生物信息学，32，2016，i174-i182DOI：10.193/生物信息学/btw266

ISMB2016

OXFORD

德BWT：并行构造Burrows-Wheeler变换，用于用deBruijn分支编码大量收集基因组

刘波，† 朱棣贤† 和王亚东*

哈尔滨工业大学生物信息学中心，哈尔滨，黑龙江150001

*信件应送达给谁。

†提交人希望知道，在他们看来，前两位作者应被视为联合第一作者。

摘要

动机：随着高通量测序的发展，组装基因组的数量继续上升。 很好地组织和索引许多组装的基因组是至关重要的，以促进未来的基因组学研究。 Burrows-WheelerTransform(BWT)是基因组索引的重要数据结构，具有许多基本应用；然而，构建BWT用于大量基因组收集，特别是对于高度相似或重复的基因组，仍然是非平凡的。 此外，由于其增量性质，最先进的方法不能很好地支持可扩展的并行计算，这是使用现代计算机加速BWT构建的瓶颈。

结果：我们提出了一种新的基于分支的BWT构造方法。 De BWT创新地用一种新的数据结构——de Bruijn分支编码来表示和组织输入序列的后缀。 这种数据结构利用deBruijn图的优势，方便了具有长公共前缀的后缀之间的比较，打破了重复基因组序列BWT构建的瓶颈。 同时，deBWT还使用deBruijn图的结构来减少后缀之间不必要的比较。 基准表明，deBWT是通过并行计算为大数据集构建BWT的有效和可扩展的。 它非常适合于索引许多基因组，例如单个人类基因组的集合，具有多核服务器或集群。

可用性和实现：deBWT是用C语言实现的，源代码在或https://github.com/hitbc/deBWThttps://github.com/DixianZhu/deBWT

联系人：ydwang@hit.edu.cn

补充资料：补充资料可在生物信息学网上查阅。

1 导言

随着高通量测序的快速发展和普遍应用，许多基因组在尖端基因组学研究中得到了测序。一千块基因工程联合会，2015年英国10K集团，2015年 例如，1000个基因组()和UK10K()项目已经对成千上万个个体人类基因组进行了测序。 此外，随着测序成本的不断降低，例如。沃森，2014年 人类样本测序的成本已经低于1000美元()，基因组的数量在未来可能会急剧增加。 在这种情况下，很好地组织和索引大量的基因组是至关重要的，以促进未来的基因组学研究。

Burrows-Wheeler Transform(BWT；)是一种自索引数据结构布伦斯和惠勒，1994Ferragina和Manzini，2000年*Hon等人*。，2004Karkkainen，2007年林*等人*。，2008李和杜宾，2009年aLangmead和Salzberg，2012年Makinen等人。，2010Cox等人。，2012辛普森和杜宾，2012年李，2012年*Cox等人*。，2011具有许多基本应用，如基因组索引（；）、序列比对(；)、基因组压缩(；)、基因组组装(；)和测序纠错()。 然而，基因组序列的BWT构建是一项非平凡的任务。 主要是BWT结构的核心是确定输入序列的所有后缀的词典顺序)。特雷根和萨尔茨伯格，2012年 由于基因组()中可能存在许多重复序列，因此直接比较所有后缀以确定它们的词典顺序的成本将非常高。 这个问题对于构建许多高度相似的基因组的BWT来说更加严重，例如大量的

单个人类基因组，因为将有许多共同的序列，使整个输入更重复。

努力推进BWT建设。微笑和图平，2007年 由于输入序列的BWT可以直接从相应的后缀数组(SA)中导出，许多现存的SA构造方法()适用于这一任务。 然而，对于大的基因组序列，例如，记忆足迹可能是不实用的。 序列超过几十个Giga基对(GBP)，如所提出的SA构建方法通常需要将整个SA存储在内存中。 虽然也有一些提出的具有较小内存足迹（；）的SA构建方法，但是它们以速度为代价，因为它们需要使用外部内存。克雷瑟和费拉吉纳，2008年依等人。，2014

大多数最先进的BWT构造方法都利用了增量构造方法，这是基于一组排序后缀的相对词典顺序不会通过添加新后缀而改变的属性。尊敬的阁下等人。，2007 ()提出了第一种使用该属性的压缩后缀数组(CSA)构造方法。 主要是将输入序列对数分割成块，并在三个步骤中递增地构建CSA从最短到最长的后缀：（1）为一个新的后缀块构造SA；（2）根据旧块内后缀的相对顺序不变的属性，将新块内的每个后缀依次插入到旧块的CSA中；对于具有相同初始字符的后缀，CSA值单调增加；（3）合并新块和旧块以更新CSA。 在这种增量分块方法中，还提出了其他BWT构造方法，它们有各种实现。 ()提出了一种类似于()的BWT构造方法BWT磁盘，它还对输入序列进行对数分区。Ferragina等人。，2012Hon等人。,2007 但它对每个新块进行了修正DC3算法的排序，并利用BWT()的最后一次映射(L F-映射)属性来合并新的和旧的块。 Ferragina和Manzini，2005年刘等人。，2014年b()还以类似的方式提出了BWT施工方法ParaBWT。 它使用了一个最长的公共前缀表，方便对新添加的后缀进行排序，同时也融合了基于LF映射的新旧块。 该方法的主要贡献是实现了对新添加块的排序的并行化，有利于处理大的输入序列。 除了为一个或多个大序列构造BWT之外，这种方法还用于索引大量序列读取。 Bauer等人。 （2013）提出了BCR，一种构造大读取集BWT的算法。 它使用SA的特定分区，即。 通过它们在相应的排序读取上的位置将所有后缀划分为块。使用这个分区，读取的标记（表示为特定字符）可以完全用于提高块的排序和合并的效率。 ()还提出了一种类似的方法，RopeBWT2，提高了处理不同长度序列的能力。李，2014年

除了分块增量方法外，还有其他方法Karkkainen，2007年提议的办法。 ()提出了一种以不同的分块方式构造BWT的方法。*Liu*等人。，2014年a 它采样一组后缀作为拆分器，将所有后缀绑定到不同的块中，对于每个块，所提出的方法用差分覆盖样本(DCS)来处理所有后缀）。 ()提出了一种图形处理单元(基于GPU)BWT构造方法CX1，用于索引大量短读取。 该方法的主要思想是通过它们的初始k-mers来绑定所有后缀，并使用基于GPU的基数排序来寻址所有的回收箱。 这种方法主要是设计用于长度有限的读取，因为基数排序依赖于连接到读取的辅助字符。

增量分块方法的主要优点是，基于LF映射特性，它提供了一种有效的方法来比较具有长公共前缀的后缀，这对于大规模重复基因组的BWT构建至关重要。 然而，由于增量性质，这种方法不适合并行计算。 考虑到组装基因组的快速增加，输入序列将比以前大得多。 在这种情况下，用并行计算处理大序列是有利的，特别是现代服务器和集群比以前有更多的CPU核和RAM。 最近的研究为并行计算的BWT构建做出了努力。 对于大序列，ParaBWT实现了并行BWT结构；然而，结果表明，它是不可扩展的，即。 当几个线程（例如）时，加速会饱和。 使用八个线程。 这主要是由于增量分块方法的瓶颈。 作为一种非增量方法，CX1是可扩展的；然而，它对输入序列的长度有限制。

在此，我们提出了一种基于deBruijn分支的BWT构造函数(deBWT)，这是一种新的可伸缩并行BWT构造方法，它从deBruijn图(dBG)中支持)。 在以往的研究中()探讨了dBG与后缀trie之间的关系，但在BWT结构中仍未得到充分的应用。Marcus等人。，2014 德BWT的主要贡献是用dBG的性质来表示和组织输入序列的后缀，即输入序列(S)中相同k-mer的所有副本都会崩溃到序列(S)的dBG的同一顶点)。 deBWT的关键点是用一种新的数据结构de Bruijn分支编码来表示后缀，它是从输入序列的dBG的单路径导出的。 这种数据结构便于比较具有长公共前缀的后缀。 此外，deBWT通过其初始k-mers将整个BWT划分为块，并使用dBG的属性来避免对某些块进行不必要的排序，即。 根据图的拓扑结构，可以在恒定时间内导出某些块的BWT字符。

我们用不同的数据集和结果对deBWT进行了基准测试表明它具有快速的速度和对多个线程的良好可伸缩性。 特别是，deBWT非常适合于BWT构建一组高度相似的基因组序列，如多个人类基因组，这可能在未来的基因组学研究中有着广泛的应用。

2 方法

2.1 初步的

让DNA序列，G，是字母表R¼fA；C；G；TG上的序列，具有jGJ字符。 进一步地，我们假设要索引的序列是S¼G$，其中$是辅助字符，S的字母表的词典顺序是A<C<G和T<$。 此外，S½i]；我¼O；。。。；j Gj表示S的第i个字符，S½i；j]表示S的子串，从S½i开始]，从S½j结束]。

S的后缀是子串S½i；jGj]，i¼O；。。。；j GJ和SA是一个函数，SA½i]¼j，i¼O；。。。；j Gj，其中j是S的第一个最小后缀的起始位置$是S即B的字符的排列[illegible] − 1]，如果 *SA½i]* O，还有 *b*[illegible] ¼ $，否则。

的dBG，D$是一个有向图，其中顶点由G的所有k-mers组成。 每个顶点都表示为KM,,

i· 1；d. . . 在哪里D是不同的k-mers的总数。 对任何人来说 一对顶点 D，公里i；公里j 在那里 是个导演 边缘

公里i ! 公里j只有当KMi 和知识管理j 有一个k-1重叠，即。公里i½ k－1]¼公里j½ k－2]. 与 这个 定义， a 准备好 了 的 马克斯-
非支架路径可以从D导出。 在这里，最大非分支路径表示满足以下内容的路径
条件：(I)对于第一个顶点，内度为0或>1，以及

外度为1；(Ii)对于最后一个顶点，外度为0或>1，以及内度1；(Iii)对于所有其他内部顶点，内、外度正好是1()。 托梅斯库和梅德韦杰夫，2016年这种路径通常被称为“unipath”或“unitigs”，它们是com-
家利用于基因组的新组装(；Gnerre等人。，2011
Zimin等人。，2013托梅斯库和梅德韦杰夫，2016年；)。我们用过了
术语“unipath”在下面的章节中。 为了方便dis-
脑震荡，我们分配了D的每一个单径 一种身份，
uj¼1；；； . . . ·u· 在哪里 ·u· 是Unipath的总数。
在下面的小节中，我们介绍了基于单路径的BWT
施工方法首先(2.2节)，然后概述

de BWT（第2.3节)和关于方法各步骤实施情况的更详细信息(第2.4-2.7节）。

2.2 基于Unipath的BWT结构

2.2.1 后缀的unipath表示

的DNA序列，G可以用具体的行走来表示

d BG D，即。 它等于折叠D的顶点的特定有序列表的序列，公里；公里. . . ；公里 ，
每个知识管理的地方. . . 我¼0；jGj-k是相应的k-mer

列表中定D的特定边缘。 有了这个观察，我们有以下引理。

引理1： 一个DNA序列可以用一个有序的dBG单条列表来表示。

这个引理很容易通过折叠所有这些边来证明，

公里 ! 公里 在命令的列表中 公里；公里；. . . ；公里 ，
知识管理 和知识管理 分别是单出和单入ver-

冰。 因此， 的 命令 名单 可以 是 重新写作为
u；U. . . ；U 在哪里j是uni-的总数-
表示G的路径，每个U我¼1；；jU. . . j代表一个unipath
还值得注意的是，清单中的一些身份可能彼此相同，因为Unipath可以有多个副本。 为每个美国我们进一步将其在G上的起始位置定义为Unipath变化点(UCP)，UCP。 补充图1中有一个插图。 此外，我们有以下推论。

推论1：每个后缀都可以表示为一个有序的单路径列表，或者一个附加有序的单路径列表的特定单路径的子字符串。

的每一个后缀，S½i；jgj]，都可以直接表示

订购清单KMi；公里ip1. . . ；公里Gj- 在那里i 是COR-
在I位回答k-mer。 如果位置I是UCP，则k-mers的有序列表可以重新写入为有序列表

否则，它可以重写为

Unipaths，Uj；Ujp1；；
u. . .

2.2.2基于unipath的两个后缀比较

给定S，Suf的两个后缀 ¼S½i；jGJ]和Suf ¼S½j；jGj]，哪里
i 我<jGJ-k和j<jGJ-k，它们之间的比较两个后缀可以推断为以下两种情况。

首先，考虑到两个有序的k-mers列表，
公里；公里 ；. . ；公里 还有 公里；公里 ；. . . ；公里 如果
公里 公里两个后缀的词典顺序可以是很容易由它们的初始k-mers决定。

其次，如果知识管理 ¼公里比较更复杂。 在这种情况下，两个后缀是D上的两个行走 从一开始
相同的顶点，因为相同的k-mers的所有副本都会崩溃到
dBG的相同顶点。 因此，SI；UCP ip1 — 1 ¼ j； jp — 1，
因为这两个字符串是相应单路径的相同子字符串。 插图在里面。图1a

然后它成为对齐之间的迭代比较
单垫，即。 如果这两个 和美国jp1 不同的，不同的Unipath，U ip1
词汇顺序可以确定，否则我们需要COM-
在我们遇到两个不同的单径()之前，请注意以下单径。无花果。 1b

考虑到dBG的性质，如果这两个单位，U ip1 还有 u jp1 是相同的，是U的起始顶点 ip1p1 还有 u jp1p1 必须是具有相同前体的两个顶点；因此，第一个顶点
两个顶点的k-1字符必须相同，即。 比较一下
只能通过检查两个Uni-的k-字符来完成-
路径，即。 g ucp ip1p1 和GUCP jp1p1 þk-1 无花 bk-1
此外，如果U的最后一个顶点，(u)是一个单独的顶点
比较也可以省略，因为下一个单路径必须是同样的。

通过这些观察，我们设计了一种新的数据结构，称为deBruijn分支编码()。无花果。 1d 德布鲁因分支编码G，dE定义为所有这些char-的级联-
G½þk]；我¼0；. . . ；j Gj-k-1符合这一条件
公里 对应于多出顶点。 的每个角色
d E 也被称为分支字符。 然后，为每一个苏菲 ，
j<gj-k，我们定义了它的投影后缀 d E 作为
d E /ðjÞ；d E －1 其中/ðjÞ是dE 在S中j位置之后的第一个分支字符的坐标，以及 ·d E· 是长度
d E.

引理2：对于至少kbp长的两个后缀，它们的词典-
如果两个后缀的初始k-mers是相同的，则可以通过比较它们由DNA序列的deBruijn分支编码定义的投影后缀来确定其顺序。

这个引理很容易用上面提到的观察来证明，它提供了一个成本效益的解决方案，比较两个后缀与长公共前缀。

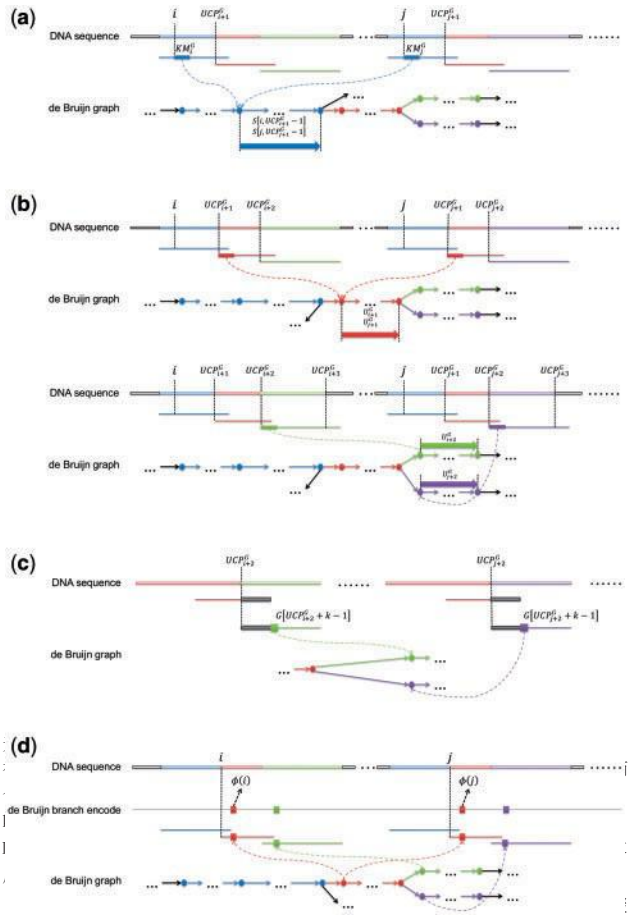
2.2.3BWT的k-mer分区

根据SA的定义，从相同的k-mers(假设它是KM)开始的后缀在SA中构成一个连续块，作为a
后缀具有不同的初始k-mer(例如， 公里必须更大或比从KM开始的所有后缀都小。 因此，给定D 可以将S的BWT划分为 ·d· þk个部分。 ·d· 他们的COR-

回应 ·d· D的K-MERS，即。 每个人 ·d· 零件在-
卷所有的后缀与一个特定的k-mers的D以及每一个

- h i σ 是UCP最近的
s 一; UCP^g − 1 ; U^g ; . j . ; U^g 在
哪里^g jp jp h jp
定位I下游，和SI；UCP^g $jp1$ − 1 i 是子串的
唯一的KM^g 在(补充图。 1)。 我们任期
此列表作为后缀的Unipath表示。

其余的k部分对应于一个特定的后缀，其起始
在S末端之前，位置小于kbp。 这个分区
(称为BWT的k-mer分区)可以分两步构造：(I)
对D的所有k-mers进行排序^g 以及S的最后k个后缀-
图形顺序；(Ii)将S的所有>kbp长后缀放入



在其unipath表示的相应位置上有两个不同的unipath时，才能确定词典的顺序，否则，需要更多的unipath。 在这种情况下，两个后缀具有相同的unipath(红色unipath)，连续到第一个unipath(蓝色unipath)，因此比较继续到第三个unipath。 对于第三个单径，可以确定词典的顺序，因为两个后缀在第二个单径的末尾分别指向两个不同的分支(绿色和紫色。 (c)由于d BG的性质，两个不同的单径在第一个k-mers时必须彼此不同。 此外，如果两个不同的k-mers具有相同的前驱，它们的第一个(k-1)字符必须与它们的前驱的最后一个(k-1)字符相同（图中的灰色段），

i.e。 这两个分支k-mers只有在它们的第k个字符上是不同的（图中标记为绿色和紫色的块）。 在这种情况下，它只需要比较两个单径的k-字符，以确定它们是否是相同的单径。 (d)有了这个属性，de Bruijn编码，d E^{g} 定义为连接所有这些字符的字符串，

G½xþk]；x¼0；。.. ；j Gj-k-1，沿着DNA序列，G，其中k-mer，公里^{g} 在G的位置x是多出顶点的副本。 每个G½xþk]s也被称为分支字符（在图中标记为彩色块）。 对于G的位置j，/ðjÞ定义为BrðjÞ在dE上的位置^{g} 其中BrðjÞ是下游的分支字符，最接近S中j的位置。

相应的部分由它们的初始k-MERS。 经过两步，的BWT结构成为· *d*· 子问题，即。 另外确定D中每个后缀的词典顺序^{g}由于在k-mer分区的构建过程中，不同k-mer的后缀的词典顺序已经被隐式地确定。

因此，对于每个D^{g} 部分对应于各种初始k-mer，任务是比较一系列具有相同初始k-mer的后缀，可以借助deBruijn分支编码dE来实现^{g}。 此外，该问题可以通过以下引理进一步简化。

引理3：如果一个顶点KM$_{j}$ D^{g} （一¼1；D.. . ^{g} ）是单入的，相应的BWT部分由jKM组成$_{j}$相同的字符，其中jKM$_{j}$j是KM的拷贝数$_{i}$ 在G中，除非该部分涉及第一个后缀。

这个引理是通过观察得到的，如果dBG的一个顶点是单入的，那么这个k-mer的所有后缀都必须具有相同的前一个字符，即。 这部分的BWT纯粹是jKM$_{j}$知识管理前体的第一个字符的j副本$_{i}$。 这是除了涉及第一个后缀的部分，因为第一个后缀的BWT字符是$的。 利用这个引理，只需要对多个顶点标记的部分进行排序。 因此，最多有美国^{g} 子问题，作为这样的顶点，必须是单路径的初始k-mers。

2.3 概述deBWT方法

与许多情况一样，输入不仅是一个，而且是一组DNA序列，如一组基因组、染色体或组装的连体，我们重新定义了要索引为S¼G的序列$_{1}$ g$_{2}$.. . g$_{nd}$$，在哪里N$_{d}$ 是输入DNA序列的数目，每一个 g$_{i}$；i¼1；；N. . . $_{d}$ 是一个特定的序列。 是另一个辅助字符，字母表变成A<C<G<T<<$。

德BWT在以下三个专业中构造了S的BWT阶段。

i. dB G构建和分析：deBWT构建所有的dBG涉及序列fG$_{1}$；g$_{2}$；. . . ；g$_{nd}$ 使用用户定义的参数k，对k-mers进行排序以构建BWT的k-mer分区，识别所有的单路径以及多出顶点和多入顶点，并解决与单入顶点对应的所有部分。

ii. 德布鲁因分支编码生成：deBWT计算S(dE)的德布鲁因分支编码^{s}），回收与BWT未解部分对应的S的所有后缀，并计算它们的投影后缀。

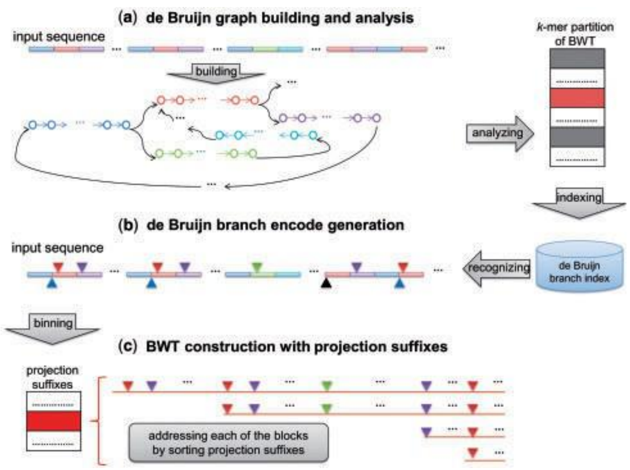
iii. 具有投影后缀的BWT构造：对于每个未解决的BWT部件，deBWT构建所涉及的投影后缀的SA，以确定该部件的BWT字符。

方法的示意图在。图2

2.4 建筑和分析

它需要所有的k-mers以及它们在fG中的拷贝数$_{1}$；g$_{2}$；. . . ；g$_{nd}$ 建立k-mer分区，解决单入部分，即。 一个k-mer计数任务。 在当前版本的deBWT中，水母(c,)三月AIS 还有 金斯福德， 2011 是 已使用。 作为 两者兼而有之 的 的 在后面的步骤中需要多输入和多输出顶点，deBWT计数dBG的边，即。 (kþ1)-mers，而不是k-mers。

为了构建BWT的k-mer分区，deBWT按其字典顺序对所有(kþ1)-mers进行排序，并分别将具有相同第一kbp前缀的所有(kþ1)-mers合并到k-mers中来构建分区。 对于每个k-mers，它的拷贝数是通过直接总结相应的(kþ1)-mers来计算的。



无花果。 2. 德BWT方法的示意图。 (a)De BWT最初使用用户定义的参数k构建输入序列的d BG，该参数确定顶点的大小。 然后对dBG进行分析，以构建BWT的k-mer分区，并识别所有的unipath(图中的彩色条表示输入序列的各种unipath的副本)。 使用unipath，所有的多入顶点和多出顶点都由基于哈希表的数据结构de Bruijn分支索引进行索引。 此外，所有的多个顶点都被标记。 在这种情况下，红色块表示dBG的“红色”单路径的第一个k-mer，它是一个多入顶点，灰色和白色块分别表示其他多入和-out k-mer。 (b)De BWT扫描输入序列以识别具有de Bruijn分支索引的分支字符（标记为输入序列上方的彩色反向矩形）并生成de Bruijn分支编码。 同时，具有初始k-mers的后缀对应于多个in顶点，即。 属于BWT未解决部分的后缀也用索引识别（标记为输入序列下面的彩色矩形）。 此外，对于未解决部分中的每个后缀，deBWT计算其/ð•Þ值以确定相应的投影后缀，并将其记录到deBruijn分支索引中。 使用de Bruijn分支索引，de BWT通过对投影后缀进行排序来处理所有未解决的BWT部分

同时，在合并过程中也可以识别出多出顶点，deBWT记录了在后面的步骤中构建deBruijn分支编码的所有多出顶点。

然后根据排序识别多个输入顶点

(kþ1)----名单。 也就是说，deBWT将排序的(kþ1)-mer列表划分为四个有序列表，每个列表对应于特定的初始字符A、C、G或T。 然后，de BWT通过列表上的四路合并来识别所有多个in顶点。 在合并过程中，两个任务同时完成，即。 如果一个顶点被识别为Singin，则deBWT将其分配给相应的(kþ1)-mer的第一个字符和解决BWT部分的拷贝数；否则，deBWT将k-mer记录在另一个数据结构中以生成后一步/ð•Þ功能。

值得注意的是，由于辅助字符#的存在，deBWT将所有在#之前和旁边至少有一个副本的k-mers分别识别为多出和多入k-mers。

这个步骤的并行化很简单。 那个k-mer

计数可以直接并行化(c,ai)。 三月和金斯福德，2011年有许多可行的并行整数排序方法来排序(kþ1)-mers，我们使用了一种简单的方法，即。 实现了一个基数排序，将所有(kþ1)-mers划分为块，并通过整数快速排序进一步并行处理每个块。 合并k-mer列表不是并行执行的；但是，成本与(kþ1)-mers的数量是线性的，并且不昂贵。

2.5 生成deBruijn分支编码和投影后缀

De BWT构建了一个基于哈希表的数据结构de Bruijn分支索引，以索引所有多入和多出k-mers(补充图。 2)。 利用该索引，通过扫描S一次同时生成deBruijn分支编码和/ð•Þ函数。

De BWT最初将空字符串分配为d E^{s}一个柜台CD E^s 记录dE的长度^s 和一个线性表，PDE^{s}它记录了k-MERS的所有副本的位置和/ð•Þ值(补充图。 2)。 每个多个k-mers占据一系列的PdE细胞^s 作为自己副本的子表。 每个子表都可以使用特定的指针访问。 然后，DeBWT从上游到下游扫描S，以检查每个k-mers。 对于S的位置I，如果相应的k-mer，KM^{s}是一个多个k-mer，deBWT记录字符S½i－将CDE和CDE值þ1进入相应的PDE子表；如果KM^s 是一个多出k-mer，deBWT附加分支字符S½i þ 去 d E^s 还有 更新CDE^s. 给你，S½i－是 的 BWT

分支字符必须是KM投影后缀的第一个字符^{s}和CDE^s þ1是它在dE上的位置^s.

本步骤的并行化实现如下。 德BWT将S分为P段，并将每个段分配给特定的线程以生成本地dE^s 和/ð•Þ值。 值P等于线程数。 值得注意的是，所有线程共享同一个哈希表和PDE^s 数据结构具有读写锁，但有自己的CDE^s 和dE^s。 当所有线程完成自己的任务时，deBWT会追加所有本地的dES，并更新线程p2生成的本地/ð•Þ值 f1； 2； ...；由下面的运算得到pg：假设长度为 d E^s 的 的 j-th 分段 是 l^{dE}_j; j2f1； 2； ...；帕格，为了 每个人 多个k-MERS，KM^{s}在第p段(p>0)中^s; cor-

响应/ð•Þ值/ðiÞ更新为/ðiÞþ ...

2.6 带投影后缀的BWT结构

对于对应于多个k-mers的BWT部件，deBWT用dE构造投影后缀的SA^s 和/ð•Þ函数。 SA是通过直接快速排序所涉及的投影后缀来构建的。 由于所有未解决的部分都是独立的，因此并行完成任务也很容易。

我们对原快速排序方法的递归过程进行了修改，以提高效率。 也就是说，对于传输到递归函数中的特定子数组后缀，deBWT首先检查所有后缀是否具有相同的BWT字符。 如果是这样的话，deBWT将子数组标记为排序，因为这些后缀的精确字典顺序对于BWT构造是不必要的；否则，deBWT调用原始递归函数来进一步排序子数组。 这种修改也可以看作是引理3的延伸。

2.7 额外处理

在上述所有操作之后，仍然有N_d ×k个未解决的BWT字符，每个对应于其中一个后缀，它们在#或$之前的起始位置小于k个。 这些后缀最初被搁置，de BWT独立地构建此类后缀的SA以填充BWT字符串。 由于这种后缀很少，这种排序是通过直接比较后缀的原始序列来实现的。

3 结果

我们用三个模拟各种实际应用场景的数据集对de BWT进行了基准测试。 一个数据集由10个硅质人类基因组组成(总共30.9GBP)。 每个基因组都是通过将1000个基因组()的特定样本的变体整合到人类参考基因组G RCH37/Hg19中产生的。 该数据集模拟了多个单个人类基因组的索引，在基因组研究中有许多应用。 数据集由一组模拟连体组成。 长期阅读测序技术，如单分子实时测序，已经改善了人类基因组组装的ContigN50，>1000万bp，我们随机从10个硅质人类基因组中提取了3000个序列(每个序列约10Mbp长，共30.2GBP)，其中包含一个内部脚本，该脚本是从Wgsim模拟器(；m)中修改的)。1000基因组计划联合会，2015年(http://www.pacb.com/blog/toward-platinum-genomes-pacbio-releases-a-new-更高质量的-chm1-组装到-ncbi/)，*Li等人。*，2009年bhttps://github.com/LH3/WGSIhttp://hgdownload.soe.ucsc.edu/down负荷 （三）数据集由八个灵长类基因组组成，包括长臂猿、大猩猩、猩猩、恒河猴、狒狒、黑猩猩、倭黑猩猩和人类(从：tml下载)。 该数据集评估了deBWT索引更多样化基因组的能力。

该基准是在一个有四个Intel的服务器上实现的Xeon E4820CPU（总共32核）在2.00GHz和1Ter字节RAM，运行Linux Ubuntu14.04。三月,AIS和Kingsford，2011年De BWT使用水母(版本2.1.4；c)来实现输入序列的k-mer计数(参数k配置为31)。 最近发表的两种方法，RopeBWT2()和ParaBWT(版本1.0.8-二进制-x86_64)()，也在同一数据集上进行了比较。李，2014年*Liu等人。*，2014年b

首先，我们用32个线程测试了de BWT的性能，表1i.e。 与服务器的所有32个CPU核心一起运行。 Para BWT也使用32个线程运行，但是Rope BWT2使用其默认设置运行，因为它不支持并行计算。 经过的时间()表明，deBWT和ParaBWT具有可比较的速度，而RopeBWT2则较慢，这可能是因为它不支持并行计算，而且该算法不适合长序列。 我们进一步研究了deBWT的处理，发现由于其输出文件的格式，deBWT的速度在很大程度上被水母减慢。 水母的默认输出是一个未发布格式的二进制文件。 作为关于格式的细节对我们来说是未知的，我们使用了“转储’命令Jellyfish将输出文件转换为文本文件，然后以我们自己的格式将文本文件转换为二进制文件，作为进一步步骤的输入。 这个文件转换花费了两个小时的所有三个数据集，即。 大约60%-70%的总运行

时间到了。 如果输出格式，时间成本将大大降低水母是可用的，或者如果其他k-mer计数工具与类似使用性能和可读输出格式。 扣除

表1。 具有32个CPU核的运行时间（以分钟为单位）

方法	人类基因组	人类连体	灵长类动物基因组
de BWT	134	129	330
de BWT（无转换）	48	56	100
第B WT段	241	262	180
绳索BWT2	1094	2247	1546

“de BWT”表示de BWT和“de BWT”(没有conver-西恩)“扣除水母输出格式转换的时间。

文件转换的时间，deBWT比其他两种方法快得多。

我们调查了deBWT()的各个步骤的时间成本。 主要是观察到两个问题。表2

首先，deBWT的大多数核心步骤，即。 k-mer计数、kmer排序、deBruijn分支编码和/ð•Þ值生成和投影后缀排序是有效的。 这是因为几个原因。 (i)de Bruijn分支代码大大降低了用长公共前缀排序后缀的成本。 我们研究了生成的deBruijn分支代码的长度，发现对于基因组和Contigs数据集，它们的长度分别比原始输入序列短一个阶。 在这种情况下，投影后缀之间的比较比原始后缀要便宜得多。 此外，BWT的k-mer分区也有助于减少许多不必要的比较操作。 这些步骤的设计适合并行计算，可以充分使用多个CPU核。 值得注意的是，除了并行实现deBWT的核心步骤外，最先进的k-mer计数工具也具有良好的并行化。 由于k-mer计数仍然是一个广泛应用的开放问题，这一步有几个选择。 Deorowicz等人。,2015表2我们还尝试了一个更新的发布工具KMC2（），并获得了更快的速度（）。 然而，KMC2也输出一个二进制文件，这是很难直接解释的。 如果最先进的k-mer计数工具具有一个易于解释的输出文件，这将是有益的。 （三）除了并行性外，还有多个步骤，即。 k-mer计数、k-mers的基数排序、deBruijn分支编码和/ð•Þ值生成具有准线性时间复杂度。

第二，I/O操作是减缓方法的主要问题。 除了上面提到的文件转换步骤之外，在“deBruijn图分析”和“附加处理”步骤中也有许多I/O操作。 也就是说，在dBG分析步骤中，deBWT需要合并记录k-mers四个有序列表的文件来识别多个k-mers；在附加处理步骤中，deBWT需要将从文本(fasta格式)索引的大序列转换为二进制格式，并合并各种BWT部分。 虽然这些操作理论上具有较低的时间复杂度，但它们也取决于计算机文件系统的性能以及程序的实现。

表2。 德BWT的各个步骤的时间（以分钟为单位）

步骤	人类基因组	人类的连体	灵长类的基因组
第一阶段：dBG建设与分析			
克默计数	16	16	26
文件转换	87	74	229
k-mer排序	3	3	8
d BG分析	8	7	19
第二阶段：产生德布鲁因分支编码和投影后缀			
德布鲁因分支编码和/ð•Þ价值生成	9	12	16
第二阶段：BWT建设与投影后缀			
投影后缀排序	4	12	10
额外处理			
用KMC2进行K-mer计数	7	8	12
额外处理	7	6	22

这也是我们今后进一步优化这些操作的重要工作。

除了上述两个问题外，投影后缀排序步骤的时间成本特别关键，因为它是处理输入序列内长时间重复的核心步骤)。 的

这一步的总时间成本是 $\sum_{j=1}^{T} U_j$ 在哪里$_j$ 是时候用于解决BWT的第j个未解决部分。 作为每个联合国-已解决的零件由流索处理，时间成本可表示如下（；）：宾利和塞奇威克，1997年Karkkainen，2007年

$$\sum_{i=1}^{n_j} DP\delta PS_i\text{þ}$$

，在哪里$_j$ 是数量未解决部分j和DPδPS中涉及的投影后缀$_i$þ是第一投影后缀的区别前缀的长度(表示为PS$_j$)的部分。这里，PS的区别前缀$_i$ 是PS的最短前缀$_i$这是确定相应部分的BWT字符所必需的。 对于高度相似的输入序列，G_1，g_2，...，g_{nd}，DPδPS的上限$_i$Þ是O

·d E^{g_m}。 在理论上，由于存在长时间的重复，在哪里g_m 是具有最长的deBruijn分支编码的输入序列-英和德^{g_m}是相应的deBruijn分支编码的长度。 例如，两个序列G$_i$ 和G$_j$ 可以几乎相同；因此，区分前缀的长度可以接近序列的deBruijn分支编码的长度。 作为G的分支字符的总数$_m$，这个值^{g_m}不仅取决于有多少个UnipathSG$_m$ 有，但也有多少副本的联合声明。

虽然理论上界较大，但DPδPS$_i$þ也很大程度上取决于基因组变异的分布以及输入序列的重复性，这可能会使其在实践中降低。 为了更准确地调查时间成本，我们评估了DP$_j$ 值和DP$''$ 基准中最重复的数据集的值(即。——10个人类基因组数据集)，其中DP$_j$ 和DP$''$ 分别是j-未解决部分中投影后缀的区别前缀的均值和最大长度。 一系列的分位数的DP$_j$ 和DP$''$ 显示了10个人类基因组数据集的值。表3 这些分位数表明，对于大多数块，区别前缀是短的，例如。 DP的0.90分位数$''$ 是11872，表明对于90%未解决的BWT部分，区分前缀的最大长度小于11872个字符。 通过简单的比较来确定他们的词典顺序并不昂贵。 因此，这一步的总体成本并不高，尽管仍然有一小部分（<0.1%）的BWT部件具有长的区别前缀(平均值>298K)。

表4我们还使用8、16、24个线程运行de BWT来研究它的可伸缩性。 结果()表明，deBWT可以随着线程的增加而逐渐加速，即。 它具有良好的可扩展性。 然而，ParaBWT的速度与线程上的各种设置几乎相同。 这可能是由于ParaBWT方法的增量性质，这可能限制了它在现代服务器和集群上的性能。 具有不同数量线程的deBWT的各种步骤的时间是在。图3 它表明这两个核心步骤，deBruijn分支编码和/ð•Þ值生成和投影后缀排序（图中的步骤4和5）是最可伸缩的步骤，即。 它们随着线程数量的增加而加快。 这一特性有利于用更多的计算资源实现该方法。

我们进一步在硅人基因组数据集上运行deBWT用k参数上的各种配置来研究其EF-fect。表5 从结果可以观察到，在很大范围的k参数上，即。 ¼23-31，总运行时间接近，但对于较小的k参数，时间消耗较高。 这可能是由于中等长的k-mers(如23到31-MERS)可能有类似的能力跨越短重复。 在这里

在这种情况下，dBG的结构没有太大的变化k配置，即。 图中有相似数量的单条径及其副本。 然而，当k较小时，unipath将更短，并有更多的副本，这将使deBruijn分支编码更长，需要对更多的投影后缀进行排序。 k>32可以有更好的跨越重复的能力，这可能会提高整体性能；然而，它需要更多的RAM空间，因为k-mer不能由一个64位存储细胞。

表6de BWT()的内存占用取决于方法本身和使用的k-mer计数工具。 水母和KMC2的内存使用是高度可配置的，我们将它们设置为使用相对较大的内存来尽可能快地完成k-mer计数步骤。 de BWT的三个阶段的主要RAM成本是不同的。在第一阶段，主要成本来源于k-mer排序的数据结构。 简单地说，de BWT使用像PDE这样的线性表^s 要绑定所有k-mer；但是，表的每个单元格需要16个字节来记录k-mer的字符串及其拷贝数。 二期费用较为复杂。 需要同时将输入序列，deBruijn分支索引和生成的deBruijn分支编码保存在内存中。 因此，内存使用主要取决于几个问题，即。输入序列的大小、多输出和-ink-mers的数目以及多输出和-inkmers的副本数。 最后两项分别确定deBruijn分支编码的长度和未解决后缀的数量，需要记录在内存中。 多重输出和-ink-mers及其拷贝的数量与输入基因组的重复性密切相关。 我们对这两个人类数据集（因为它们更重复)进行了统计，并() 观察了两个问题。表7

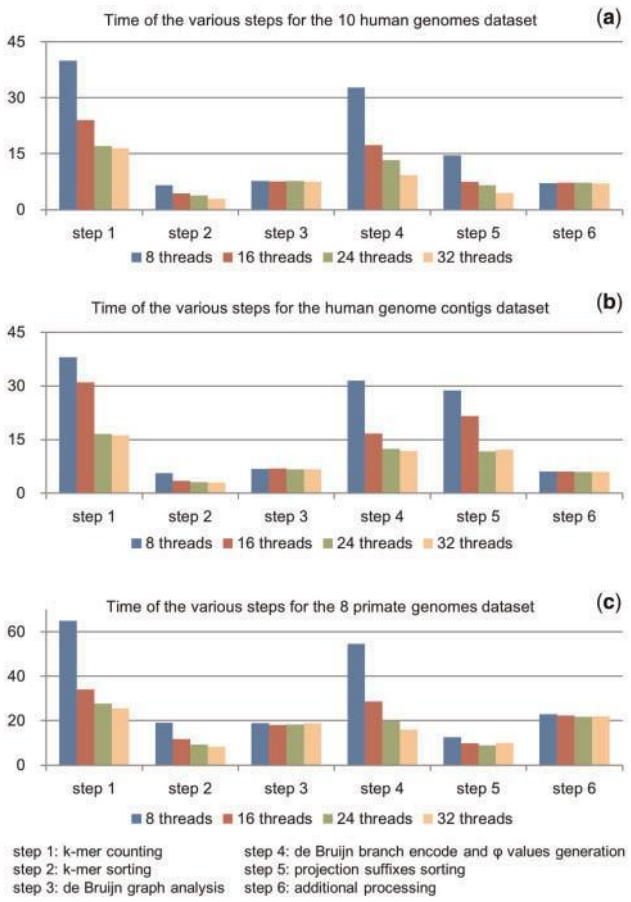
表3。 DP的数量$_j$ 和DP$''$ 十个人的价值观-奥美斯数据集

数量	0.50	0.90	0.95	0.99	0.999	0.9999
DP$_j$	107	588	2382	95 019	298 598	515 006
DP$''_j$	1760	11	238	1 925	3 232	3 387

表4。 使用不同数量的线程运行时间（以分钟为单位）

方法	8个线程	16条线	24条线	32个线程
人类基因组				
deBWT	194	153	142	134
de BWT（无转换）	109	68	56	48
Para BWT人类连体	265	240	240	241
de BWT	183	154	123	129
VDE BWT（无转换）	116	86	56	56
第B WT段	294	277	276	262
灵长类的基因组				
de BWT	423	355	332	330
de BWT（无转换）	193	125	105	100
第B WT段	196	182	181	180

“de BWT”表示de BWT和“de BWT”(没有conver-西恩）‘扣除水母输出文件的格式转换时间。



无花果。 3. 时间消耗的各个步骤的deBWT。 酒吧尊重-
请注明deBWT的各种步骤的经过时间(以分钟为单位
10个人类基因组数据集(A)、人类基因组序列数据集(B)和8个灵长类基因组数据集(C)。 相同颜色的条形对应于特定数量的线程，即。 蓝色、红色、绿色和紫色条分别为8、16、24和32螺纹

首先，对于这两个数据集，倍数和数量
在k-mers中，远小于jSj，即。 输入序列的字符数。 因此，与整个输入序列相比，哈希表的成本并不昂贵。 此外，还值得注意的是，对于高度相似的基因组，与所涉及的基因组的增量相比，多出和-ink-mers的数量增加要小得多，因为有许多常见的序列，它们不会在dBG中引入新的分支。

二，多出和-ink-mers的拷贝数
也低于jSJ，尽管人类基因组是重复的。 在这种情况下，deBruijn分支编码可以看作是一个比S短的DNA序列，因此空间成本不大。 主要成本来源于多个k-mers的副本，因为它需要记录/ð•Þ值和BWT字符，每个副本都有几个字节。

第三步的RAM成本也与第二阶段相似。 要对投影后缀进行排序，需要保留deBruijn分支编码和/ð•Þ值以及RAM中多个k-mers副本的BWT字符。

4 讨论

许多基因组的良好组织和索引将在未来的基因组学研究中受到广泛的需求，随着组装的迅速增加

表5。 在k参数上具有各种配置的IN硅人基因组数据集的运行时间（以分钟为单位）

方法	k¼19	k¼23	k¼27	k¼31
de BWT	142	124	131	134
de BWT（无转换）	75	51	47	48

“de BWT”表示de BWT的运行时间，“de BWT（无转换）”扣除水母输出文件格式转换的时间。

表6。 内存足迹与32个CPU核心（千兆字节）

方法	人类基因组	人类的连体	灵长类的基因组
de BWT	120/78/38	120/63/34	235/203/58
第B WT段	30	30	29
绳索BWT2	30	24	40
补充KMC2	119	119	119

对于内存列中deBWT的‘x/y/z’，x、y和z值分别表示水母、deBWT的第1阶段和deBWT的第2阶段和第3阶段的内存足迹。

表7。 关于硅质人类基因组和连体的统计

统计数字	人类基因组	人类的连体
输入序列的长度	30955436371	30200003020
不同的k-mers	5073730669	4025285321
多出k-mers	18820763	17238123
多个k-mers	18821805	17237511
多输出k-MERS的副本	2364004617	2301293218
多个k-mers的副本	2364445711	2300904807

基因组。 作为重要的基因组索引数据结构，BWT可能有许多应用；然而，为大量基因组构建BWT，特别是高度相似的再测序基因组(例如。 许多人类个体基因组)，仍然是一项非平凡的任务。 此外，由于最先进方法的增量性质，很难用可伸缩的并行计算构造BWT。 这是充分利用现代服务器或集群的计算资源来处理大量数据的瓶颈。

我们提出了一种新的并行BWT构建方法deBWT，以打破瓶颈。 德BWT的主要贡献是它基于dBG的后缀表示和组织，这有利于后缀与长公共前缀的比较，避免不必要的比较。 此外，由于其非增量设计，deBWT对各种计算资源具有良好的可扩展性。 这些特性使得deBWT非常适合于构建具有现代服务器或集群的高度相似或重复基因组的大型集合的BWT。在实验中，deBWT在检索多个单个人类基因组和连体的速度上取得了实质性的提高。 更多样化的基因组，例如。 多个灵长类基因组，deBWT也显示出更快的速度和更好的并行化；然而，改善较小，可能是由于dBG的密度较低。 也就是说，有更多的k-mers和unipath需要处理，但输入的总体重复性低于高度相似的基因组。

与最先进的方法相比，deBWT有明显的-
更大的内存占用。 有潜在的解决方案来减少deBWT各个阶段的记忆足迹。

对于第一阶段，将k-mers合并成几个子集是可行的
并分别对内存有限的每个子集进行排序。 的

多个子集的结果可以直接合并到具有小内存空间的所有k-mers的有序列表中。

对于第二阶段，也可以通过只保留/ð•Þ值和BWT字符的比例来减少内存占用，这可以用以下策略来实现。 因为在第二阶段之前，所有的多入k-mers及其拷贝数都是已知的，所以它可以将整组多入k-mers划分为几个子集。 每个子集都有有限数量的k-mer副本。 因此，第二阶段可以在输入序列上多次扫描，而不是一次。 在每次扫描时，只识别、记录和输出相应子集中的多个k-mers的副本，并将其输出到具有有限RAM空间的特定文件中。 由于所有子集在第三阶段是相互独立的，所以可以分别处理子集的文件以生成BWT的各个部分。 进一步地，可以将BWT部件直接合并完成施工。 这一战略对于有限的工作空间是可行的，但由于需要多次执行第2阶段而牺牲了时间。

对于第三阶段，它还可以在内存中只保留BWT分区未解决的比例，因为所有这些分区都是独立的。

在deBWT上有两个可能的改进，这对我们来说是未来的重要工作。

首先，de BWT通过快速排序直接对投影后缀进行排序。由于deBruijn分支编码也可以看作是一种特殊的DNA序列，因此也可以使用其他方法进一步加速投影后缀排序步骤。 例如，利用DCS提出的方法加速了原始输入序列的二进制后缀的排序。 Karkkainen（2007年）这种方法也可以用于分类二值投影后缀，而不会丧失并行计算的能力，因为它是非增量的。

第二，对于当前版本的deBWT，I/O密集型步骤仍然没有优化，这减缓了速度。 我们计划进一步优化I/O密集型步骤，以提高deBWT的效率。 同时，由于k-mer计数仍然是一个开放的问题，先进的k-mer计数工具正在()开发中，我们还计划用其他更先进的k-mer计数工具代替水母，或者通过直接访问默认的水母输出文件来删除文件转换步骤，打破方法的实际瓶颈。Perez等人。，2016

供资

这项工作得到了国家自然科学基金（编号：61301204和31301089）、国家高技术研究开发计划(863）（编号：2015AA020101、2015AA020108和2014AA021505)和国家的部分支持科技重大项目(编号：2013ZX03005012)。

利益冲突：无申报。

参考资料

鲍尔，M。 j.等人。 （2013年）用于构造和反转字符串集合的bwt的轻量级算法。 *西奥。 Comput。 SCI，483，134-148。*

宾利，J。 L.和Sedgewick，R.（1997）排序和搜索字符串的快速算法。 在第8届离散算法学术研讨会论文集ACM，旧金山，加利福尼亚州。

伯罗，M。 还有惠勒，司法部。 （1994）块排序无损数据压缩算法。 技术报告124，数字设备公司，加利福尼亚州。

考克斯，A。 j.等人。 （2011年）海报摘要：无假设检测剪接JNC-*RNA-Seq数据中的问题。 在CSHL基因组信息学会议记录。 春港，纽约。*

考克斯，A。 j.等人。 （2012年）用Burrows-Wheeler变换大规模压缩基因组序列数据库。 *生物信息学，28，1415-1419。*

克雷瑟，A。 和费拉吉纳，P。 （2008）外部存储器中后缀阵列结构的理论和实验研究。 *算法，32，1-35。*

Deorowicz，S。 等人。 （2015年）KMC2：快速和资源节约的k-mer计数。 *生物信息学，31，1569-1576。*

费拉吉纳，P。 和曼齐尼，G。 （2000年）机会数据结构与应用。 In：第41届计算机科学基础学术年会论文集)。 加州雷东多海滩，390-398。

费拉吉纳，P。 和曼齐尼，G。 （2005年）压缩文本索引。 j ACM，52，552-581。

费拉吉纳，P。 *等人。* （2012年）外部内存中的轻量级数据索引和压缩。 *算法，63，707-730。*

格内尔，S。 等人。 （2011年）从大量平行序列数据中提取哺乳动物基因组的高质量草稿。 *Proc。 纳特尔。 阿卡德。 SCI。 美国，108，1513-1518。*

尊敬的W。 k.等人。 （2004年）压缩后缀阵列和FMIndex在搜索DNA序列中的实际方面。 *In：第六次算法工程与实验研讨会论文集和第一次分析算法与组合研讨会论文集。* 新奥尔良，洛杉矶，美国，2004年，pp。 31–38.

尊敬的W。 k.等人。 （2007年）构建带有大字母的压缩后缀数组。 *算法，48，23-36。*

卡尔凯宁，J。 （2007年）通过分块后缀排序在小空间中快速BWT。 *西奥。 Comput。 SCI，387，249-257。*

林，T。 w.等人。 （2008年）DNA压缩索引和局部比对。 *生物信息学，24，791-797。*

朗米德，B。 和Salzberg，S。 L.（2012）与Bowtie2的快速间隙阅读对齐。 *纳特。* 方法，9，357-359。

李，H。 和杜宾，R。 (2009a)与Burrows-Wheeler变换快速准确的短读对齐。 *生物信息学，25，1754-1760。*

李，H。 等人。 (2009b)序列对齐/映射格式和SAMtools。 *生物信息学，25，2078-2079。*

李，H。 （2012年）探索单样本SNP和indel调用与全基因组从头组装。 *生物信息学，28，1838-1844。*

李，H。 （2014年）快速构建长序列读取的FM索引。 *生物信息学，30，3274-3275。*

刘，C。 等人。 (2014a)GPU加速BWT结构，用于大量收集短读。 阿希夫，1401.7457

刘，Y。 等人。 (2014年b)并行和空间效率高的构建Burrows Wheeler变换和后缀数组用于大数据组数据。 IEEE/ACM Trans。 Comput。 *比奥尔。 生物信息，出版。*

Makinen，V。 等人。 （2010年）高度重复序列收集的存储和检索。 *J.Comp。 比奥尔，17，281-308。*

马克,AIS，G。 和金斯福德，C。 （2011年）一种快速、无锁的方法，有效地并行计算k-mers的发生情况。 生物信息学，27,764-770。 马库斯，S。 *等人。* （2014年）分裂MEM：泛基因组ANA的图形算法-带有后缀跳过的裂解。 *生物信息学，30，3476-3483。*

农，G。 等人。 （2014年）使用d关键子串在外部内存中构建后缀数组。 *ACM Trans。 通知。 第32章，第1条。*

佩雷斯，N。 *等人。* （2016年）k-mer计数算法的计算性能评估。 *j。 Comp。* 比奥尔，23，248-255。

辛普森，J。 T.和杜宾，R。 （2012年）利用压缩数据结构高效地重新组装大型基因组。 *基因组研究，22,549-556。*

斯迈思，W。 F.和Turpin，A。 H.（2007）后缀数组构造算法的分类。 *ACM Comput。 Surv.，39，第4条。*

1000基因组计划联合会（2015年）人类遗传变异的全球参考。 自然，526，68-74。

英国10K联合会（2015年）英国10K项目确定了健康和疾病中罕见的变异。 自然，526，82-90。

Tomescu，A。 I.和梅德韦杰夫，P。 （2016年）通过Omnitigs安全和完整的连体组装。 重组2016，INBI，9649：152-163

崔根，T。 J.和萨尔茨伯格，S。 L.（2012）重复DNA和下一代测序：计算挑战和解决方案。 *纳特。 Rev。* Genet，13，36-46。

华生，M。 （2014年）照亮DNA测序的未来。 *基因组生物。，*15，108.

子民，A。 v.等人。 （2013年）Ma Su RCA基因组汇编程序。 *生物信息学，*29，2669–2677.