序列与要编码的符号之前上下文的汉明距离很小。预测过程与一组权重组合起来，并且这些权重是自适应学习的。ARM 和 CDNA 算法的压缩效率明显高于替代类的压缩算法。Cao 等在 2007 年提出一个纯粹的统计 DNA 算法：XM 模型算法[42]，这是一个针对生物序列压缩的算法，它依靠专家提供的符号概率估计来进行算数编码。算法思想是，一个位置的概率分布基于之前的符号，为了形成符号的概率分布，算法维护一组专家，其符号的预测别组合成单个概率分布。专家是任何可以在某个位置提供概率分布的实体。XM 算法通过形成符号的概率分布并使用压缩方案对其进行编码来压缩每个字符。相应地，解码器可以看到所有先前的解码符号，能够计算相同概率分布并且可以恢复该位置处的符号。上述这些算法都是计算密集型的，这使得它们仅适用于压缩小序列。Armando[43]等提出基于有限上下文模型（Finite-Context Model,FCM）的压缩算法，在该模型中，解决了包括处理反向重复的机制，以及使用竞争编码数据的多个有限上下文模型，实验结果表明，这种方法以长压缩时间为代价，在压缩增益方面有着良好的结果。

目前使用较多的统计模型主要包括基于有限上下文的模型(Finite-Context Model,FCM)[44]和隐马尔科夫模型(Hidden Markov Model, HMM)[45]。

隐马尔科夫模型的主要作用是识别 DNA 数据中的编码区和非编码区，以便针对性地进行压缩。HMM 运行原理如下:

HMM 本身由初始概率分布π（向量）、状态转移概率分布A（矩阵）及观测概率分布B（矩阵）组成。形式化定义如下：

设 Q 是和 V 分别表示所有可能的状态集合和所有可能的观测集合：

$$q = \{q_1, q_2 \ \dots \ q_N\}, \qquad V = \{v_1, v_2 \dots . v_M\}$$

其中，N 是可能的状态数，M 是可能的观测数。

I 是长度为 T 的状态序列，O 为对应的观测序列

$$I = \{i_1, i_2 \ \dots \ i_T\}, \qquad O = \{o_1, o_2 \dots . o_T\}$$

A 表示状态转移概率矩阵：

$$A = \{A_{i,j}\}_{NxM} \tag{2-2}$$

其中，$a_{i,j}$是在时刻 t 处于状态q_i的条件小在时刻$t+1$转移到q_j的概率

$$a_{i,j} = p(i_{i+1} = \ q_j), i = 1, 2, \dots . N, ; j = 1, 2, \dots . M \tag{2-3}$$

B 表示观测概率矩阵：

$$B = [b_j(k)]_{NxN} \tag{2-4}$$

$b_j(k)$是在时刻 t 处于状态q_j的条件下生成观测V_k的概率。

$$b_j(k) = p(o_t = \ v_k \mid i_t = \ q_j), \quad k = 1, 2, \dots . M; j = 1, 2, \dots . N$$

π是初始状态的概率向量：

$$\pi = (\pi_i)$$

π_i是$t = 1$处于状态q_i的概率。

$$\pi_i = P(\ i_i = \ q_i), \quad i = 1, 2, \dots N$$

隐马尔科夫模型用三元组符号表示，即为

$$\lambda = （\mathbf{A}, \mathbf{B}, \pi） \tag{2-5}$$

转移状态矩阵A与初始状态概率向量π确定了隐藏的马尔可夫链，生成不可观测的状态序列。观测概率矩阵B确定了如何从状态生成观测，与状态序列综合确定了如何产生观测序列。因此，用来识别 DNA 中的编码区和非编码区是有效的。

2.4.2 基于字典思想的 NDA 数据压缩算法

基于字典的压缩算法与基于统计思想压缩算法的不同之处在于，不需要预先进行字符分布的统计，或者动态预测每个字符出现的概率。根据预先创建的字典，将待压缩数据与字典进行映射。静态字典模型要求预先准备好一个字典，

需要额外的信息维护字典，同时字典信息也需要在压缩文件中进行保存。因为静态字典的这些缺点，产生了基于字典思想的自适应压缩算法，可以动态地将未出现在字典中的数据作为字典项加入到字典中，LZ 系列算法即根据这一思路产生。

LZ77[46]使用前向缓冲区（待编码区的小段）和一个滑动窗口（搜索区）来实现。滑动窗口是个历史缓冲器，它被用来存放输入流的前 n 个字节的相关信息。前向缓冲区与动态窗口相对应，被用来存放输入流的前 N 个字节。

算法核心思想是利用该字典保存带压缩数据中最近扫描（编码）过的字符串，然后对字典后面的待编码的数据在字典中寻找最大子串的匹配，并输出三元组编码结构{偏移位移，匹配字符串长度，下一个字符}，输出三元组编码后，字符指针与待压缩数据指针均向后移动“匹配字符串长度+1”的位置，重复该过程。执行过程如图 2-8。

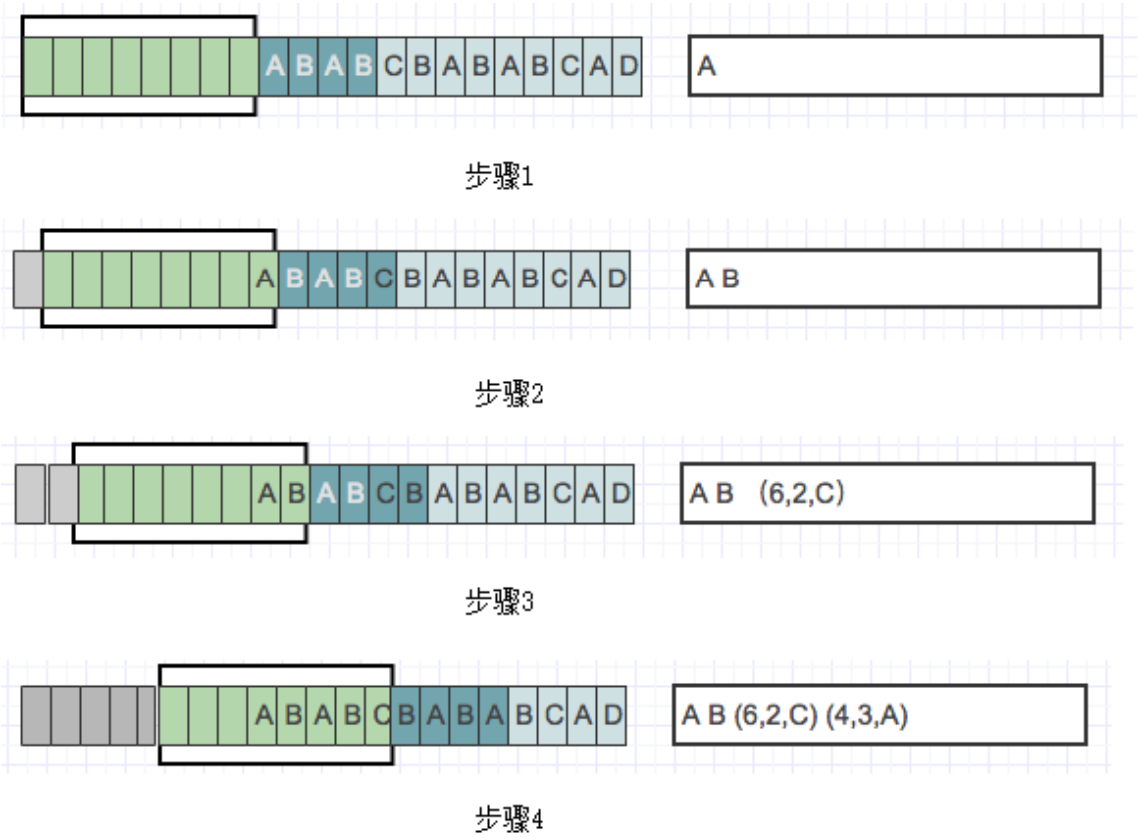


图 2-8 LZ77 算法过程

执行步骤如下：

（1）初始化时，因为滑初始化，因为滑动窗口没有匹配短语，所以记为标记。

（2）滑动窗口移到 A,缓冲区为 BABC，没有匹配到短语，依然把 B 标记为 B。

（3）滑动窗口在向右移 6 位时可以匹配到短语 AB，AB 长度为 2，短语后面的字母为 C，所以标记为（6,2,C）。

（4）滑动窗口向右移动 4 位时可以匹配到滑动窗口的短语 BAB，长度为 3，短语后面的字母为 A，所以标记为（4,3,A）。

（5）如此执行下去，直到缓冲区没有数据为止。

解压缩过程为压缩过程的逆过程。但是都需要维护好在扫描编码过程中建立的字典。LZ77 算法中利用滑动窗口中扫描匹配短语的代价最高。而该算法只使用一个小的窗口保存扫描过的数据，会存在有价值字典条目丢失的情况。

LZ78[47]算法采用动态树状词典维护历史字符串，以弥补 LZ77 算的不足。在压缩时维护一个动态词典，其中包括了历史字符串的位置与内容，压缩情况分为三种：

（1）若当前字符 c 未出现在词典中，则编码为（0，c）。

（2）若当前字符 c 出现在词典中，则与词典做最长匹配，然后编码为（prefixIndex，lastChar），其中 prefixIndex 为最长匹配的前缀字符串，lastChar 为最长匹配后的第一个字符串。

（3）对最后一个字符特殊处理，编码为（prefixIndex）。

LZ78 算法通过数输入缓存的数据预先进行扫描，并与其维护的字典中的数据进行匹配。通过扫描，输出数据在字典中的位置、匹配的长度以及无法匹配到的数据，并将结果添加到字典中。LZ78 算法字典的规模增长较慢，在处理大数据方面有优势。

LZW[48]算法是 LZ78 算法的衍生算法，由 Lemple-Ziv-Welch 三人共同创造，并以他们的名字命名。在压缩比和压缩效率上有提升。LZW 使用 4096 个条目的字典，每个序列被编码为 12 位，在压缩过程中，搜索与字典匹配的最长序列。当找到这样的匹配时，算法输出序列的索引，并在字典中插入一个新条目。字典最初在 256 个条目中填充 256 个可能的字节值，并允许解压缩程序从压缩数据中重建字典，而不必在压缩文件中存储任何有关它的内容。LZW 算法的核心思想是提取原始文本文件中的不同字符，将出现过的字符映射到字典上，然后采用替换思想，利用对应字典项在字典中的索引来替代原始文本文件中相应的字符，减少原始数据的大小。LZW 算法采用这种巧妙的方式，使

得字典不是事先创建好的，而是根据原始文件动态创建时，解码时从已经编码的数据中还原出字典。

2.4.3 基于参考基因组的 NDA 数据压缩算法

人类基因组计划绘制了人类的基因图谱，得到了 30 亿个碱基对的序列，投入 30 亿美元，使用第一代测序技术，将测序得到的序列组装（拼接/装配/Assembly），这个序列也称为最初的参考基因组。尽管人类基因组大约由 30 亿个碱基对组成，但是有文献表明任何两个人类基因组都有超过 99%的相似性[49]。基因组数据的相似性表明，可以利用这种冗余设计压缩方案。

对于已知基因组序列的物种，采用重测序方法，可以利用已知的参考基因组序列作为参考序列，在此基础上通过短读序列与参考基因组的映射，得到大量的单核苷酸多态性位点（SNP）、拷贝数变异（CNV），插入缺失和结构变异等信息，重测序数据压缩通过记录短读序列与参考基因组的差异信息来进行压缩。

基于参考基因组的 NDA 数据要流程如下：

1. 参考基因组的选择，为保证压缩效率，一般选择具有高度相似性的同源物种序列作为参考基因组。

2. 序列映射（mapping），确定 reads 与参考基因组的匹配位置，reads 长度、匹配类型、差异位置、差异类型、差异内容。

3. 采用高效编码方式对映射结果进行压缩编码。

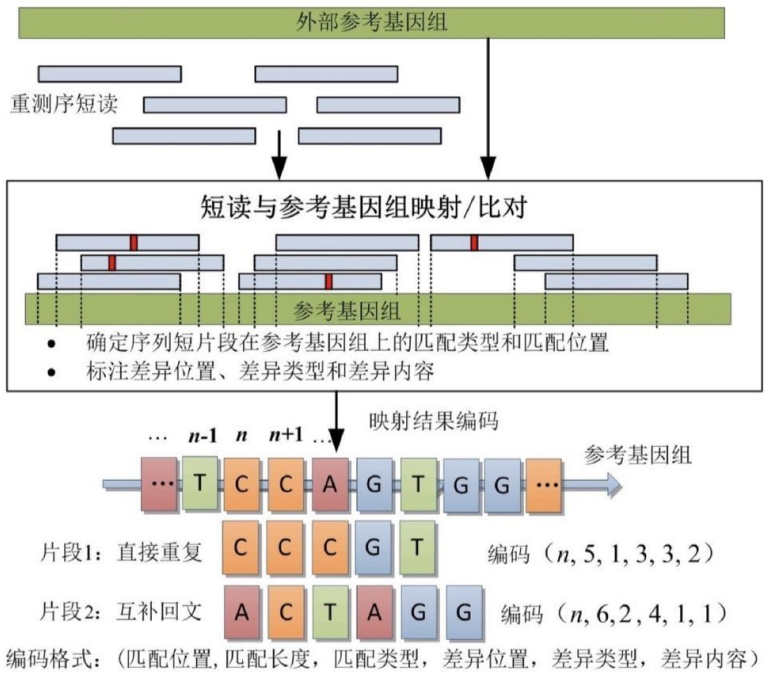


图 2-9 基于参考基因组的数据压缩

原理如图 2-9 所示，其中 reads 的匹配程度可归纳如表 2-2：

表 2-2 映射结果

匹配	描述
匹配位置	距离参考基因组起始位置距离
匹配类型	1.直接重复；2.互补回文
差异位置	相遇短读首字符的位置
差异类型	1. 插入；2.删除；3.替换
差异内容	A、C、G、T、N

重测序压缩过程的关键步骤是将 NGS 测序产生的 reads 与参考基因组进行序列比对，即短读映射，找出短读（read）与参考基因组之间的差异。例如 CRAM[50]采用基于 BAM 的输入，使用哈夫曼（Huffman）对短读序列和参看基因组之间的差异进行编码，CRAM 还使用基于 de Bruijn 图的装配方法来构建未映射到参考基因组的短读序列。NGC[51]通过遍历 SAM 格式的每一列对齐短读序列，利用映射到相同基因组位置的短读序列拥有共同特征的特点，对每个列的数据采用游程编码的方案存储映射的短读序列。HUGO[52]使用一种特定的压缩方案来存储 SAM 格式的短读序列，不精确的映射或未映射的短读序列将被拆分成较短的子序列，并针对不同的参考基因组重新对齐，直到它们全部被映射为止。

基于参考基因组的 DNA 压缩算法 可以获得较好的压缩比，但是受以下两个因素的制约：第一，对参考基因组的依赖性过强，某些物种不存在参考基因组，则此不能采用此方法；第二，参考基因组需要作为压缩数据的一部分，解压缩的过程中，仍需使用参考基因组，故需将参考基因组保存，丢失参考基因组，则数据无法进行解压还原，直接影响压缩数据的使用。

2.4.4 基于从头测序的 NDA 数据压缩算法

基因组从头测序[53]（De novo sequencing）是指对基因组序列未知或者没有近源物种基因组信息，直接对某个物种的基因组进行测定，利用生物信息学手段对测序序列进行拼接和组装，从而得到全基因序列图谱。

根据物种基因组的复杂度，特别是重复区域的大小和数量等信息，科学的制备各种长度插入片段的测序文库，合理的使用不同的高通量测序技术，能够高效经济的完成高等动植物的基因组图谱绘制。De novo 测序的主要策略：一是对

短片段 Shotgun 文库（300-1000 bp）进行深度测序，确保序列覆盖度和测序准确性，获得基因组基本序列信息；二是构建较长插入片段度的 mate pair 文库（3kb、8 kb、10 kb、20 kb 等等）并测序，确定短片段序列间的相对位置，通过拼接组装获得基因组序列框；三是通过 PCR 扩增技术获得序列间断开部分（gap）DNA 片段并进行一代测序，从而获得完整基因组序列。

从头测序的 DNA 数据压缩方法为：（1）短读（read）拼接，即将小段的短读拼接成一个长段，作为临时参考基因组；（2）采用基于参考基因组的压缩方式，依次进行短读映射。通过短读序列可以还原完整的 DNA 序列。目前 NGS 短读拼接算法包括 OLC（Over layout consensus）[54]算法和 de Bruijn 图算法[55]等。

相比基于重测序的 DNA 数据压缩方法，从头测序数据压缩不依赖与具体的外部参考序列，但是仍受限于短读拼接技术。从头测序的压缩方法较基于重测序的 NDA 压缩方法拥有更好的通用性，但是基于重测序的 DNA 压缩方法压缩效果更优。

2.4.5 DNA 序列压缩常用评价标准

对于 DNA 序列压缩算法的性能分析，主要从压缩比、压缩时间、鲁棒性、资源消耗几方面进行评价：

（1）压缩比

数据压缩比率可以定义为压缩前的数据大小与压缩后数据大小的比率。即

$$Compression\ \ Ratio = \frac{\text{Uncompressed Size}}{\text{Compressed Size}} \tag{2-6}$$

很多时候，也可以采用节约的空间来衡量压缩的效果。即

$$Space\ \ Savings = 1 - \frac{\text{Compressed Size}}{\text{Uncompressed Size}} \tag{2-7}$$

例如，一个文件从 10MB 压缩为 2MB，则压缩的比率可以认为是 80%。

压缩比（Compression Ratio）是评估压缩算法的最重要指标之一，用以表示压缩后信息量减少的程度。在衡量 DNA 序列压缩算法的压缩比是，通常使用如下的计算方法：表示单个碱基符号平均所需的比特数（Bit per base，BPB）。使用如下的压缩比计算公式：

$$Compression\ \ Ratio = 1 - \frac{|O|}{log_{k^*}^{k} \times |I|} \tag{2-8}$$

其中 I 为输入的序列的长度，O 为输出序列的长度。k 为输入序列中符号字母表（Symbol alphabet）中包含项目的个数，k*为输出序列符号字母表中包含项目的个数。在 DNA 序列压缩算法中，输入序列由 4 个输入符号（A，C，G，T）

构成，有 k=4，输出序列为 0，1 构成的二进制数据，故 k*=2，即压缩比公式为：

$$Compression\ \ Ratio = 1 - \frac{|O|}{2\times|I|} \tag{2-9}$$

从上述公式中可以看出，压缩比越大，压缩效果越好。

（2）压缩时间

压缩时间是评价压缩算法性能的另一项重要指标。压缩时间直接关系到算法是否具有实际应用价值。尤其是基因测序数据，数据量极大，压缩时间制约着算法能够被大规模推广应用。由于压缩时间受算法具体运行是硬件设备的性能影响，难以给出同一的评价标准，直接进行算法间的时间对比。在这里引入处理字节的平均周期（Cycle per byte，CPB）的概念来衡量压缩算法在特定设备上的运行速度。用以衡量平均每压缩一个符号所需的硬件运算周期，由此推算压缩时间，进行压缩效率的比较。解压时间也是一个重要的影响因素，解压时间是将压缩数据再次恢复为原始数据所用的时间，解压时间直接影响压缩后数据能够快速使用，可以参照压缩时间评价标准，进行评价。

（3）鲁棒性

鲁棒性表示系统在扰动和不确定情况下，仍能保持他们的特征的行为。在压缩算法中，鲁棒性可以用来描述同一压缩算法在不同的 DNA 序列片段上压缩的稳定程度。由于 DNA 序列的不同片段具有不同的生物学含义，直接影响序列上碱基的分布情况。一种压缩算法在一些基因片段上的表现良好，在其他一些片段上的压缩效果可能会不同。这就要求算法必须能够适应不同片段的数据特点，在不同的个体，不同片段位置上都能够保持稳定的压缩性能。鲁棒性在很大程度上影响着算法的实际使用价值。

（4）资源消耗

资源消耗既包括算法运行所需的硬件消耗，例如对计算机 CPU 时钟周期和内存容量等消耗。也包括实际运算时间等现实性消耗。一些数据如果需要一次性加载到内存中进行压缩处理，则内存容量将成为制约。一些算法在对模拟数据进行压缩时有很好的压缩效果，但是在实际压缩过程中资源消耗太大，只能对 DNA 的部分片段进行压缩，从而只能停滞在理论阶段，无法进行大规模实际应用。

2.4.6 不同压缩算法的性能比较

DNA 的数据的压缩方法对比如下表 2-3 所示：

表 2-3 DNA 压缩方法比较

类型	代表算法	优点	缺点
基于统计	GeNML、MFCompress	利用统计学思想归纳 DNA 中的重复片段，碱基的分布概率模型	1.统计得到的概率模型不具有通用性 2. 对高通量 DNA 数据压缩有限
基于字典	LZ77、LZ78、LZW	相比统计法，不需要预先进行统计	对高通量 DNA 数据压缩有限
基于参考基因组	CRAM、DNAzip	1.对高通量测序数据的压缩效果好，能获得很高的压缩比 2. 算法实现较简单	1. 对参考基因组的依赖性太强 2. 资源耗费大
基于从头测序	Gramtools、Quip	1.对高通量测序数据能获得较高的压缩比 2. 不依赖外部参考基因组，完备性好	受制于拼接技术

从上面的表格可以看出，对于高通量 DNA 测序序列，统计法和字典法的应用范围有限。而基于参考基因组和从头测序适应性更好。在实际应用中，权衡利弊，根据实际，选择合适的 DNA 数据压缩方法。

2.5 本章小结

本章首先详细介绍了 DNA 数据的特点，以及测序数据存储格式 FASTQ、SAM/BAM、FASTA 的特点，对 FASTA、SAM/BAM、FASTA 的格式压缩原理进行了说明。并对国内外经典的 DNA 序列压缩算法的原理进行分析。指明了各种压缩算法的优缺点和适用范围。

第 3 章 基于参考基因组的数据压缩算法 RVZip

3.1 引言

NGS 数据可以分为两大类：（1）原始 NGS 数据：通常以 FASTQ 形式存储并包含原始测序数据（2）对齐的原始的 NGS 数据：除了原始数据之外，还包含其余参考基因组的对齐。通常存储为 SAM 文件。

我们将重点放在对齐（基于参考基因组）的数据压缩上。因为这些数据都非常大（高达 TB），并且通常用于下游应用程序。同时，这些文件的大小是研究人员和机构之间传输和处理 NGS 数据的主要瓶颈所在，因此，良好的压缩算法是 NGS 数据处理最基本的需求。

在本章中，首先介绍了一下 RVzip 算法的整体框架，接着详细介绍了一下 RVZip 垂直压缩策略，针对质量分数的压缩是 NGS 数据压缩的重点，论文对 NGS 质量数提出稀疏化及均值化处理方案。最后进行了压缩试验，并对试验结果进行了分析。

3.2 RZZip 算法的设计

3.2.1 RVZip 算法整体框架

SAM/BAM 文件包含成千上万条短读（reads）序列，这些短读主要包括碱基、质量分数、和其他信息。这里重点处理的是碱基和质量分数。碱基最直观地代表了测序的内容，而质量分数则对应表示测序碱基正确性。因为采用基于参考基因组的压缩策略，SAM/BAM 文件内容，映射到参考基因组后，就会出现完全匹配和非完全匹配的情形。对于同源物种，相似度在 90%以上，将完全匹配信息和异常信息区分出来，既有利于有针对性地设计不同压缩方案，同时也利于下游的数据处理。RVZip 正是基于这种思想提出的 DNA 数据压缩算法。RVZip 的算法整体架构如图 3-1，核心是对待压缩数据进行分割，使用不同的压缩编码方法，有针对性地进行压缩，以获得最佳压缩比。

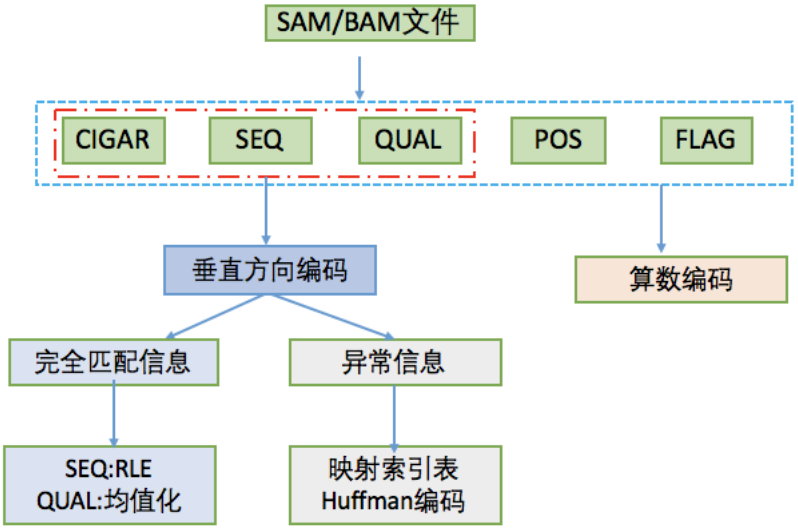


图 3-1 RVZip 算法框架

3.2.2 RVZip 算法垂直压缩策略

游程编码（Run-length Encoding， RLE）是常用的压缩方法，可用来对一些含有重复字符的字符序列进行编码的方案。一个游程长度（RL）表示由字母表中的一个字符和表示这个字符长度的整数组成的序列对。例如“AAA”用游程可表示为{3A}。游程编码（RLE）就是这样的游程（RL）的有序序列。例如：一个 DNA 序列 S=‘AACTTT’，则游程编码可表示为{A2，C1，T3}。

游程编码的思想是：一段连续相同的字符可以用游程（RL）表示，存储占位包括游程所代表的字符，和连续字符个数。很显然，这种简单的编码策略有其适用的范围，如果编码序列含有很长的相同字符，那么编码效果会很好。但是这对 DNA 序列无效的，因为大多数的游程（RL）会非常短，需要大量的这样的游程（RL）来进行编码。我们提出的压缩方案要基于减少游程（RL）的数量，然后再用一些通用的编码方案和压缩算法进行有效的编码。

垂直差异编码是基于参考基因组的压缩编码。与其他基于参考序列的压缩方法类似:我们不编码 reads 序列 S 本身，而是编码它与某些参考序列 Reference 的差异。在这里我们引入新字符'E'来扩展现有的{A，C，G，T}字母表。E 表示 read 中的值与参考基因组相比没有变化，即完全匹配。可以用这个特殊字符替换掉 read 中映射为参考序列的相等的字符。可以看出，E 越多，则说明待编码的序列与参考基因组的相似度越高。

对现有的游程编码（RLE）方法的改进是将这种碱基差异编码的读取方向，从“水平”方向，变成“垂直”方向。这意味编码的基本字节流不是按照短读序列（reads）一条条读取处理，而是按照参看基因组的位点位置，逐个处理。这样处理充分利用了生物学的特性，垂直（vertical）压缩编码利用生物学特性，在同一个位点上，reads 序列会有相似的表现，某些 reads 变异也集中在一些固定位点上，或者具有前后相关性。利用这一特点，采用垂直编码，利于将相似的结果集中在一起，进行压缩编码处理。就可以减少游程编码所需游程（RL）的数量。表 3-1 列出了垂直和水平不同方向进行游程编码的长度对比数据。

表 3-1 垂直方向和水平方向游程编码比较

	Reseq/chr20 (human)	Reseq/hm (human)	Reseq (A.thaliana)	Reseq (E.coli)	Reseq (E.coli,PE)	RNA-seq (E.coli)
Horiz DRLE counts	43,532,619	900,546	13,341,776	4,303,863	191,384,090	56,295,486
Vert DRLE counts	28,855,452	824,683	8,912,294	3,587,923	48,162,138	1,348,494
Ratio[%]	0.66	0.92	0.67	0.83	0.25	0.02

通过上述的数据分析可以看出，在垂直方向上采用游程编码相比水平方向，游程编码中游程的数量有大幅度减少，在 RNA-seq 的水平和垂直方向编码的结果对比可以看出，游程个数，垂直方向编码仅为水平方向编码游程的个数的 2%。这说明垂直方向上进行游程编码能够更好地利用测序的生物学特性，不同基因测序序列在同一位点上的相似度要远大于不同位点的相似度。也表现了在垂直方向上进行数据压缩的巨大潜力。

本算法将 BAM 文件的中的短读序列，按照序列起始位点（Start Alignment）的顺序，将每一条短读序列参照参考基因组（reference）坐标序列进行映射，形成完整的短读序列位点映射图。如图 3-2 所示：

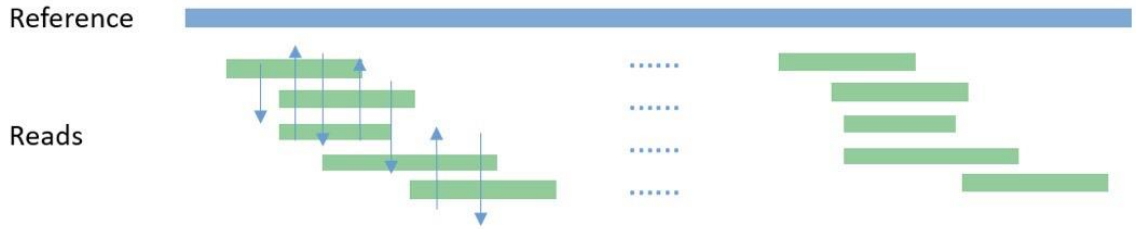


图 3-2 短读序列映射结果

在垂直方向上，压缩编码的流程如下，示意图如图 3-3:

（1）通过替换与相应参考基因组碱基匹配的信息，将完全匹配的值用“E”代替，得到查分信息。

（2）按照垂直的方向，使用游程编码（Run-length encoding， RLE）处理：每遇到一个不同于最后的编码的位点是，就会生成一个新的游程（RL）

（3）将字母表中的字符映射为字节值，同时用一个字节表示游程（RL）的长度。

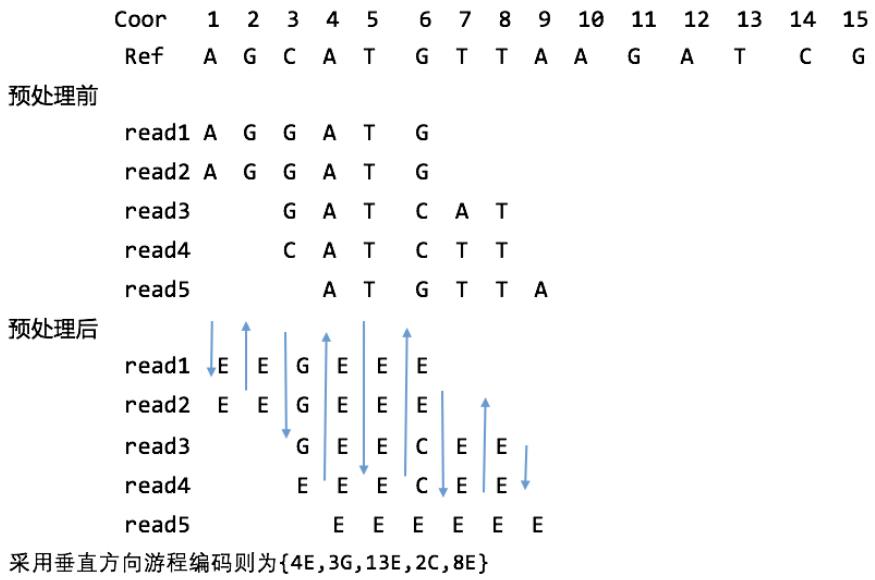


图 3-3 垂直编码流程

上述流程可以完成垂直方向的压缩编码，通过短读序列映射，形成短读序列基于参考基因组的重叠映射分布，与参考基因组的逐位点比对，识别出完全匹配和非完全匹配的数据，形成差异化编码的数据基础。以垂直方向进行游程编码，利用了上述所论证的增加数据有效重复部分，增长游程编码有效编码距离，提升游程编码的整体效率。同时，差异化编码的策略也有利于对数据进行有效分割，通过分割成完全匹配和非完全匹配的碱基，一方面可以针对性进行压缩，

同时非完全匹配的碱基位点单独处理，也有利于下游进行变异位点识别等功能实现。的在具体编码的过程中，可以结合情况，对存储字节流进行优化，可以使用通用压缩算法对存储的字节流再进行压缩。

3.2.3 RVZip 算法短读压缩策略

短读（reads）是关键的压缩信息因为它承载了大部分下游应用所使用的信息。因此它的压缩是十分重要的。对 reads 的压缩重点是对 reads 映射到参考基因组后的数据的处理。映射结果分为 M(匹配)、X(替换)、I(插入)、D(删除)四种情形。X(替换)和 D(删除)的情形，处理相对统一，不会对编码位点造成影响。对于 I(插入)的情形，基于参考序列，找不到对应的位点，则需要制定对应的处理策略，这里采用将 I(插入)依附于前一个位点的方式进行预处理。图 3-4 示意了短读序列的处理流程。

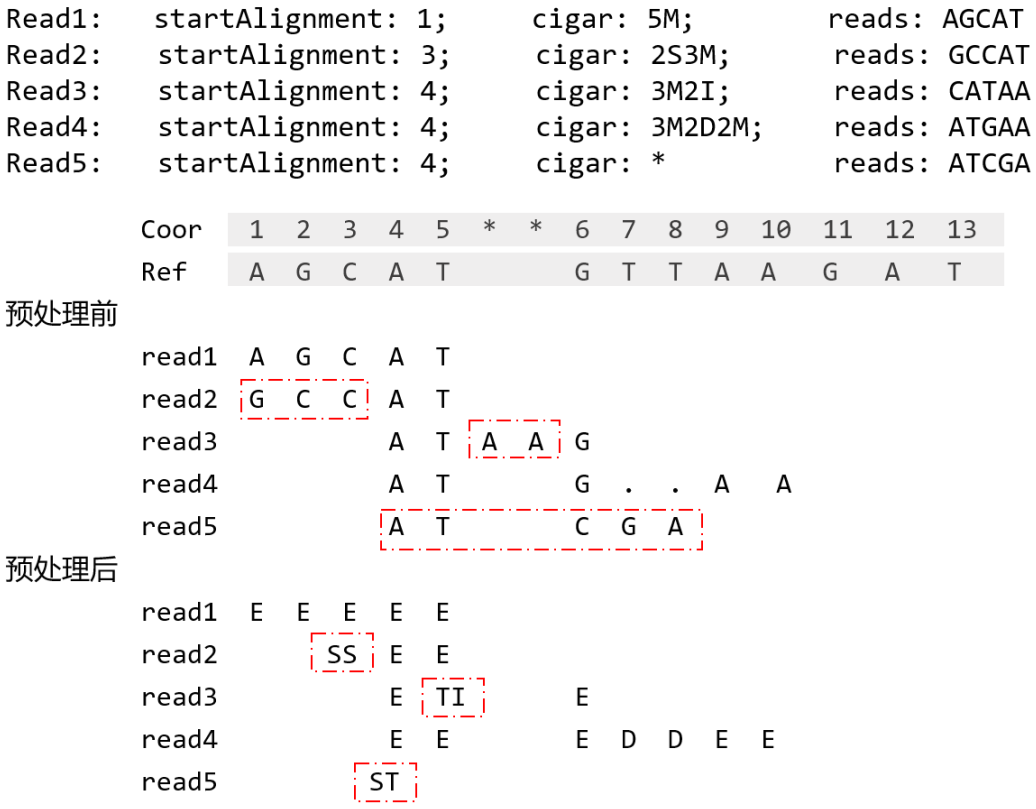


图 3-4 短读序列处理

利用 cigar 与 reads 结合，与 reference 形成对比，预处理可采用如下几种处理策略：

（1）当 cigar 为 * 的情形，则整个 reads 序列和 reference 匹配度偏低，直接将此 reads 起始位点的部分置为 ST，并将整条 reads 写入异常信息中。

（2）当 cigar 为 I 时（即 Insert 情形），将插入值的前一个位点特殊标记一下（置为 AI、CI、GI、TI 中某一个），并将具体插入的值写入异常信息部分。

（3）Delete 情形，用 D 代替删除的位置，并写入异常信息。

（4）S 的情形，在 reads 起始或结束会有类似 6s 的情况出现，在该位点标注为 SS，并直接拷贝对应长度的字符到异常信息中。

通过上述的短读序列处理策略，一方面，可以有效地将相对于参考基因组的匹配信息和非匹配信息进行区分，同时对非匹配信息的各种异常情形，按照生物学特性和生物学含义，针对性地进行处理，使处理后的结果既能完整展现其变异信息，保证压缩和解压缩的数据完备性，也使得变异信息和完全匹配信息的处理能够统一化，保证了算法处理具有较好的通用性和适应性。

在短读序列的处理中，整体采用前述的垂直压缩策略，并创建异常信息映射表，用以单独记录异常信息。对于短读，在垂直方向上进行游程编码，编码的过程中，异常和非异常信息均采用垂直游程编码，遇到异常信息，使用上述短读处理后的信息进行占位，同时将对应的异常信息写入异常信息映射表。当整个处理流程执行完成时，一方面在垂直方向上可以完整统一的完成数碱基的游程编码，同时异常信息映射表也完整记录了异常信息值。既保证了数据处理流程的高效性，同时保持了数据信息的完整性。

3.2.4 RVZip 算法质量分数压缩策略

根据统计研究，质量数的分布是相对随机的，这些数据的熵很高，导致很难压缩。还有证据表明质量评分在测序过程中引入了一定量的噪声而被破坏[40]。并且下游的应用程序，在使用 reads 序列的时候，对于质量数的处理，通常是使用启发方式的（heuristic manner）。

对于序列比对算法和变异体识别更是如此。基于这些观察，质量分数的有损压缩（相对于无损压缩）是保持下游应用良好性能同时，显著降低存储需求的必然选择。 对于质量分数的压缩方法，Slimgene[56]采用相邻质量分数之间差异的固定阶马尔科夫编码(fix-order Markov)，并使用哈夫曼编码来做压缩预测。Fastqz[57]使用固定长度编码，使用特定的字节模式表示质量分数超过 30 的部分，并将所有质量较低的分数值量化为 2。DEETL[58]首先应用 BWT 变换(Burrows-wheeler Transform, BWT)对 reads 序列进行变换，并在质量分数上面进行相同的 BWT 变换。然后扫描由 BWT 产生的核苷酸后缀，找到以相同的的 K 个碱基开头，同时还至少共享 K 个碱基前缀的后缀组。并将组内所有的质量数都转换为

平均质量值。PBlock[59]允许用户确定每个符号最大的失真阈值。文件中的第一个质量分数被选择成为第一个“代表”，如果产生的失真值落在阈值内，那表进行量化。如果超过阈值，则新的质量分数将取代代表，并继续该过程。该算法对“代表”和运行的长度进行跟踪，最后对代表和运行长度进行无损压缩。

Illumina 提出了一种新的分级方案，通过应用 8 级映射来降低质量分数的字母大小（见表 3-2）。这种分箱方案已经在最先进的压缩工具 CRAM 和 DSRC2[60]中实现。表 3-2 显示了 Illumina 提出的质量数 8 级划分规则。

我们在此，结合 Illumina 的分档方案，首先对质量数进行稀疏化处理，转变成 8bin 的处理结构。在同一位点下，相同的碱基表现统计分析发现其质量数也是趋于相似的，由此进行质量分数的均值处理。在较好保留质量分数精度

表 3-2 Illumina 提出的 8 层次划分

Quality Score Bins	New Quality Score
N(no call)	N(no call)
2-9	6
10-19	15
20-24	22
25-29	27
30-24	33
35-39	37
>=40	40

的前提下，降低编码的冗余。对于该位点上的异常信息，建立异常信息的映射关系表，对异常信息进行单独处理与存储。

质量分数可以用来衡量对应碱基的测序错误率。对于与参考基因组完全匹配的碱基，该位点下完全匹配的碱基的质量分数，相互间的差异度较小，是趋于相似的，对其进行质量分数均值化处理，对质量分数正确性的损失较小。而异常信息，由于异常位点是测序中关注的重点，且变异类型多样化，均值化处理处理会造成较大的损失。因此对异常的位点，质量分数单独记录处理，一方面保证压缩后质量数信息的完整性，同时也保证下游进行数据处理时，仍然可以有效利用质量分数信息。图 3-5 示意了质量分数的处理策略。

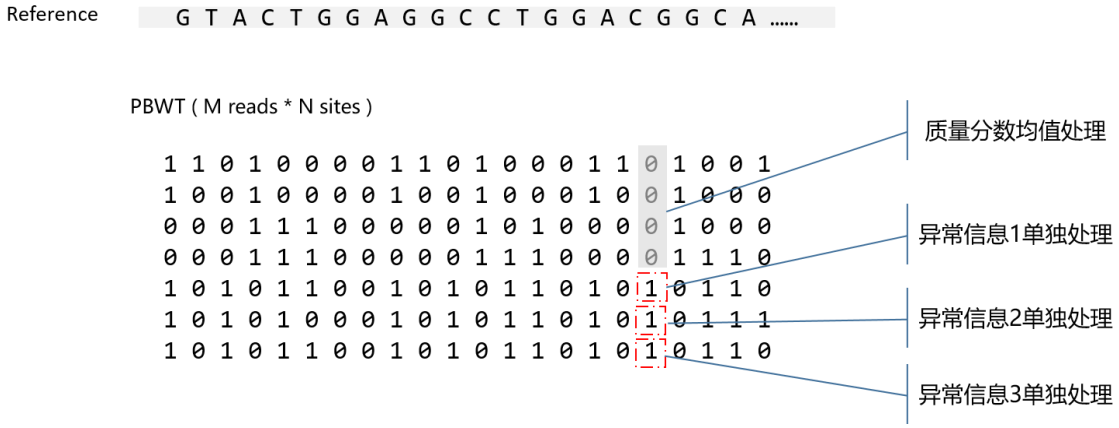


图 3-5 质量分数与异常信息质量分数处理

3.2.5 RVZip 其他数据流压缩策略

在实际的编码过程中，一些区间数值的出现概率较大，基于统计数据可以发现，Mate Pair 序列的相对距离符合高斯分布，通过对 read 统计均值(mean)和标准差(Standard Deviation)，确定高斯分布的区间，在区间内采用 Huffman 编码，对非此区间内的序列进行单独编码，以此保证整体获得较短的编码长度。对 Mate Pair 序列相对距离的统计结果如图 3-6。

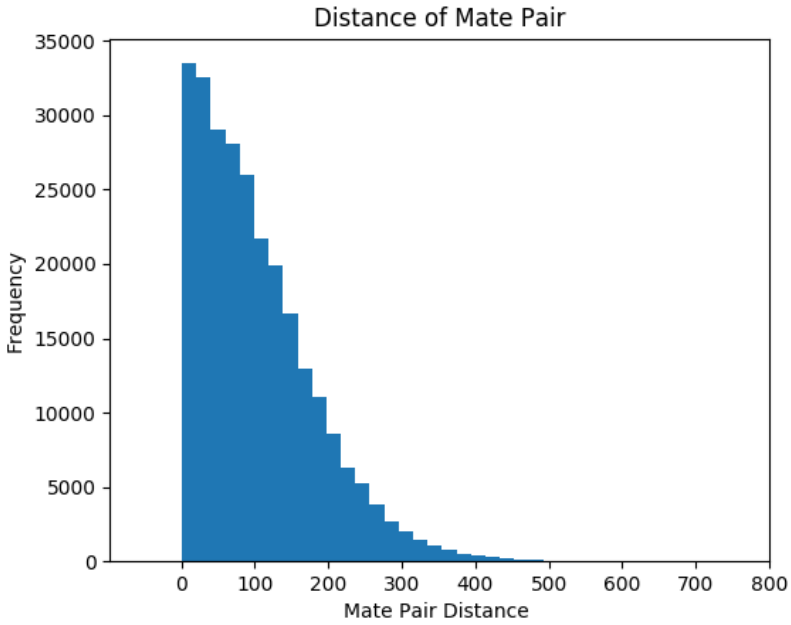


图 3-6 Mate Pair 序列相对距离统计

对于距离信息，例如短读序列的起始位点信息，采用相对位置替代绝对位置的方式，只需要选定合理的参照标准，记录与参照标准的差值，可以有效减少编码的长度。

对于短读序列的长度，单个 SAM/BAM 文件，采用既定的测序方式测序得到，短读序列的长度整体符合某一长度规律。设计一张短读序列长度映射表，只需要 6bit 就可以表示 64 种不同的长度，再设计一张对应的长度真实值对应表，在存储序列长度时，就可以减少数据存储量。

对于序列的一些注释信息，如短读序列的标识名称等，本身记录的信息一方面与测序工具等信息相关，另一方面起到不同序列间标识作用。这里可以根据情况，采用重新命名的方式，进行二次编码，既可以保留原信息中的有效部分，同时也可以剔除对下游数据分析没有直接影响的冗余信息，提高数据的压缩效率。另一方，在压缩过程中对注释信息的统一重新编码，也可以保证不同版本、不统一格式下注释信息的统一化处理，保持压缩数据更好的通用性。

3.3 RVZip 试验结果

我们以 NC_008253.fa 作为参考基因组，同时使用 Mason 模拟测序数据。通过 Mapping 工具生成 SAM 文件后，采用 Samtools 转化为 BAM 文件，再对 BAM 文件进行压缩。并与高通量测序数据压缩软件 CRAM 的压缩效率进行比较。

在这里，我们进行了如下三组测试，测试环境如下：i7 处理器，8G 内存，Ubuntu 16.04LTS。测试结果如表 3-3.

表 3-3 压缩结果比较

测序轮次	第一轮	第二轮	第三轮
短读（reads）条数	300000	1000000	2000000
SAM 文件(MB)	148	586	1420
BAM 文件(MB)	21	68	143
CRAM 压缩后(MB)	7.6	26	56
CRAM 压缩比	36.1%	38.2%	39.16%
RVZip 压缩后(MB)	12	30	48.3
RVZip 压缩比	57.1%	44.1%	33.8%

从上面的统计结果可以看出，随着测序深度的增加（即相同参考参考基因组的情况下，reads 条数增大），RVZip 显示出越来越高的压缩比，说明其在垂直方向上有了更好的压缩效果。

同时我们对上述各轮次压缩中具体的压缩的不同部分占比进行分析，得到如下的结果。测试结果如表 3-4 所示

表 3-4 不同部分压缩占比

测序轮次	第一轮	第二轮	第三轮
reads 正常匹配部分	11.6%	12.9%	13.6%
reads 异常匹配部分	36.8%	38.5%	41.3%
reads 正常匹配部分质量数	12.4%	11.8%	9.3%
reads 异常匹配部分质量数	39.2%	35.4%	35.8%

从上面的结果来看，上述结果质量分数占比仍然比较大，在相同变异位点比率的情况下，各部分的占比无明显变化。但是随着待压缩数据深度的增大，正常匹配部分的质量数占比在降低，说明均值化的处理策略在一定程度上有利于提升压缩比。还通过增加变异位点的数量，发现当变异位点数量明显增加时 reads 异常匹配部分的质量分数占比随之提高。

3.4 本章小结

在本章中，我们提出了基于参考基因组的高通量测序数据压缩的 RVZip 压缩算法。算法采用利用 DNA 数据的生物学含义，采用垂直压缩的压缩方向，有利于连续相同字符串产生，对提升游程编码的效率有重大意义。同时针对质量数，提出了质量数范围稀疏化处理以及新相同位点的均值化处理方案。实验结果表明，该方法取得了较好的压缩率，可以节省更多的存储空间。

第 4 章 基于自索引结构的数据压缩算法 SRVZip

4.1 引言

压缩数据可以减少存储空间和数据的传输时间。但是使用压缩后的数据前需要对数据进行解压缩。随着测序数据量的增大，解压过程需要许多计算资源，耗费很长时间，而且解压后的数据加载到内存中，这对内存的容量也提出了新的要求。如果能让 DNA 序列在压缩状态下就可以直接查询、检索，将有效地节约计算资源和硬件资源。同时，如果获得指定序列的信息，无需对整个压缩文件进行完整解压缩，将极大提升按需解压的效率。

使用索引技术可以在一定程度上可以实现对文本字符串的的高效搜索。传统的索引技术需要事先对原始文本建立辅助的数据结构，称之为索引，然后利用索引数据和原始文本信息实现快速搜索。传统索引技术的一个缺点就是，索引文件需要事先创建进行维护，而且索引文件通常都非常大，会造成巨大的空间浪费，空间率较低，例如全文索引的空间是原始文本大小的 4~20 倍。如果能够创建一种索引，不用维护巨大索引文件，能够在对庞大数据文本进行压缩的同时，生成自索引，利用自索引即可对原文本进行高效搜索，利用索引可以把压缩前的原始文本恢复出来，则将大大节省存储空间，提升处理效率。

在压缩状态实现序列的常规分析是未来发展的主要趋势，采用适当的索引结构可以实现压缩状态下的解压缩。目前基于压缩序列数据的索引结构包括：Burrows-Wheeler 索引、压缩后缀数组和 LZ 索引[61]。

自索引[62]是以压缩形式表示文本集合的数据结构，不仅支持随机访问文本，而且还支持索引模式匹配。 在过去的十年中发明的，它们极大地成功地大幅度减少了通用文本索引（如后缀树或数组）带来的空间负担。基于这种思想，我们提出基于自索引结构的压缩算法，利用自索引结构，仅需要付出极小的代价，存储少量的索引文件，就可以对压缩文件进行局部解压缩。

在本章中，首先介绍了一下算法的整体框架结构，接着重点介绍了一下 SRVZip 自索引的数据结构，以及实现随机解压缩的过程。最后对算法进行试验，并从压缩效率上对算法进行分析。

4.2 SRVZip 算法设计

4.2.1 SRVZip 算法总体思想

BWT（Burrows-Wheeler transfer）[33]变换，被广泛应用于压缩技术中，当对一个字符串进行 BWT 变换的时候，可以对字符进行一次有规律的重排序，变换的目的是方便后续进行查找。。BWT 算法通过重排序，可以是字符中相同的字符串变成一些连续重复的字符。对连续重复字符的处理，有利于提升压缩比。并且 BWT 操作数据变换的过程是可逆的，本身不会减少数据量，但是变换后的数据更容易压缩。FM-index 是构建在 Burrows-wheeler 的索引算法，是一种全文索引结构。它占用空间的大小依赖于建立索引的文本的可压缩性，易压缩的索引结构占用的空间小，不易压缩的索引结构占用的空间大。采用后缀数组数据结构，可促进压缩序列上搜索效率，同时 FM-index 对后缀数组也进行了压缩，使压缩后的空间占用降低，并且压缩以后的索引结构没有带来查询性能的明显下降。

基于此类思想，文章采用 PBWT（positional Burrows-Wheeler transform ）算法，PBWT 中数据不仅可以表示自身的信息量，同时前一列数据还可以表示后一列数据的位置索引，因此使用 PBWT 数据结构处理数据，一方面可以完整保存数据自身信息，同时也可以用于保存后一列数据的索引信息。采用类似 FM-index 的思路，引入 checkpoint，间隔一定区域存储索引信息，以此达到对指定区域进行解压缩。

4.2.2 SRVZip 自索引结构

BWT（Burrows-Wheeler Transform）转换也叫轮转排序，排序过程分为以下三步，过程见图 4-1：

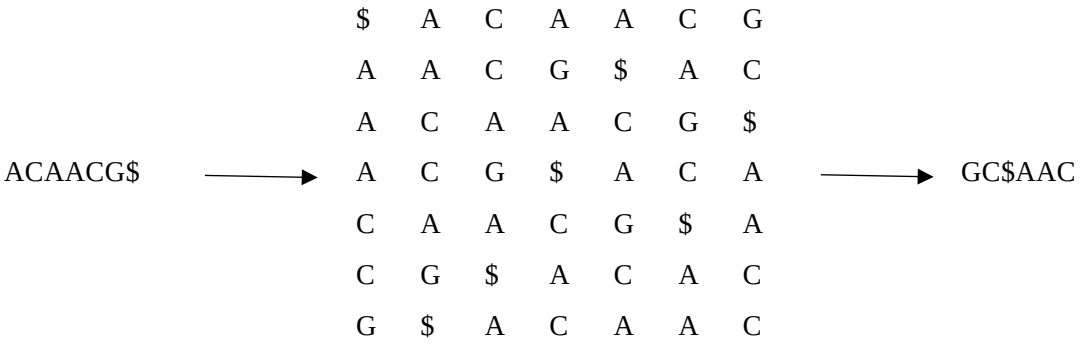


图 4-1 BWT 算法

（1）在字符串末尾增加 S，表示比任何字符都小；

（2）构造一个概念矩阵 M，首先对字符串的$进行循环位移，把每一行都看作一条新的字符，然后按首字母单词序进行排序；

（3）取出矩阵 M 的最后一列即为构成的字符串（即为压缩后的字符串）。

使用 BWT 算法进行编码的流程如下：

(1)首先，BWT 先对需要转换的文本块，进行循环右移，每次循环一位。对长度为 n 的文本块，进行 n 次循环重复，这样就得到 n 个长度为 n 的字符串。如下图中的“Rotate Right”列。（其中‘#’作为标识符，不在文本块的字符集中，这样保证 n 个循环移位后的字符串均布相同。并且定义'#'小于字符集中的任意字符）。

(2)对循环移位后的 n 个字符串按照字典序排序。如下图中的“Sorted (M)”列。

(3)记录下“Sorted (M)”列中每个字符串的最后一个字符，组成了“L”列。(其中"F"列是“Sorted (M)”列中每个字符串的前缀)

过程如表 4-1 所示：

表 4-1 BWT 算法进行编码

NO.	Rotete Right	Sorted（M）	F	L
1	banana#	#banana	#	a
2	#banana	a#banan	a	n
3	a#banan	ana#ban	a	n
4	na#bana	anana#b	a	b
5	ana#ban	banana#	b	#
6	nana#ba	na#baba	n	a
7	anana#b	nana#ba	n	a

例如：源字符串“banana#”就转换为了“annb#aa”。在某些情况下，使用 L 列进行压缩会有更好的效果。“L”列就是编码的结果。

因为 BWT 变换进行的是循环移位，有如下性质：

（1）L 的第一个元素是文本块中的最后一个元素。

（2）对于 Sorted（M）中的每一行，第一个元素都是最后一个元素的下一个元素，即对于文本块而言，同一行中 F 是 L 的下一个元素，L 是 F 的前一个元素。

使用 BWT 算法进行解码的流程如下：

(1)通过"F"列中的元素，找到他前面的字符，就是对应的同一行“L”列；

(2)通过“L”列中的元素，找到他在“F”列中的对应字符位置。但是“L”中有 3 个字符 a，如何对应 F 中的 3 个 a 呢？因为 L 是 F 的前一个元素，多个具有相同前缀的字符串排序，去掉共同前缀后相对次序没有变化。所有遇到多个相同的字符，相对位置不变。

(3)转到(1)，直到结束。

因为 F 列是已经排序的，可以从 L 列获得，所有只需要保存 L 列就可以。从 L 列中的字符获取在 F 列中的位置时，需要：

(1)前缀和数组，记录小于当前字符的字符数个数。

(2)count 计数，计算 L 中从开始位置到当前字符位置等于该字符的字符数。(保证多个相同字符下"L"到“F”的相对位置不变)。

BWT 算法进行解码的过程如表 4-2 所示。

表 4-2 BWT 算法进行解码

NO.	Sorted（M）	F	L	LF
1	#banana	#	a	# → a
2	a#banan	a	n	a → n
3	ana#ban	a	n	n → a
4	anana#b	a	b	a → n
5	banana#	b	#	n → a
6	na#baba	n	a	a → b
7	nana#ba	n	a	b → #

BWT 算法的处理是 PBWT 进行处理的基础，其中设计理念也是本算法的理念原型。

PBWT（positional Burrows-Wheeler transform）算法由 Richard Durbin[63]于 2014 年提出，该算法基于 BWT 算法，但是前一列数据是后一列数据的位置索引，对于具有前后关联性的数据，利用 PBWT 算法进行重排序，可以是相同的字符靠近在一起，有利于提升压缩比。算法示意图如图 4-2.

PWBT 可以按照字符的顺序进行重排序，如上示意图中，因为前一列数据是后一列数据的索引，将前一列数据为 0 所对应的当前数据提升到该列的上层，

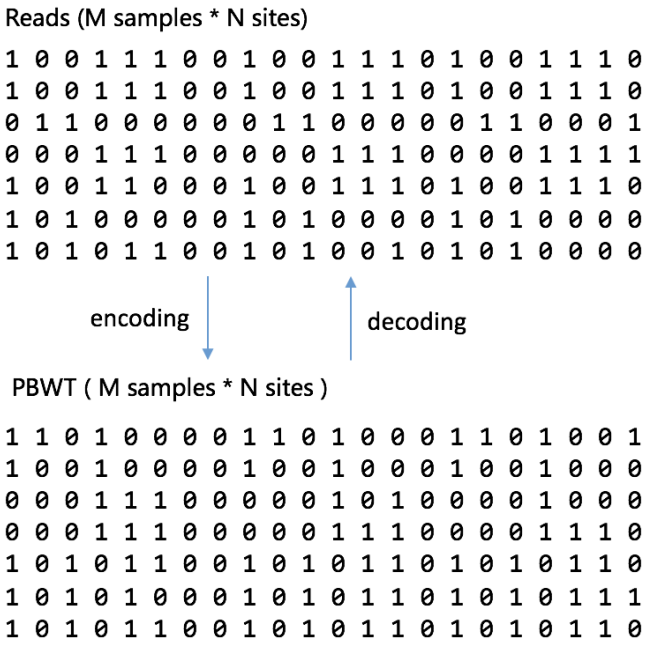


图 4-2 PBWT 算法

将前一列数据为 1 所对应的当前数据下降到该列的下层。对于基因序列，前后关联性较明显的情况下，可以是数据进行较好的划分。Richard Durbin 给出了实现的算法，借助额外两个数据记录索引位置即可。数据既能表示自身的信息值，同时也代表后一列的位置，无需额外存储额外的索引结构，即可完成重排序，并且该过程可逆。算法伪代码如下算法 1：

Algorithm1　PBWT BuildPrefixArray

算法 1：PBWT 构造前缀数组

输入：原始数组信息 y[k]

输出：构造完成的前缀数组 a[k]

1：　$u = 0, v = 0$, $create\ \ empty\ \ arrys\ \ a[], b[]$

2：　$\boldsymbol{for}\ \ i = 0 \quad to \quad M - 1 \quad \boldsymbol{do}$

3：　　$\boldsymbol{if}\ \ y_i^k[k] = 0 \quad \boldsymbol{then}$

4：　　$a[u] = \ \ a_k[i], u = u + 1$

5：　　$\boldsymbol{else}$

6：　　$b[v] = \ \ a_k[i], v = v + 1$

7：　$a_{k+1} = the\ \ concatenation\ \ of\ \ a\ \ followed\ \ by\ \ b$

8：　$\boldsymbol{return}\ \ a_{k+1}$

原始 PBWT 算法是针对排列整齐的数据进行处理，即存在许多长度相同且已经对齐的序列。本文采用基于参考基因组的压缩方法，序列映射到参考基因组以后，由于起始位置和长度的差异，并非整齐的数据。由此，在这里提出动态 PBWT 算法（Dynamic Burrows-Wheeler transform，DPBWT ）。我们通过两个一维数组记录短读（reads）序列的起始、终止位置，在进行 PBWT 编码处理时，短读序列动态加入、退出编码序列。算法实现伪代码如算法 2。

Algorithm2　Dynamic PBWT BuildPrefixArray

算法 2：动态 PBWT 构造前缀数组

输入：原始数组信息$y[M]$，序列起始、终止位置信息$start[M]$ $end[M]$

输出：构造完成的前缀数组$a[M]$

1：$create\ arrys\ start[M], end[M]\ \ to\ record\ N\ reads's\ start\ alignme$
$u = 0, v = 0\ , create\ empty\ arrys\ a[], b[]$

2：$\boldsymbol{for}\ i = 0\ \ to\ \ M - 1\ \ \boldsymbol{do}$

3：　　$\boldsymbol{if}\ y_i^k[k] = 0\ \ and\ start[i] \le k \le end[i]\ \boldsymbol{then}$

4：　　$a[u] = a_k[i], u = u + 1$

5：　　$\boldsymbol{else}$

6：　　$b[v] = a_k[i], v = v + 1$

7：$a_{k+1} = the\ concatenation\ of\ a\ followed\ by\ b$

7：**return** a_{k+1}

从上述算法中可看出，动态 PBWT 引入了$start[M], end[M]$ 两个一维数组，用于确定当前所处理位点对应的短读（reads）序列需要动态加入到处理队列中。

基于参考基因组进行差异化编码的时候（0 表示该位点碱基与参考基因组对应碱基相同，1 表示该位点碱基与参考基因组对应碱基不同），采用垂直编码方式，结合动态 PBWT 算法，可以使变异位点和非变异位点进行重排序，并且提现了前后相关性，这利用了基因遍历在一些位点的前后关联性的联动特性。

质量数的压缩可以采用未变异的连续位点均值化处理，变异位点单独记录的处理策略。既保证了较好的压缩效率，同时损失较小。

4.2.3 SRVZip 随机解压缩

在 BWT 算法中，采用 FM-index 索引方式，可以加速查找和解压缩，但是需要额外存储 FM-index 的索引信息。

LF-Mapping 进行还原，就是将最后一列的某个字符 x 映射到第一列的过程。如下图，先定义两个参数 C[c]和 Occ[c]。c 代表某字符，r 代表位于矩阵中的哪一行。C[c]的定义是，字典序小于字符 c 的所有字符个数。Occ[c,r]表示在 BWT（T）中第 r 行之前出现字符 c 的个数。

实现解压缩主要依赖 Last First(LF) Mapping，算法的实现伪代码如算法 3：

Algorithm 3 LF-Mapping(r)
算法 3: LF-Mapping

输入：BWT 压缩后的字符串数组 BWT[]

输出：还原的字符串 T

```
1：  r = 1
2：  T = ""
3：  while BWT[r] ≠ $ do
4：      T = prepend BWT[r] to T
5：          c = BWT[r]
6：          r = C[c] + Occ[c,r] + 1
7：  end while
7：  return T
```

通过上述算法可以发现，FM-index 在进行解压缩的时候，需要反复调用 Occ 函数，但是由于 Occ 的值是固定的，可以预算计算好，存储下来，以便计算时直接查找使用。但是根据 Occ 的定义，$Occ[X,i]$表示的意思是在最后一列字符串中，前 i 个字符里有几个字符 X。以人类基因组为例，3G 的长度，一个$Occ[X,A]$占用 4B，一行占用 16B，3G 行，则仅 Occ 就占用 48GB 的存储空间，无疑十分浪费。bowtie 在使用 FM-index 时，并未每一行都设置 4 个字符（A、C、G、T）对应的 Occ，而是间隔一段距离存一段 Occ（Bowtie 每个 448 行设置一个），此即为一个 checkpoint。

SRVzip 进行解压缩的算法伪代码如算法 4，即为压缩过程的逆过程。

Algorithm4　Dynamic PBWT Decoding

算法 4：　动态 PBWT 解压缩

输入：BWT 压缩后的字符串数组 BWT[]

输出：还原的字符串 T

1：　$get\ \ start[M], end[M]$

2：　$u = 0, v = 0, p = k + 1, q = k + 1$

3：　$get\ \ create\ \ empty\ \ arrays\ \ a[], b[], d[], e[]$

4：　$\boldsymbol{for}\ \ i = 0\ \ \ to\ \ \ M - 1\ \ \ \boldsymbol{do}$

5：　　$\boldsymbol{if}\ \ reads's$ start array and end array between $process\ \ section\ \ then$

6：　　$\boldsymbol{if}\ \ d_k[i] > p\ \ \ \boldsymbol{then}\ \ p = \ \ d_k[i]$

7：　　$\boldsymbol{if}\ \ d_k[i] > q\ \ \ \boldsymbol{then}\ \ q = \ \ d_k[i]$

8：　　$\boldsymbol{if}\ \ y_i[k] = 0\ \ \boldsymbol{then}$

9：　　$a[u] = \ \ a_k[i], d[u] = p, u = u + 1, p = 0$

10：　　$\boldsymbol{else}$

11：　　$b[v] = \ \ a_k[i], e[v] = q,\ \ v = v + 1, q = 0$

12：　$a_{k+1} = the\ \ concatenation\ \ of\ \ a\ \ followed\ \ by\ \ b$

13：　$d_{k+1} = the\ \ concatenation\ \ of\ \ d\ \ followed\ \ by\ \ e$

14：**return** $the\ \ concatenation\ \ of\ \ a_{k+1}\ \ followed\ \ by\ \ d_{k+1}$

在解压过程中，无需借助额外的索引信息，即可从头解压，直到结束。在此，我们类似 FM-index，每个一段区间，存储解压前此位点对应原始数据，作为 checkpoin。当对指定区间段进行解压缩是，只需要从该区间段所对应最近的前一个 checkpoint 开始解压缩，直到解压到所需位置为止。如此即可实现局部解压缩。

4.3 算法设计与结果

基于动态 PBWT 自索引结构，无所过多额外的索引结构信息进行存储，即能完成局部随机解压缩。我们采用每间隔 1000bp 设置一个 checkpoint 的分布条件下，单个 checkpoint 的存储空间与测序数据的深度有关。假设单个位点下对应碱基数量为 10 个，则仅需记录这 10 个位点相对于参考基因组的是完全匹配还是存在变异（用 0 表示完全匹配，用 1 表示存在变异）。然后将这一列对应的 0、1 匹配结果值进行存储即可。深度为 10 的情形下，平均 2Byte 的存储空

间即可存储下一个 checkpoint 的值，对参考基因组序列长度为 10000000bp 的情形下，共需要 10000 个 checkpoint，测序深度 10 倍的情形下，checkpoint 文件的总大小为 200KB。在整个压缩文件中占比较小。表 4-3 列出了在不同参考基因组及测序深度情形下，以参考基因组对应序列长度为 10000000bp 为例自索引结构 checkpoint 占据的存储空间大小。

表 4-3 动态 PBWT 解压缩算法

测序深度	checkpoint 间隔 500bp	Checkpoint 间隔 1000bp
10 倍	400K	200KB
20 倍	250KB	500KB
30 倍	300KB	600KB
50 倍	350KB	700KB

在局部随机解压缩方面，可以根据指定要求，快速解压指定起始、终止位点的短读序列。解压过程如下：

（1）用户指定解压某一区域的短读序列。

（2）找到小于等于该区域起始位置的 checkpoint 值。

（3）以该 checkpoint 值为起始，进行解压缩，直到解压到用户指定区域结束为止。

基于上述过程，我们在解压指定区域时，若 checkpoint 恰为指定区域起始位置，则没有任何的多余解压，解压到的数据即为所需数据。如果指定区域距离 checkpoint 有一定距离，则通过多解压 checkpoint 到指定区间起始位置的内内容，造成了一定的计算效率浪费。由于 checkpoint 的分布相对密集，这种额外解压部分占比非常小。以每个 500bp 设置一个 checkpoint，最坏情形下，则多解压 499bp 的压缩数据。

通过上述分析可以看出，基于动态 PBWT 算法的自索引解压缩结构，既保证了较小的索引信息存储，同时在进行精确局部解压缩时，也表现了较好的随机解压效果。

4.4 本章小结

在本章中，提出了基于 PBWT（positional Burrows-Wheeler transform ）算法的改进动态 PBWT 算法，该算法主要利用 PBWT 具有自索引功能的特点，数据既能代表具体的数值，同时又具有指示后一列索引位置的功能。动态 PBWT 可以保证对于长短不一短读序列的有效处理，增加的 PBWT 的适应性。PBWT 数

据结构的特点，对于前后具有关联性的数据处理具有更优的处理效果。针对局部解压缩的功能，受 FM-index 的设计思路启发，预先存储一些 checkpoint 信息，用以记录对应位点压缩前的位置索引信息。进行局部解压缩时，可以直接从 checkpoint 位置往后依次解压，直到需要的信息解压完成即可。实验结果证明，这种局部解压缩的方式，无须对整个压缩文件进行解压就可以获得指定区域的信息，大大提升了处理速度，也降低了对内存的要求。

结 论

高通量测序技术的发展，产生了急剧膨胀的数据，带来的巨大的存储和传输压力。DNA 数据压缩技术是缓解这一压力的有效途径。本文讨论了不同的测序数据以及对应的压缩方法，并给出了相应的理论分析。同时，针对 BAM 文件，提出了基于参考基因组的自索引压缩策略。本文主要工作如下：

（1）调研了高通量数据集的存储格式，以及现有的压缩算法。分析了测序数据的生物特性，同时通过分析表明，对质量分数的有损压缩，在提高压缩性能的同时，在下游分析中还能保持较好（有时甚至更优）的性能。

（2）在基于参考基因组进行差异化压缩编码的方案基础上，提出了 RVZip 压缩算法。该算法采用垂直方向的编码方式，同时对质量数采用稀疏化处理和均值处理相结合的方式，获得较好的有损压缩性能，并对压缩数据终端其他信息流采用统计分析的处理策略，有针对性地进行处理，实验表明压缩效果更优。

（3）针对数据需要随机解压缩和快速检索的需求，在分析自索引压缩技术原理的基础上，借鉴了 Bowite 中 FM-index 的设计思路，利用 PBWT 的自索引数据结构特性，提出 SRVZip 压缩算法，该算法是改进的动态 PBWT 算法，实现数据压缩与局部解压缩。实验表明，自索引技术的引入，在随机解压缩上有较好的性能。

本文提出的基于自索引结构的高通量基因重测序数据压缩方法虽然取得了一定的效果，但是在压缩比和压缩效率上，还有进一步提升的空间。为此进一步的研究工作如下：

（1）针对质量分数在压缩结果中占比过大的情况，本文采用异常位点无损压缩，完全匹配位点均值化处理的策略。虽然有较好的压缩效果，但有损压缩仍存在数据损失，后期可以考虑与 LZ78 编码及马尔科夫链技术相结合，探讨质量数的高效无损压缩方式。

（2）不同测序平台产生的不同质量和长度的高通量测序数据，在今后的工作中，考虑对数据的兼容性，探讨更加统一的处理标准，对不同平台的数据均可进行有效处理。

（3）研究更加具有针对性的测序数据压缩方式。可以从生物学角度出发，针对某些特定物种和具有某些生物学特征的基因片段进行压缩，以达到更好的压缩效果。

综上所述，本文提出的基于自索引结构的基因数据压缩算法在提高压缩率上面有所进展，但是仍有较大的提升空间，可视为高通量基因重测序数据压缩的有益尝试与探索，可为相关研究的进一步发展提供借鉴。

参考文献

[1] Margulies, Marcel, et al. "Genome sequencing in microfabricated high-density picolitre reactors." Nature437.7057 (2005): 376.

[2] Goodwin, Sara, John D. McPherson, and W. Richard McCombie. "Coming of age: ten years of next-generation sequencing technologies." Nature Reviews Genetics 17.6 (2016): 333.

[3] Green, Eric D., Edward M. Rubin, and Maynard V. Olson. "The future of DNA sequencing." Nature News 550.7675 (2017): 179.

[4] Siva, Nayanah. "1000 Genomes project." (2008): 256.

[5] International Cancer Genome Consortium. "International network of cancer genome projects." Nature 464.7291 (2010): 993.

[6] Xue, Yali, et al. "Deleterious-and disease-allele prevalence in healthy individuals: insights from current predictions, mutation databases, and population-scale resequencing." The American Journal of Human Genetics 91.6 (2012): 1022-1032.

[7] Mardis, Elaine R. "DNA sequencing technologies: 2006–2016." Nature protocols 12.2 (2017): 213.

[8] Montoya, Juan Sebastian, and Toshiro Kita. "Exponential Growth in Product Performance and Its Implications for Disruptive Innovation Theory." International Journal of Business and Information 13.1 (2018): 1-36.

[9] Collier, Paul. "The technical challenges of the Large Hadron Collider." Phil. Trans. R. Soc. A 373, no. 2032 (2015): 20140044.

[10] Shannon, Claude Elwood, and John McCarthy, eds. Automata Studies.(AM-34). Vol. 34. Princeton University Press, 2016.

[11] Grumbach S, Tahi F. Compression of DNA sequences[C]//Data Compression Conference, 1993. DCC'93. IEEE, 1993: 340-350.

[12] Rivals, Eric, et al. "A guaranteed compression scheme for repetitive DNA sequences." Data Compression Conference, 1996. DCC'96. Proceedings. IEEE, 1996.

[13] Chen X, Kwong S, Li M. A compression algorithm for DNA sequences and its applications in genome comparison[J]. Genome informatics, 1999, 10: 51-61.

[14]Matsumoto T, Sadakane K, Imai H. Biological sequence compression algorithms[J]. Genome informatics, 2000, 11: 43-52.

[15]Behzadi, Behshad, and Fabrice Le Fessant. "DNA compression challenge revisited: a dynamic programming approach." Annual Symposium on Combinatorial Pattern Matching. Springer, Berlin, Heidelberg, 2005.

[16]Srinivasa, K. G., et al. "Efficient compression of non-repetitive DNA sequences using dynamic programming." Advanced Computing and Communications, 2006. ADCOM 2006. International Conference on. IEEE, 2006.

[17]纪震, 周家锐, 朱泽轩. 基于生物信息学特征的 DNA 序列数据压缩算法[J]. 2011.

[18]Zhu, Zexuan, et al. "DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm." IEEE Transactions on Evolutionary Computation 15.5 (2011): 643-658.

[19]周家锐. 基于 Memetic 优化的高维代谢组特征数据智能加权算法研究. Diss. 浙江大学, 2014.

[20]Benoit, Gaëtan, et al. "Reference-free compression of high throughput sequencing data with a probabilistic de Bruijn graph." BMC bioinformatics 16.1 (2015): 288.

[21]De Bruijn, Nicolaas Govert. "A combinatorial problem." Koninklijke Nederlandse Akademie v. Wetenschappen 49.49 (1946): 758-764.

[22]Korial, Ayad E., and Ali Kamal Taqi. "Propose a Substitution Model for DNA Data Compression." DNA sequence 179.39 (2018).

[23]Janin, Lilian, Ole Schulz-Trieglaff, and Anthony J. Cox. "BEETL-fastq: a searchable compressed archive for DNA reads." Bioinformatics 30.19 (2014): 2796-2801.

[24]Selva, Jeremy John, and Xin Chen. "SRComp: Short read sequence compression using burstsort and elias omega coding." PloS one 8.12 (2013): e81414.

[25]Hach, Faraz, et al. "SCALCE: boosting sequence compression algorithms using locally consistent encoding." Bioinformatics 28.23 (2012): 3051-3057.

[26]Patro, Rob, and Carl Kingsford. "Data-dependent bucketing improves reference-free compression of sequencing reads." Bioinformatics 31.17 (2015): 2770-2777.

[27]Jones, Daniel C., et al. "Compression of next-generation sequencing reads aided by highly efficient de novo assembly." Nucleic acids research 40.22 (2012): e171-

e171.

[28]Christley, Scott, et al. "Human genomes as email attachments." Bioinformatics 25.2 (2008): 274-275.

[29]Brandon, Marty C., Douglas C. Wallace, and Pierre Baldi. "Data structures and compression algorithms for genomic sequence data." Bioinformatics 25.14 (2009): 1731-1738.

[30]Kingsford, Carl, and Rob Patro. "Reference-based compression of short-read sequences using path encoding." Bioinformatics 31.12 (2015): 1920-1928.

[31]Cánovas, Rodrigo, Alistair Moffat, and Andrew Turpin. "Lossy compression of quality scores in genomic data." Bioinformatics 30.15 (2014): 2130-2136.

[32]Malysa, Greg, et al. "QVZ: lossy compression of quality values." Bioinformatics 31.19 (2015): 3122-3129.

[33]Burrows, Michael, and David J. Wheeler. "A block-sorting lossless data compression algorithm." (1994).

[34]Kreft, Sebastian, and Gonzalo Navarro. "Self-indexing based on LZ77." Annual Symposium on Combinatorial Pattern Matching. Springer, Berlin, Heidelberg, 2011.

[35]Watson, James D., and Francis HC Crick. "Genetical implications of the structure of deoxyribonucleic acid." Nature 171.4361 (1953): 964-967.

[36]Kaneko, Takakazu, et al. "Complete genome structure of the nitrogen-fixing symbiotic bacterium Mesorhizobium loti." DNA research 7.6 (2000): 331-338.

[37]Cock, Peter JA, et al. "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants." Nucleic acids research 38.6 (2009): 1767-1771.

[38]Li, Heng, et al. "The sequence alignment/map format and SAMtools." Bioinformatics 25.16 (2009): 2078-2079.

[39]Stothard, Paul. "The sequence manipulation suite: JavaScript programs for analyzing and formatting protein and DNA sequences." (2000).

[40]Allison, Lloyd, Timothy Edgoose, and Trevor I. Dix. "Compression of strings with approximate repeats." ISMB. 1998.

[41]Loewenstern, David, and Peter N. Yianilos. "Significantly lower entropy estimates for natural DNA sequences." Journal of computational Biology 6.1 (1999): 125-142.

[42]Cao, Minh Duc, et al. "A simple statistical algorithm for biological sequence compression." Data Compression Conference, 2007. DCC'07. IEEE, 2007.

[43]Pinho, Armando J., et al. "Finite-context models for DNA coding." Signal Processing. InTech, 2010.

[44]Pinho, Armando J., Diogo Pratas, and Paulo JSG Ferreira. "Bacteria DNA sequence compression using a mixture of finite-context models." Statistical Signal Processing Workshop (SSP), 2011 IEEE. IEEE, 2011.

[45]Crouse, Matthew S., Robert D. Nowak, and Richard G. Baraniuk. "Wavelet-based statistical signal processing using hidden Markov models." IEEE Transactions on signal processing 46.4 (1998): 886-902.

[46]Ziv, Jacob, and Abraham Lempel. "A universal algorithm for sequential data compression." IEEE Transactions on information theory 23.3 (1977): 337-343.

[47]Ziv, Jacob, and Abraham Lempel. "Compression of individual sequences via variable-rate coding." IEEE transactions on Information Theory 24.5 (1978): 530-536.

[48]Welch, Terry A. "Technique for high-performance data compression." Computer 52 (1984).

[49]International Human Genome Sequencing Consortium. "Initial sequencing and analysis of the human genome." Nature 409.6822 (2001): 860.

[50]Fritz, Markus Hsi-Yang, et al. "Efficient storage of high throughput DNA sequencing data using reference-based compression." Genome research 21.5 (2011): 734-740.

[51]Popitsch, Niko, and Arndt von Haeseler. "NGC: lossless and lossy compression of aligned high-throughput sequencing data." Nucleic acids research 41.1 (2012): e27-e27.

[52]Li, Pinghao, et al. "HUGO: Hierarchical mUlti-reference Genome cOmpression for aligned reads." Journal of the American Medical Informatics Association 21.2 (2013): 363-373.

[53]Zhang, Jing, et al. "PEAKS DB: de novo sequencing assisted database search for sensitive and accurate peptide identification." Molecular & Cellular Proteomics 11.4 (2012): M111-010587.

[54]Miller, Jason R., Sergey Koren, and Granger Sutton. "Assembly algorithms for next-generation sequencing data." Genomics 95.6 (2010): 315-327.

[55]Schatz, Michael C., Arthur L. Delcher, and Steven L. Salzberg. "Assembly of large genomes using second-generation sequencing." Genome research 20.9 (2010): 1165-1173.

[56]Kozanitis, Christos, et al. "Compressing genomic sequence fragments using SlimGene." Journal of Computational Biology 18.3 (2011): 401-413.

[57]Bonfield, James K., and Matthew V. Mahoney. "Compression of FASTQ and SAM format sequencing data." PloS one 8.3 (2013): e59190.

[58]Janin, Lilian, Giovanna Rosone, and Anthony J. Cox. "Adaptive reference-free compression of sequence quality scores." Bioinformatics 30.1 (2013): 24-30.

[59]Cánovas, Rodrigo, Alistair Moffat, and Andrew Turpin. "Lossy compression of quality scores in genomic data." Bioinformatics 30.15 (2014): 2130-2136.

[60]Roguski, Łukasz, and Sebastian Deorowicz. "DSRC 2—Industry-oriented compression of FASTQ files." Bioinformatics 30.15 (2014): 2213-2215.

[61]Kreft, Sebastian, and Gonzalo Navarro. "Self-indexing based on LZ77." Annual Symposium on Combinatorial Pattern Matching. Springer, Berlin, Heidelberg, 2011.

[62]Green, Edward A., Luis Rivas, and Mark Kreider. "Self-indexing data structure." U.S. Patent No. 8,775,433. 8 Jul. 2014.

[63]Durbin, Richard. "Efficient haplotype matching and storage using the positional Burrows–Wheeler transform (PBWT)." Bioinformatics 30.9 (2014): 1266-1272.

攻读硕士学位期间发表的论文及其它成果

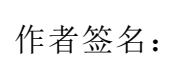
（一）发表的学术论文

[1] 荣河江，王亚东 基于基因本体的相似度计算方法[J]. 智能计算机与应用（已录用）

哈尔滨工业大学学位论文原创性声明和使用权限

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《基于自索引结构的高通量基因组重测序数据压缩算法》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：  日期：2018 年 6 月 21 日

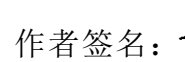
学位论文使用权限

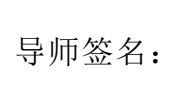
学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名：  日期：2018 年 6 月 21 日

导师签名：  日期：2018 年 6 月 21 日

致　谢

本课题是在导师王亚东教授的悉心指导和殷切关怀下完成的，在此衷心感谢导师对我的谆谆教导，以及对我在生活、工作、学习、研究等各方面的无私帮助。王老师严谨细致、实事求是的治学态度，认真勤奋、不知疲倦的工作作风，以及对事业的执着追求都将使我终生难忘，并时时鞭策我努力工作，在科学研究的道路上奋发向上。

特别感谢刘博教授在本课题中的悉心指导。

感谢实验室的汪国华、李杰、刘健等老师，他们对待科研工作一丝不苟，为我们树立了良好的榜样，并在日常的学习和生活中给予了我很多帮助，在此向他们表示衷心的感谢。

本课题整个实验过程中得到了实验室全体老师和师兄们的热情帮助和支持，感谢实验室姜涛、高阳、初砚硕、王芳、崔哲、官登峰师兄给予我的帮助，给我提出了许多中肯的建议和宝贵的意见。

感谢实验室的刘亚东、苏俊豪、张振东等同窗、师弟们给实验室带来的朝气和欢乐，他们使我每天的实验室生活都充满快乐。

感谢我的父母、我的家人对我工作的大力支持，感谢他们对我生活无微不至的关怀和照顾。