**用大型凸台构造压缩后缀阵列*s***

永凯议员1林德华1，Kunihiko Sadakane2和宋永金3

1 香港大学计算机科学及资讯系统系，

wkhon，twlam@csis.hku.hk

*{ }*

2 日本九州大学计算机科学与通信工程系，

[sada@csce.kyushu-u.ac.jp](mailto:sada@csce.kyushu-u.ac.jp)

3 新加坡国立大学计算学院，

[三星@comp.nus.edu.SG](mailto:ksung@comp.nus.edu.sg)

**摘要。** 最近在压缩后缀数组方面的研究已经产生了两种突破性的索引数据结构，即压缩后缀数组(CSA)[7]和FM索引[5]。 其中任何一种都使在主存储器中存储全文索引成为可能，即使对于具有数十亿个字符(如人类DNA)的文本数据也是如此)。 然而，用有限的工作内存（即不构造后缀数组）构造这样的索引数据结构并不是一项琐碎的任务。 本文解决了这个问题。 目前，只有CSA允许一种空间效率高的构造算法[15]。 对于字母表上长度为n的文本Tσ， 这个 算法 需要 *o(Σn* 日志 *n)* 时间到了 还有 (2小时0 +1+g)n位 工作空间，其中H0 是T的0阶经验熵而g是任何非零常数。 当字母表大小Σ较小时，该算法就足够好了。 对于包含蛋白质、汉语或日语的文本数据来说，这是不实用的，因为字母表可能包含多达几千个字符。



*| |*

*| |*

本文的主要贡献是一种新的算法-结构CSA在O(N日志 n)时间使用(H0 +2+ g)n位工作空间。 注意，我们算法的运行时间与alpha-无关-打赌 尺寸 还有 的 空间 需求 是 更小 作为 它 是 很可能 那个 *h*0 > 1. 本文还对FM指数的空间效率建设做出了贡献。 我们证明FM指数确实可以在O(N)时间内直接由CSA构造。

# 1 导言

信息技术和生物技术的进步产生了大量的文本数据。 特别是，许多这样的文本没有词

*s*这项工作得到了香港研资局赠款HKU-7024/01E、日本教育、科学、体育和文化部赠款和NUS学术研究赠款R-252-000-119-112的部分支持。

T.Ibaraki，N.Katoh和H.Ono(Eds。 )：ISAAC2003，LNCS2906，pp。 240–249,2003.

§c 斯普林格-维拉格柏林海德堡，2003年



边界。 典型的例子包括DNA、蛋白质、汉语和日语。 简单的文本可以包含数百万甚至数十亿个字符。 为了帮助用户有效地定位所需的信息，必须使用一些数据结构对文本进行索引，以便利用快速搜索算法。 对于具有单词边界的文本（例如英语），可以使用倒置索引[9]；该数据结构允许快速查询，并且具有空间效率。 但是，不能对没有单词边界的文本使用倒置索引。 在这种情况下，后缀树[19]和后缀数组[17]是最有用的数据结构。 它们在许多领域都有应用，包括数字图书馆[20]、文本数据挖掘[24]和生物研究[10]。

后缀树是一个强大的索引，因为它允许我们使用时间线性地搜索模式长度，而不依赖于文本大小。 对于字母Σ上的n个字符序列，构建后缀树需要O(N)时间。 然后，模式P可以位于O(P+oc)时间，其中oc是出现的次数。 对于后缀数组，构造和搜索分别需要时间O(N)[3,13]和O(Plogn+oc。 然而，后缀树和后缀数组需要大量的内存空间。 这两种数据结构都需要O(nlogn)位；然而，与后缀数组相关联的常数较小。 对于人类DNA（长度约30亿），最著名的后缀树和后缀数组的实现分别需要40G和13G[14]。 用于从中国新闻来源获得的几年的文本数据1 [6]，后缀树和后缀数组分别需要18G和8G。 这种内存需求远远超过普通计算机的容量(现在PC机的主存容量高达4G)。 为了解决内存空间问题，提出了将索引数据结构存储在二次存储[2,12]中。 但搜索时间恶化很多。

| |

| |

最近，Grossi和Vitter[7]提出了一种名为压缩后缀数组(CSA)的空间高效索引数据结构，将空间需求从O(nlogn)位降低到O(nlogΣ)位，更重要的是，压缩后缀数组仍然支持高效搜索。 准确地说，搜索时间最多增加了logn的一个因子。 在实践中，用于人类DNA的CSA占用了大约2千兆字节，并且可以将CSA存储在PC的主存储器中。

| |

构造压缩后缀数组，一种天真的方法是先构建后缀数组，然后将其转换为压缩后缀数组。 这种构造方法需要O(nlogn)时间，但它需要的工作空间远大于O(nlogΣ)位。 换句话说，在PC的主记忆中存储人类DNA的CSA是可行的，但在PC上构建它是不可行的。 这绝对限制了CSA的应用。 为了解决这个工作记忆问题，Lam等人。 [15]开始直接构造压缩后缀数组的研究。 特别是，他们给出了一种算法，它只使用O(n日志Σ)位内存，并在O(Σn日志n)时间内运行。 该算法可以在20小时内在PC上对整个人类DNA进行索引。 [15]的想法是建立压缩

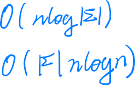
| |

| |

| |

1 消息来源为台湾中央通讯社。





242 W.-K.Hon等人。

递增后缀数组，逐字符。 借助修改后的红黑树，每插入一个字符，压缩后缀数组可以在O(Σlogn)时间内更新。 由于文本有n个字符，所以可以使用O(Σnlogn)时间获得压缩后缀数组。

| |

| |

对于DNA（只有四个不同的字符）或非常小的字母的文本，上述算法的效率仍然是可以接受的。 但是，对于其他类型的文本，如中文或日文，其字母至少由几千个字符组成，上述算法的效率迅速下降。 本文给出了一种更实用的构造压缩后缀数组的算法，将时间从O(Σnlogn)减少到

O(nlog|Σ|)位空间复合体

O(n log n)，

| |

同时保持 y.2事实上，我们的算法

需要较少的工作空间；更准确地说，它减少了(2H0 +1+s)n空间要求[15](H0 +2+s)n，其中s是任何非零常数，和H0 是文本的0阶经验熵。 注意H0 至多是原木Σ，很可能大于1。 因此，我们的算法允许我们在普通PC上索引比以前的算法更长的文本。

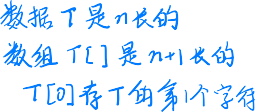
| |

新算法不是按字符构建CSA字符，而是将文本划分为日志n段，其中每个段的长度为n/logn。假设我们已经为第一个I段构建了一个CSA，我们表明我们可以在O(N)时间内将(I+1)-第段与CSA“合并”以形成更大的CSA。 因此，构建整个压缩后缀数组的总时间是O(nlogn)。 从技术上讲，新算法避免了使用短序列的平衡树作为中间表示。 这种短序列的更新本质上是缓慢的。 相反，我们设计了一个基于Rice代码[21,8]的更优雅的表示。



除了CSA之外，Ferragina和Manzini[5]还提出了另一种称为FM-index的方案来压缩后缀数组。 FM索引可以与CSA的存储需求和搜索效率相匹配，在某些情况下甚至更好。 然而，在文献中也没有直接的方法来构建FM指数。 本文的另一个贡献是第一个使用有限工作空间构造FMindex的算法。 我们表明，一旦构建了CSA，我们就可以在O(N)时间内从CSA构建FM索引，而无需额外的工作存储。

本文的其余部分组织如下。 第二节回顾了后缀数组和压缩后缀数组。 第三节介绍了CSA的新构造算法，第四节讨论了FM索引的空间效率构造。 我们在第五节中总结了这篇论文。



# 预科

让Σ成为一个字母，$是一个特殊的字符，而不是在ΣWe假设.

$在词典上比Σ中的任何字符都要小，它用来标记文本的末尾。 考虑在Σ上定义的长度-n文本T，它是表示的

2 对文本索引构造算法的探索是为大的阿尔法赌注量身定做的，这不是一个新的想法。 例如，Farach[3]将线性时间后缀树构造算法从常量大小的字母表扩展到一般字母表。

由数组T[0.n]=T[0]T[1].T[n]，其中T[n]=$。 对于我=0，1，2，...，n，T*i* =T[i.n]=T[i]T[i+1].T[n]表示从第一个位置开始的T的后缀。

|  |  |  |
| --- | --- | --- |
| *i* | *T[i]* | *ti* |
| 0 | *a* | *$* |
| 1 | *g* | *卡奇$* |
| 2 | *a* | *$* |
| 3 | *a* | *会计$* |
| 4 | *g* | *CCG$* |
| 5 | *g* | *CG$* |
| 6 | *c* | *g$* |
| 7 | $ | $ |

|  |  |  |
| --- | --- | --- |
| *i* | SA[i] | *t*SA[i] |
| 0 | 7 | $ |
| 1 | 2 | *$* |
| 2 | 0 | *$* |
| 3 | 3 | *会计$* |
| 4 | 1 | *卡奇$* |
| 5 | 4 | *CCG$* |
| 6 | 5 | *CG$* |
| 7 | 6 | *g$* |

|  |  |  |
| --- | --- | --- |
| *i* | Ψ[i] | T[SA[i]] |
| 0 | 2 | $ |
| 1 | 3 | *a* |
| 2 | 4 | *a* |
| 3 | 5 | *a* |
| 4 | 1 | *c* |
| 5 | 6 | *c* |
| 6 | 7 | *c* |
| 7 | 0 | *g* |



**无花果。 1.**后缀、后缀数组和acaaccg$的压缩后缀数组

{ }

| |



后缀数组[17]是T的n+1后缀的排序序列，由SA[0.n]表示]。 形式上，SA[0.n]是整数集0、1、...、n的排列，因此，根据词典顺序，T萨[0] *<*萨[1] *<...<*SA[n]. 。 有关示例，请参见图1。 即SA[i]表示T注的第i个最小后缀的起始位置，SA[0]=n，这样对于剩余的SA值，每个都可以用logn位表示，因此后缀数组可以使用(n+1)logn位存储。3 给定文本T和后缀数组SA[0.n]，可以在不再扫描T的情况下找到T中任何模式P的出现。 准确地说，它需要O(Plogn+oc)时间，其中oc是发生的次数。

对于每一个我=0，1，2，...，n，定义SA*−1*. 作为整数j，使SA[j]=我直观地，SA*−1*[i]表示T的词序*i* 在T的后缀中，T的后缀数在字典上小于或等于后缀T*i*.

一般来说，为了表示字符串X在T的后缀之间的关系，我们使用阶(X，T)来表示T的所有后缀之间X的lex-order。 因此，SA*−1=顺序i(t)。*

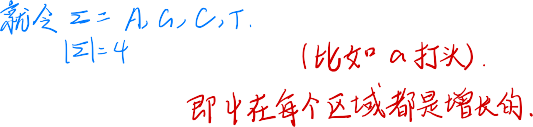
基于SA和SA*−1*，文本T的压缩后缀数组[7,22]存储一个数组Ψ[0.n]，其中Ψ[i]=SA*−1*对于我来说[SA[i]+1]=1，2，...，n，而Ψ[0]被定义为SA*−1*[0]。 有关示例，请参见图1。

注意，Ψ[0.n]包含n个+1个整数。 存储函数的一种简单方法需要(n+1)日志n位，这与存储SA的空间相同。 然而，Ψ[1.n]总是可以被划分为严格增加Σ序列，这允许它简洁地存储。 图1中最右边的表说明了这种递增性质，并基于以下引理证明了正确性。



| |

3 在本文中，我们假设对数的基数是2。



244 W.-K.Hon等人。

**引理1。** *（[15]）对于每一<j，如果T[SA[i]]=T[SA[j]]，则Ψ[i]<[j]。*

对于任何字符c∈Σ，定义l*c* 是最大p，使TSA[p−1] *<c*

和r*c* 是最大p，使TSA[p] ≤c。



**引理2。** （[15]）如果l*c* ≤r*c，Ψc 。rc正严格增加。*

**推论1。** *Ψ[1.n]至多可以被划分为严格增加Σ序列。*

| |

基于增加性质，Grossi和Vitter[8]表明Ψ可以编码在(H0 +2+o（1）n位使用Rice代码[21]。 而且，每个Ψ 价值 可以在O（1）时间内检索。 基于他们的编码方案， 我们 观察另一个财产，其说明如下。

**引理3。***. 假设我们可以为i=1，2，...，n依次枚举Ψ[i]，然后在O(N)时间内，我们可以使用(H)对Ψ函数进行编码*0 +2+o（1）n位。



# 我们的算法

我们想为文本T构造Ψ[1，n。*...* 要做到这一点，我们首先将文本分成n/A连续区域，例如T 1*，T* 2*，tn/A*, 每个人 与 *a* )人物。4 该算法从T函数开始，逐步建立Ψ函数*n/A*然后T*不适用−1tn/A,...* 最后得到所需的Ψ函数 *t* 1*t* 2 *...tn/A* =. 在那里 都是 二 主要 步骤。

=o( *n*

日志

* 1. 基本步骤：计算T的Ψ函数*n/A*.
  2. 合并步骤：对于我=n/A1，n/A2，。。。，计算T的Ψ函数*it j* 基于T*i* 以及T的Ψ函数 *j*在那里 *j* 表示字符串 *t我+1t*我+2 *...tn/A*.

− −

## 时间复杂性：基本步骤

求T的Ψ函数*n/A*, 我们 首先 计算 的 萨 为了 *tn/A* 通过 后缀排序。 从那以后 的 长度 的 *tn/A* 是 *a，*  后缀 分类 带走 *o(a* 日志 *a)*  时间到了 [16]。 后来，SA*−1* 函数可以在O(A)时间内计算。 然后结合SA函数，得到O(A)时间内的Ψ函数。 因此，基本步骤的总时间为O(日志A)。

## 时间复杂性：合并步骤

想想我1，2，。 *. .* ，n/A1，让T*i* =c1*c*2 *ca*。 让苏夫1*，苏菲*2*, . . . ，苏菲a* 是T的最长后缀*it j*。 没错，苏菲*k* =c*kc*k+1 *cat j 对于k=1，。。。a。*

···

∈ { − } ···

迭代步骤可分为三个部分：

4 也就是说，我们让T *i...* =T[(i−1)A]T[(i−1)A+1]T[i A1]。



1. 排序后缀suf1*，苏菲*2*, . . . ，苏菲a* 在它们之间找到每个后缀的词汇顺序。



1. 为每一个冲浪者*i*计算顺序(SUF*i，Tj*).
2. 计算T的Ψ函数*it j*.

第(a)部分可以基于以下引理来完成。

**引理4。** *最长的T后缀it j 可以在O(A日志A)时间排序。*



让SA吧 *t*  表示T的后缀数组 *j*，萨*−1* 表示SA的逆 *t* ,

*t tt t*

和Ψ*t t*  表示T的相应Ψ函数 *j*。 对于任何字符c，我们使用l的表示法*c* 和r*c* 表示最大p，使Ts*j* a*t t* [p−1] <c和最大p，使Ts*j* a*t t* [p] 分别≤c。 下面的引理显示了如何计算顺序(C*kc*k+1 *cat j，Tj*)对于k=A，A1，。。。因此



··· −

实现(b)部分)。



**引理5。** *设X为任意字符串，c为任意字符。 设B表示集合*

{b|b∈*c，rc*]∧ψ*t t [b]≤顺序(X，T j*)}. *然后，*

.*j订单(cX，T)=*

*lc* − 1 *如果B是空的*



最大{b|b∈B} *否则*

**引理6。** *给定T的Ψ函数 j。 计算顺序(Ckc*k+1 ···c*lt j，Tj*)

*对于所有1≤k≤A可以在O(Alogn)时间内完成。*

接下来我们讨论(C)部分，它计算T的Ψ函数*it j*. 注 就这样 定义， a ψ 职能 是 基本上 a 数组 的 lex-orders 的 的 后缀 T*it j* 其中 他们自己， 列举 在 一些 具体的 命令。 去 计算 ψ 为了

我们必须知道订单*it j*)的每个后缀s *j*。 我们也需要



*tit j*,

在下面，我们提出了一个函数f，它映射T的现有后缀的lex-order *j* 在 *t j* 去 那些 在 *tit j*, 还有 a 职能 *g* 其中 地图 词汇-命令 为了 的 新的 增加 后缀 (即。， 的 最长的 后缀) 去 那些 在T*it j*。 这两个函数计算所有所需的lex-order，然后用于 建筑 的 需要的 Ψ功能 T*it j*.

*订单(SUFk，Tit j*每1≤k≤1。



首先，下面的引理显示了与SA相关的函数f*−1*和

*tt*

1

萨*−*

*titt*

[m*j*在那里m和m*j* 是T中的位置 *j* 和T*it j* 相当于

相同的T后缀 *j* (即M*j* =m+|T *i*|).

回想一下那个suf*k* 表示字符串c*kc*k+1 ···c*at j*.

**引理7。** *假设SA−1*[m]=j。 *然后是SA−1* [m*j]等于*

*tt titt*

*小的地方*

f(j)=j+#(订单*k，Tj*)≤j)，

*表示订单的数量 ，Tj)是的*

*(sufk*



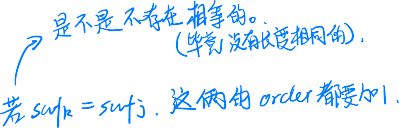
*呃大于或等于j，因为*

#(订单*k，Tj*)≤j)

1≤k≤A。

*观察1f对j∈严格增加[1，|T j*|].



**引理8。** *在O(N)时间内，f可以存储在O(N)位中，从而可以在恒定时间内计算每个f(J。*



函数g，它计算剩余后缀的lex-order

*tit j*, 是 显示 在 的 下一个 引理。

**引理9。** *所有j∈[1，A]，SA−1*

*I tt t*

*[j]等于*

*g(j)=订单(sufj，Tj*)+#(苏菲*k ≤苏夫j*),



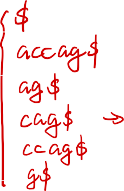
*在哪里#(苏菲k ≤苏夫j)表示suf的数目k 那是斯迈勒 或等于sufj所有1≤k≤A。*

*是的*

**观察2每j，k，如果suf***j <苏夫k然后g(J)<g(K)。 换句话说，*

*在suf的上升lex-阶中，g(K)严格增加k.*

**引理10。** *观测值1和2中严格增加的序列f和g的值是不同的，它们跨越[1，|T it j*|].

基于引理7、9和10，我们准备描述如何完成(C)部分)。 让Φ=Ψ*t t*  还有 φ*j*  =ψ*t itt.*



## 引理11。

*1.* φ*j[f(j)]=f(Φ[j])为j=1、2，。。。，T j .*

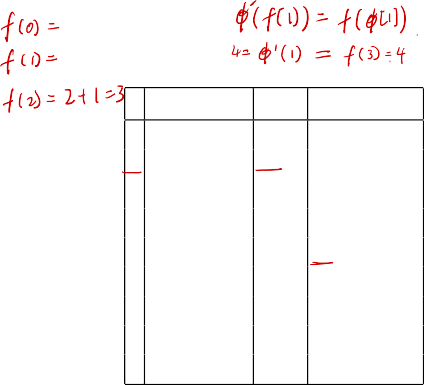
| |

*2.* φ*j*[g(A)]=f（Φ[0]）。



*3.* φ*j[g(j)]=g(j+1)，j=1，2，。。。，−1。*

有关函数Φ和Φ的示例，请参见图2和图3*j*以及它们与f和g的关系。



I SA*t* 1 *tΦj*T萨*t* 1 *t t* [一]

0 8

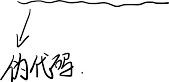
8 $



|  |  |  |  |
| --- | --- | --- | --- |
| *i* | 萨*t t* [一] | Φ[i] | *t*s*j* a*t t* [一] |
| 0 | 5 | 4 | $ |
| 1  2 | 3  1 | 3  5 | *交流$*  *阿加克$* |
| 3 | 4 | 0 | *c$* |
| 4 | 0 | 2 | *卡加奇$* |
| 5 | 2 | 1 | *加科$* |

|  |  |  |
| --- | --- | --- |
| 1 6 | 4 | *交流$* |
| 2 2 | 6 | *Acagac$* |
| 3 4 | 7 | *阿加克$* |
| 4 7 | 0 | *c$* |
| 5 1 | 2 | *仙人掌$* |
| 6 3 | 3 | *卡加奇$* |
| 7 5 | 1 | *加科$* |
| 8 0 | 5 | *$* |

**无花果。 2.** 功能Φ和Φ*j*。 Φ和Φ*j*  表示文本T的Ψ函数 *j* =Cagac$和文本T 1*t j 分别=gcacagac$。*

计算所需Φ的算法*j*  如下所示。 通过引理10，将递增序列f和g合并，得到的值从1到 t*it j* . 所以 我们合并f和g，在此期间计算Φ*j* 引理11顺序的值。 a 伪码 的 的 算法 是 显示 在 图 4.

| |

因此，我们有以下引理。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *j* | *f(j)* | Φ[j] | f(Φ[j]) | *j* | *g(j)* | g(j+1) |
| 0 |  | 4 |  |  |  |  |
| 1 | 1 | 3 | 4 |  |  |  |
|  |  |  |  | 3 | 2 | (6) |
| 2 | 3 | 5 | 7 |  |  |  |
| 3 | 4 | 0 | 0 |  |  |  |
|  |  |  |  | 2 | 5 | 2 |
| 4 | 6 | 2 | 3 |  |  |  |
| 5 | 7 | 1 | 1 |  |  |  |
|  |  |  |  | 1 | 8 | 5 |

**无花果。 3.** 函数f和g。 f(Φ[j])和g(j+1)值（下划线）形成Φ*j*  当我们合并f和g时。 注意，（6）表示设置Φ的特殊情况*j*当j=A时，[g(j)]至f(Φ[0]。

**对于t=0，1，...，|T** *it j|*

*jf ← 1,*

*jg ← 1.*

**如果t=g(A)**

**否则，如果t=f(j***f* )

*φj*[t]←f（Φ[0]）；

**其他** *φj*[t]←g(j*g* + 1), *jg* ++;

*φj*[t]←f(Φ[j*f* ])， j*f* ++;

**无花果。 4.** 伪码来构造Φ*j* 按顺序排列。

**引理12。** *假设我们得到了 苏菲j 在集合中k 1k 和顺序(sufj，Tj)所有1j 答，以及T的Ψ函数 j。 则Ψ函数为Tit j 可以在(H)中构造*0 *+2+O（1）n位在O(N)时间内。*

{ | ≤ ≤ } ≤ ≤

总之，我们有：

**引理13。** *已知Ti 以及T的Ψ函数j。 的Ψ函数计算*

*titj 需要O(日志n+n)时间。*

## 总体业绩

结合第3.1和3.2节的结果，我们总结了这一节的结果如下：

**定理1***给定长度为n的字符串T，可以在(H)中的O(nlogn)时间内计算T的Ψ函数*0 *+2+s)n位空间，对于任何s>0。*

# 空间效率建设的FM印度

除了CSA之外，还有另一个用于后缀数组的压缩索引，称为FMindex[5]，它在显示竞争的同时，在大小上显示了它的紧凑性-





最近[4]的搜索模式的性能。 该索引特别适合小字母的文本。 构造算法的核心部分涉及Burrows-Wheeler变换[1]，它是bzip2[23]等各种数据压缩算法中常用的过程。

准确地说，Burrows-Wheeler变换将长度为n的文本T转换为另一个文本W，其中W被证明是可压缩的，根据T[18]的经验熵。 转换后的文本W被定义为，如果SA[i]>0，则W[i]=T[SA[i]1]；如果SA[i]=0，则W[i]$。

−

考虑到T的Ψ，我们观察到对于任何p，SA[Ψ*k*[p]=SA[p]+k[22]。 现在，通过设置p=Ψ[0]SA*−1*[0]和计算Ψ*k*对k个=迭代

我们得到了SA[Ψ的值*k*. 立即=k，我们可以设置W[Ψ*k*[p]=T[k1]。 由于Ψ的每次计算都需要O（1）时间，所以W可以在O(N)时间内构造。

−

因此，我们有以下定理。

**定理2***给定文本T和T的Ψ函数，T上的Burrows-Wheeler变换可以直接在O(nlogΣ)位空间和O(n)时间中输出。*

| |

一旦完成Burrows-Wheeler转换，除了输出索引之外，还可以使用可忽略的空间在O(N)时间内完成构建FM-index的其余步骤。 因此，我们有以下结果：

**定理3***给定文本T和T的Ψ函数，除了输出索引外，还可以使用O(nlogΣ)位，在O(n)时间内构造T的FM索引。*

| |

# 结束语

我们提出了一种在(H)中构造CSA的算法0 +2+s)n位空间，对于一些固定的s>0。 运行时间为O(nlogn)，与字母大小无关。 相反，使用可比较空间的最快已知算法需要O(Σnlogn) 时间到了。

| |

我们还给出了第一种使用有限工作空间构造FM索引的算法。 我们表明，一旦构建了CSA，我们就可以在O(N)时间内从CSA构建FMindex，而无需额外的工作存储。

在文献中，有一种算法在O(n日志Σ)时间内构造CSA，但工作空间大大增加到O(n日志Σ)位[11]。 一个有趣的开放问题是：我们能否在o(nlogn)时间内构造CSA，而只使用O(H0 +1)n)位工作空间？

| |

| |

# 参考资料

* 1. M.布伦斯和D。j.惠勒。 一种块排序无损数据压缩算法。 技术报告124，数字设备公司，保罗阿尔托，加利福尼亚州，1994年。
  2. d. r。 克拉克和杰。 I.。 曼罗。 二级存储的高效后缀树。 *在检察院。 ACM-SIAM SODA，第383-391页，1996年。*
  3. m.法拉奇。 最优后缀树建设与大阿尔法贝。 在检察院。 IEEE FOCS，第137-143页，1997年。
  4. p.费拉吉纳和曼齐尼。 机会主义指数的实验研究。 *在检察院。* ACM-SIAM SODA，第269-278页，2001年。
  5. p. 费拉金和G。 曼齐尼。 机会数据结构与应用。

在检察院。 IEEE FOCS，第390-398页，2000年。

* 1. d. 格拉夫和K。 陈。 中国Gigaword，2003年。 http://

[//www.ldc.upenn.edu/Catalog/Catalog Entry.jsp？目录ID](http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId) =ldc2003t09。

* 1. R.格罗西和J。S.威特。 压缩后缀数组和后缀树，应用于文本索引和字符串匹配。 在检察院。 ACM STOC，第397-406页，2000年。
  2. R.格罗西和J。S.威特。 压缩后缀数组和后缀树与Appli-

文本索引和字符串匹配的阳离子。 手稿，2001年。

* 1. d. a。 格罗斯曼和奥。 弗里德。 信息检索：算法和启发式。 Kluwer学术出版社，波士顿，1998年。
  2. D.格斯菲尔德。 *字符串、树和序列的算法：计算机科学和*

计算生物学。 剑桥大学出版社，纽约，1997年。

* 1. W.K.Hon，K.Sadakane和W.K.Sung。 在构造全量指数时打破时空障碍。 在检察院。 *IEEEfocs，2003。*出现。
  2. e. 亨特，M。 p。 阿特金森和R。 w。 欧文。 大生物的数据库索引

序列。 在检察院。 VLDB，第410-421页，2000年。

* 1. p.Ko和S.Aluru。 空间有效线性时间构造的后缀阵列。 在

Proc。 CPM，第200-210页，2003年。

* 1. 库尔兹S.。 降低后缀树的空间需求。 软件实践与经验，29：1149-1171，1999年。
  2. 林文T.、沙达康、宋文强、柳文明。 一种构造压缩后缀阵列的时空高效算法。 在检察院。 COCOON，第401-410页，2002年。
  3. j. 拉尔森和K。 萨达坎。 更快的后缀排序。 技术报告技术报告LU-CS-TR：99-214，LUNDFD6/(NFCS-3140)/1-43/（1999），隆德大学，1999年。
  4. U.曼伯和麦尔斯。 后缀数组：在线字符串的一种新方法

搜索。 *SIAM计算杂志，22（5）：935-948，1993。*

* 1. g. 曼齐尼。 Burrows-Wheeler变换分析。 ACM杂志，48（3）：407-430，2001年。
  2. e. m。 麦克·克莱特。 空间经济后缀树构造算法。 *期刊*

ACM，23（2）：262-272，1976年。

* 1. T.和陈海光。 基于互信息的中文关键词提取可更新的PAT-Tree方法：知识管理的语言基础。 1999年亚洲数字图书馆会议记录。
  2. R.米。 一些实用的通用无噪音编码技术。 技术报告

JPL-79-22，喷气推进实验室，帕萨迪纳，加利福尼亚州，1979年。

* 1. K.萨达坎。 压缩后缀数组的新文本索引功能。

*算法杂志，出版社。*

* 1. j.西沃德。 bzip2和libbzip2官方主页，1996年。

[http://sources.redhat.com/bzip2/。](http://sources.redhat.com/bzip2/)

* 1. Shimozono S.、H.Arimura和S.有川。 大文本数据库中最优词关联模式的有效发现。 *新一代计算，18：49-60，2000。*