arXiv:2603.17537v1 [cs.DS] 18 Mar 2026

逆 Lyndon 数组：定义、性质与线性时间构造

Pietro Negri[1], Manuel Sica[1], Rocco Zaccagnino[1], 和 Rosalba Zizza[1]

意大利菲夏诺（SA），萨莱诺大学计算机科学系 {masica, rzaccagnino, rzizza}@unisa.it，p.negri137@gmail.com

摘要。 Lyndon 数组记录词中每个位置开始的最长最大 Lyndon 子词的长度，在词组合学中发挥着重要作用，例如可用于构造后缀数组等基本数据结构。本文介绍了逆 *Lyndon* 数组，这是针对逆 Lyndon 词的对应结构；逆 Lyndon 词是指字典序大于其所有真后缀的词。与标准Lyndon词不同，逆 Lyndon 词可能具有非平凡边界，这带来了真正的理论难题。我们证明，逆 Lyndon 数组可通过下一个更大后缀数组与一个边界校正项来刻画，并证明该校正项等于一个最长公共扩展（LCE）值。基于这一刻画，我们将 Ellert 的 Lyndon 数组线性时间构造所依据的最近后缀框架推广到逆向情形，从而为一般有序字母表得到一个 $O(n)$ 时间算法。最后，我们讨论其对后缀比较的影响，并报告了在随机、结构化和真实数据集上的实验结果，表明逆向构造在实践中与标准构造一样呈现线性时间表现。

关键词： 逆 Lyndon 词 · Lyndon 数组 · 词组合学 · 词算法 · 后缀数组

1 引言

Lyndon 词由 Lyndon [1] 提出，并在词组合学中得到了广泛研究。它们是本原词，在其循环移位中按字典序最小。等价地，非空词 w 是一个*Lyndon* 词，当且仅当对于 w的每个非空真后缀 s，都有 $w \prec s$，其中 $\prec$ 表示由字母表 Σ 上的全序诱导的字典序。

词的 *Lyndon* 数组 $\lambda[1..n]$在每个位置 i给出从 i开始的最长 Lyndon 因子的长度。Bannai 等人 [8] 证明，只需 λ即可在线性时间内计算所有极大重复，因此，高效构造 λ 的算法受到了广泛关注。Franek 等人 [6,7] 将 λ与逆后缀数组上的 下一个更小值 计算联系起来，而 Ellert [13]后来给出了第一个适用于一般有序字母表的直接线性时间算法，

该算法基于最近的较小值边，以及受 Manacher 算法 [4]启发的最长公共扩展加速机制。后续研究探讨了节省空间的变体、次线性算法，以及与 Lyndon 森林 [14,16,17]的联系。

一个密切相关的概念是逆 *Lyndon* 词：非空词w 是逆 Lyndon 词，当且仅当 $s \prec w$ 对每个非空真后缀 s [11]都成立。该概念是在研究逆规范 *Lyndon* 分解 (ICFL)时引入的。ICFL 是 CFL 的一种变体，可能产生更均衡的因子，因此非常适用于生物信息学应用 [18]。 [26] 首次提出了规范逆 Lyndon 分解的边界性质，随后在 [12]中对此作了进一步阐释。从组合学的角度看，逆向情形并非只是对标准情形作形式上的对偶化。关键的不对称之处在于：标准Lyndon词没有边界，而逆 Lyndon 词可能具有非平凡边界。因此，经典恒等式 $\lambda[i] = next[i] - i$ 不再原样成立。由此，主要难点并非只是将 Ellert 框架中的不等式反向，而是要精确确定边界如何与最近较大后缀相互作用，并证明这种相互作用仍能保持底层最长公共扩展机制的均摊线性复杂度。“边界修正”具有结构性：若不将其识别为最近较大后缀边上的一个最长公共扩展值，就必须单独进行边界计算，从而无法直接线性时间归约。我们的结果还联系了两个相近的研究方向，即一般有序字母表上的非交叉最长公共扩展查询 [24,25] ，以及基于边界的逆 Lyndon 词和 ICFL 研究视角 [12]。在逆Lyndon数组的情形下，我们证明，所需的边界修正可直接通过对应 NGS 边上的最长公共扩展得到。

贡献。 我们提出了逆*Lyndon*数组 $\lambda^{-1}[1..n]$，它记录了每个位置 i 处以 i 开始的最长极大逆 Lyndon 子词的长度。本文的主要贡献如下：

– 通过下一个较大后缀数组对 λ^{-1} 进行精确刻画，并引入边界校正项（见引理 7）；– 将边界校正项等同于最近的较大后缀边上的一个最长公共扩展值，从而无需显式计算边界，并使我们的方法区别于逆 Lyndon 文献中显式处理边界的方法（见引理 8）；– 将经典 NSS（下一个更小后缀）/PSS（上一个更小后缀）框架中的非交叉性质、链迭代和最长公共扩展加速方法，推广到全新的 NGS （下一个较大后缀） /PGS （前一个较大后缀）框架，从而得到一种可在一般有序字母表上构造逆 Lyndon 数组的线性时间算法；– 研究后缀排序的影响，表明标准 Lyndon 数组所依据的兼容性在逆向情形下没有直接对应性质，而联合使用 λ 和 λ^{-1} 仍可为若干后缀比较提供常数时间规则。

实现和基准测试脚本可在线获取[1]。[1]
https://github.com/FLaTNNBio/inverselyndonarray

2 预备知识

单词与字典序。词 $x = x[1]x[2]\cdots x[n]$ 是取自全序字母表（Σ, $<$）的符号序列。我们用 $|x| = n$ 表示其长度，用 ε 表示空单词，用 Σ^* 和 Σ^+ 分别表示所有单词和所有非空单词的集合。对于 $1 \le i \le j \le n$，子串 $x[i..j]$ 是序列 $x[i]x[i+1]\cdots x[j]$；若 $i > j$，则它等于 ε。子串 $x[1..j]$ 是一个前缀，而 $x_i = x[i..n]$ 表示从位置 i开始的后缀。

两个 Σ^* 上的字典序 $\prec$ 定义如下：$x \prec y$ 当且仅当 $y = xw$ 对于某个非空 w成立，或者 $x = urv$ 且 $y = usw$ 满足 $r < s$。两个后缀的最长公共扩展为：

$$\mathrm{lce}(i,j) = \max\{|u| \mid x_i = uv,\ x_j = uw\}.$$

我们为每个词添加哨兵字符，即 $x =$ # $x(1..n)$\$，其中在标准设定下，对于所有 $a \in \Sigma$，# $<$ \$ $< a$；而在逆向情形下，对于所有$a \in \Sigma$，# $>$ \$ $> a$。在标准情形中，将#置于所有字母表符号的严格下方，可确保 x_1 是添加哨兵字符后的词的所有后缀中字典序最小的后缀，因此所有后缀秩都各不相同，并且每条NSS/PSS边都具有唯一性。在逆向情形中，反转后的顺序使 x_1 成为字典序最大的后缀，并保证所有后缀构成严格全序，不存在并列。

非空词 w 是*Lyndon* 词，当且仅当 $w \prec s$ 对 w [2]的每个非空真后缀 s 都成立。子串 $x[i..j]$是从 i 开始的 x 的最大 *Lyndon* 子词，当且仅当它是 Lyndon 词，且 $j = n$ 或 $x[i..j+1]$ 不是 Lyndon 词。

定义1（Lyndon 数组）。词 $x[1..n]$ 的 Lyndon 数组 $\lambda[1..n]$ 定义如下：

$$\lambda[i] = \max\{m \in [1, n-i+1] \mid x[i..i+m-1]\ \textit{is a Lyndon word}\}.$$

后缀数组 $SA[1..n]$ 是满足以下条件的排列：

$$x_{SA[1]} \prec x_{SA[2]} \prec \cdots \prec x_{SA[n]}.$$

逆后缀数组 ISA 满足$ISA[SA[i]] = i$，因此 $ISA[j]$ 是 x_j的字典序排名。Franek 等人 [6] 证明了以下刻画。

定理1（[6]）。 $x[i..j]$ 是最大 *Lyndon* 子词，当且仅当：

1. 对所有 $h \in [i+1, j]$，*ISA*$[i] <$ *ISA*$[h]$，且
2. $j = n$ 或 *ISA*$[j+1] <$*ISA*$[i]$。

因此，λ 可以通过对逆后缀数组进行 NSV计算 得到，具体如下。给定数组 $A[1..n]$，NSV数组 定义如下：

$$\mathrm{NSV}[i] = \min\{j > i \mid A[j] < A[i]\} - i,$$

若不存在这样的 j，则使用 $\mathrm{NSV}[i] = n - i + 1$。

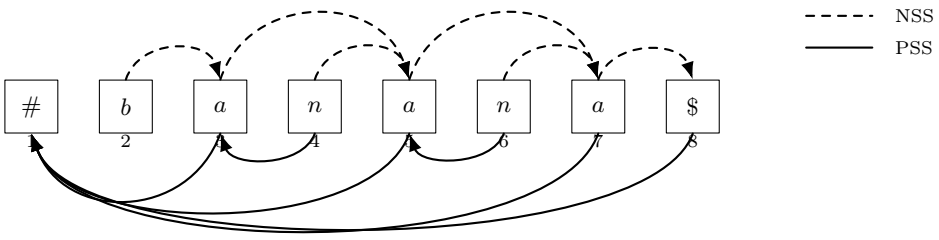

图1. $x = \#banana\$$ 的下一个更小后缀和上一个更小后缀边。虚线弧表示下一个更小后缀边，实线弧表示上一个更小后缀边。

2.1 下一个更小后缀/上一个更小后缀框架

我们简要回顾 Ellert 的框架 [13]，该框架用于在 $O(n)$ 时间内计算 λ，因为它是我们进行逆向构造的模板。本节中，我们始终假设$x = \#\,x(1..n)\,\$$，且对所有 $a \in \Sigma$ 均有 $\# < \$ < a$。

定义2（下一个更小后缀和上一个更小后缀数组）。对于一个词 $x[1..n]$，定义

$$next[i] = \min\{j \in (i, n] \mid x_j \prec x_i\},$$
$$prev[i] = \max\{j \in [1, i) \mid x_j \prec x_i\},$$

$next[1] = next[n] = n{+}1$、$prev[1] = 0$ 和 $prev[n] = 1$。

例*1.* 对于 $x = \#banana\$$，其中 $\# < \$ < a < b < n$，数组为:

i	1	2	3	4	5	6	7	8
x	#	b	a	n	a	n	a	\$
$next[i]$	9	3	5	5	7	7	8	9
$prev[i]$	0	1	1	3	1	5	1	1
$\lambda[i]$	8	1	2	1	2	1	1	1

图1展示了相应的下一个更小后缀边和上一个更小后缀边。

引理1（[13]）。对于所有 $i \in [1, n]$，下一个$[i] = i + \lambda[i]$。

引理 2（[13]）。对于所有 $i \in [2, n]$，词 $x[prev[i]..i{-}1]$ 是一个 *Lyndon* 词。

实现线性时间构造的关键性质如下。

引理 3（非交叉 [13]）。设 $l_1 < r_1$ 和 $l_2 < r_2$ 为索引对，且每一对都由下一个更小后缀边或上一个更小后缀边连接。则不可能出现 $l_1 < l_2 < r_1 < r_2$。

引理 4（链迭代 [13]）。对于任意 $r \in [2, n]$:

1. $prev[r] = prev^*[r{-}1]$，*2.* $next[l] = r$ 当且仅当 $l = prev^*[r{-}1]$ 且 $l > prev[r]$，

其中，$prev^*[r{-}1]$ 表示来自 $r{-}1$的 *PSS* 链上的一个元素。

引理 5（LCE加速 [13]）。令 $l = prev[k]$ 和 $r = next[k]$。则：

1. 若 $\mathrm{lce}(l,k) = \mathrm{lce}(k,r)$，则 $\mathrm{lce}(l,r) \geq \mathrm{lce}(k,r)$；2. 若 $\mathrm{lce}(l,k) < \mathrm{lce}(k,r)$，则 $\mathrm{lce}(l,r) = \mathrm{lce}(l,k)$，且 $\mathrm{prev}[r] = l$；3. 若 $\mathrm{lce}(l,k) > \mathrm{lce}(k,r)$，则 $\mathrm{lce}(l,r) = \mathrm{lce}(k,r)$，且 $\mathrm{next}[l] = r$。

结合 [13], 的 SMART-LCE 过程，该过程通过全局最右检查位置变量，在摊销 $O(n)$ 时间内计算所有所需的最长公共扩展值，这些引理给出了一个算法，可在一般有序字母表上以 $O(n)$ 时间和 $n()$ 空间计算 λ。

3 逆 Lyndon 数组

现在我们来建立逆 Lyndon 数组的理论。本节中始终有，$x = \#x(1..n)\,\$$，其中哨兵字符满足对所有 $a \in \Sigma$，$\# > \$ > a$。

3.1 逆Lyndon词

定义 3 ([11])。 </style}词 是一个逆 Lyndon 词，当且仅当 $s \prec w$ 对于 w的每个非空真后缀 s，均满足此条件。

与标准Lyndon词不同，逆 Lyndon 词可以有非平凡边界。例如，$dabda$ 是一个逆 Lyndon 词，其最大边界为da。以下前缀封闭性质至关重要 [11]。

引理 6 ([11])。 逆 *Lyndon* 词的每个非空前缀都是逆*Lyndon*词。

定义 4（极大逆Lyndon子词）。子词 $x[i..j]$是一个极大逆 Lyndon 子词，起始位置为 i，当且仅当它是逆 *Lyndon* 词，且 $j = n$ 或 $x[i..j+1]$ 不是逆 *Lyndon* 词。

定义 5（逆 Lyndon 数组）。逆*Lyndon*数组定义如下：

$$\lambda^{-1}[i] = \max\{m \in [1, n-i+1] \mid x[i..i+m-1] \text{ is an inverse Lyndon word}\}.$$

示例 *2.* 设 $x = \#aababbaa\$$，其中 $\# > \$ > b > a$。逆 Lyndon 数组及其对应的 NGS/PGS 数组为：

i	1	2	3	4	5	6	7	8	9	10
x	#	a	a	b	a	b	b	a	a	\$
λ^{-1}	10	2	1	3	1	4	3	2	1	1
$next_{-1}$	11	3	4	6	6	10	10	9	10	11
$prev_{-1}$	0	1	1	1	4	1	6	7	7	1
$nlce$	0	1	0	1	0	0	0	1	0	0

在 $i = 4$ 处，极大逆Lyndon子词为 $x[4..6] = bab$，因此 $\lambda^{-1}[4] = 3$。其最长边界的长度为 1，因此 $next_{-1}[4] = 6$ 和 $nlce[4] = 1$，与推论 1 一致。在 $i = 6$ 处，极大逆Lyndon子词为 $x[6..9] = bbaa$，因此 $\lambda^{-1}[6] = 4$。由于这个因子没有边界，$next_{-1}[6] = 10$ 和 $nlce[6] = 0$。

3.2 NGS/PGS 数组与它们和 λ^{-1} 的关系

定义 6（NGS 和 PGS 数组）。对于带有反转哨兵字符的词 $x[1..n]$，定义：

$$next_{-1}[i] = \min\{j \in (i, n] \mid x_j \succ x_i\},$$
$$prev_{-1}[i] = \max\{j \in [1, i) \mid x_j \succ x_i\},$$

满足 $next_{-1}[1] = next_{-1}[n] = n{+}1$、$prev_{-1}[1] = 0$ 和 $prev_{-1}[n] = 1$。

因此，在逆向情形中，相关边连接的是最近的较大后缀，而非最近的较小后缀。

引理 7（NGS 与 λ^{-1} 等价性）。设 z 是从 i 开始的极大逆*Lyndon*子词，且 $\mathrm{border}(z)$ 表示其最长真边界的长度。那么：

$$next_{-1}[i] = i + \lambda^{-1}[i] - \mathrm{border}(z).$$

证明。写作 $z = bhb$，其中 $b \in \Sigma^*$ 是边界（可以为空），且 $h \in \Sigma^+$。那么，对于某个 $\alpha \in \Sigma^+$，有 $x_i = bhb\alpha$。令 $j = next_{-1}[i]$。我们断言 $x_j = b\alpha$，也就是说，后缀 x_j 从边界第二次出现的位置开始。

首先，假设 x_j 在该次出现之前开始，设为 $x_j = \beta b\alpha$，其中有 $\beta \in \Sigma^+$ 和 $z = b\gamma\beta b$。那么 βb 是 z 的真后缀，因此根据定义 3，我们有 $z \succ \beta b$。于是 $z\alpha \succ \beta b\alpha$，也就是说，$x_i \succ x_j$，这与 j 的选择矛盾。

反之，假设 x_j 在边界之后开始，设为 $x_j = \beta\alpha$，其中有 $b = \gamma\beta$ 和γ，$\beta \in \Sigma^+$。考虑中间后缀 $x_{k_m} = \beta h\gamma\beta\alpha$。由于 $j =$下一个$_{-1}[i]$，我们有 $x_{k_m} \prec x_i \prec x_j$。然而，由 $x_j = \beta\alpha \succ \beta h\gamma\beta\alpha = x_{k_m}$可得$b\alpha = \gamma\beta\alpha \succ \gamma\beta h\gamma\beta\alpha = x_i$，这意味着 $x_{k_m} \succ x_i$，再次产生矛盾。

因此 $x_j = b\alpha$，所以

$$j = i + |z| - |b| = i + \lambda^{-1}[i] - \mathrm{border}(z).$$

边界项是与标准情形的主要结构差异。标准 Lyndon 词没有边界，因此相应的修正项始终为零。

引理 8（通过最长公共扩展求边界）。设 z 是从 i 开始的极大逆*Lyndon*子词，且 $j =$紧接着$_{-1}[i]$。则

$$\mathrm{lce}(i, j) = \mathrm{border}(z).$$

证明。沿用上述记号，由引理 7 可得 $x_j = b\alpha$。由于 $x_i = bhb\alpha$ 和$x_j = b\alpha$，后缀 x_i 和 x_j 在前缀 b 上一致，因此 $\mathrm{lce}(i, j) \geq |b|$。

对于上界，注意到 $x_j \succ x_i$ 因为 $j =$下一个$_{-1}[i]$。这两个后缀有一个长度为 $|b|$ 的公共前缀（两者都以 b开头）；在下一个位置$x_i[|b|{+}1] = h[1]$ 而 $x_j[|b|{+}1] = \alpha[1]$。由于 $x_j \succ x_i$ 且它们的前$|b|$ 个位置相同，因此比较结果由位置 $|b|{+}1$ 决定，这迫使 $\alpha[1] > h[1]$。特别是 $\alpha[1] \neq h[1]$，所以公共前缀恰好在长度 $|b|$ 处结束，并且 $\mathrm{lce}(i, j) = |b| =$ 边界(z)。

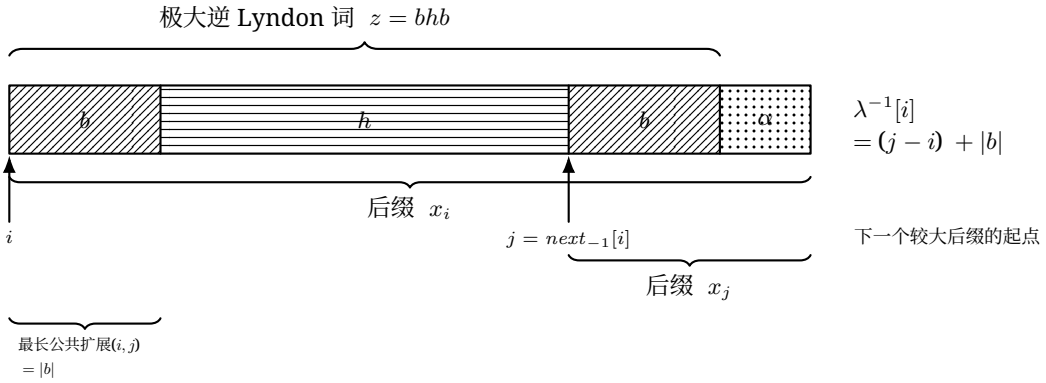

图 2。 逆 Lyndon 数组的边界修正。如果从位置 i 开始的极大逆 Lyndon 词具有 $z = bhb$ 的形式，那么下一个较大后缀从 b 的第二次出现处开始。因此，$\mathrm{lce}(i,\ j) = |b|$ 且 $\lambda^{-1}[i] = (j - i) + |b|$。

推论 1. 对于每个 i,

$$\lambda^{-1}[i] = next_{-1}[i] - i + \mathrm{lce}(i, next_{-1}[i]).$$

注释 *1*（为什么边界项具有结构性）。 推论 1 中的修正项并非后处理细节。没有引理 8，逆数组就需要为每个极大逆 Lyndon 因子显式保存边界信息。这将引入标准情形中不存在的额外计算层。引理 8 的意义恰恰在于，边界已经由算法计算出的同一最近后缀最长公共扩展信息编码。

因此，计算 $next_{-1}$ 以及相关的$nlce$ 数组，就足以恢复 λ^{-1}。

引理 9。 对于所有 $i \in [2, n]$, $x[prev_{-1}[i]..i-1]$ 都是逆 *Lyndon* 词。

证明。 设 $p = prev_{-1}[i]$ 和 $w = x[p..i-1]$。根据 PGS 的定义，对于每个 $k \in (p,\ i)$，都有 $x_p \succ x_i$ 和 $x_k \prec x_i$。由传递性，对于每个 $k \in (p,\ i)$，都有 $x_p \succ x_k$。

设 $s = x[k..i-1]$ 是 w 的任意非空真后缀，其中 $k \in (p,\ i)$。由于 $x_p \succ x_k$，设 $m \geq 1$ 为 x_p 和 x_k 不同的第一个位置；于是 $x[p+m-1] > x[k+m-1]$。如果 $m \leq i-k = |s|$，则不匹配发生在 w 和 s 中，因此 $w \succ s$。如果 $m > i-k$，那么 $s = x[k..i-1]$ 是 w 的真前缀（它们在 s 的全部 $|s|$ 个位置上相同），所以根据字典序的前缀规则，$w \succ s$。在这两种情况下，$w \succ s$，因此 w 是一个逆 Lyndon 词。

3.3 NGS/PGS 边的组合性质

现在我们证明，NSS/PSS 情形中的主要结构引理在逆情形下仍然成立。这些引理也属于一般有序字母表上基于最长公共扩展的非交叉技术这一更广泛的范畴。不过，在此处的逆 Lyndon 情形中，非交叉结构具体由 NGS/PGS 边诱导而成。

引理 10（NGS/PGS 边的不交叉性）。设 $l_1 < r_1$ 和 $l_2 < r_2$ 是索引对，且各自通过一条 *NGS* 或 *PGS* 边相连。那么，不可能出现 $l_1 < l_2 < r_1 < r_2$。

证明。反设 $l_1 < l_2 < r_1 < r_2$。由于 $l_2 \in (l_1, r_1)$ 和 l_1, r_1 由一条下一个较大后缀边或 PGS 边相连，因此所有起始位置在 (l_1, r_1) 内的后缀，其字典序都小于两个端点，故 $x_{l_2} \prec x_{r_1}$。另一方面，由于 $r_1 \in (l_2, r_2)$ 和 l_2, r_2 相连，我们得到 $x_{l_2} \succ x_{r_1}$。矛盾，由此证明该结论。

引理 11（NGS/PGS 链迭代）。对任意 $r \in [2, n]$:

1. $prev_{-1}[r] = prev^*_{-1}[r-1]$;
2. $next_{-1}[l] = r$ *if and only if* $l = prev^*_{-1}[r-1]$ *and* $l > prev_{-1}[r]$.

证明。对于第一个结论，设 $prev_{-1}[r] \neq prev^*_{-1}[r-1]$。那么存在 $r' = prev^*_{-1}[r-1]$，使得 $prev_{-1}[r] \in (prev_{-1}[r'], r')$，于是得到

$$prev_{-1}[r'] < prev_{-1}[r] < r' < r,$$

这与引理 10 矛盾。

对于第二个陈述，先假设 $next_{-1}[l] = r$。那么，起始位置位于（l，r）的所有后缀都小于 x_r，因此 $prev_{-1}[r] \notin [l, r)$，从而 $prev_{-1}[r] < l$。如果 $l \neq prev^*_{-1}[r-1]$，则存在 $r' = prev^*_{-1}[r-1]$，使得

$$prev_{-1}[r'] < l < r' < r,$$

再次与引理 10 矛盾。

反之，假设 $prev_{-1}[r] < l$ 且 $l = prev^*_{-1}[r-1]$。由于 $l \in (prev_{-1}[r],\ r)$，我们有 $x_l \prec x_r$。此外，链性质意味着，对于每个 $k \in (l,\ r)$，都有 $x_k \prec x_l$。因此，r 是 l 右侧第一个具有更大后缀的位置，即 $next_{-1}[l] = r$。

引理 12（NGS/PGS 的 LCE 加速）。令 $l = prev_{-1}[k]$ 和 $r = next_{-1}[k]$。则：

1. 如果 $\mathsf{lce}(l, k) = \mathsf{lce}(k, r)$，那么 $\mathsf{lce}(l, r) \geq \mathsf{lce}(k, r)$，且 $\mathsf{prev}_{-1}[r] = l$ 或 $\mathsf{next}_{-1}[l] = r$；2. 如果 $\mathsf{lce}(l, k) < \mathsf{lce}(k, r)$，那么 $\mathsf{lce}(l, r) = \mathsf{lce}(l, k)$，且 $\mathsf{prev}_{-1}[r] = l$;
3. 如果 $\mathsf{lce}(l, k) > \mathsf{lce}(k, r)$，那么 $\mathsf{lce}(l, r) = \mathsf{lce}(k, r)$，且 $\mathsf{next}_{-1}[l] = r$。

证明。对于第一种情况，由于 $l =$ 前驱$_{-1}[k]$ 且 $r =$ 后继$_{-1}[k]$，我们有 $x_l \succ x_k \succ x_m$，并且对所有 $m \in (l,\ r) \setminus \{k\}$，都有 $x_r \succ x_k \succ x_m$。由于 $\mathsf{lce}(l,\ k) = \mathsf{lce}(k,\ r)$，因此 $\mathsf{lce}(l,\ r) \geq \mathsf{lce}(k,\ r)$。如果 $x_l \succ x_r$，那么对所有此类 m，都有 $x_l \succ x_r \succ x_m$，因此前驱$_{-1}[r] = l$。由对称性，如果 $x_r \succ x_l$，那么 后继$_{-1}[l] = r$。

对于第二种情况，令 $c = \mathsf{lce}(l,\ k)$。由于 $x_l \succ x_k$，我们有 $x[l + c] > x[k + c]$。因为 $\mathsf{lce}(l,\ k) < \mathsf{lce}(k,\ r)$，后缀 x_r 与 x_k 共享一个长度至少为 $c + 1$ 的前缀，因此 $x[r + c] = x[k + c]$。于是 x_l 和 x_r 在 c 个位置上相同，并在下一个位置处以 $x[l + c] > x[r + c]$ 不同，所以 $x_l \succ x_r$ 和 $\mathsf{lce}(l,\ r) = c$。由于对于所有 $m \in (l,\ r)$，$x_r \succ x_m$，可得 前一个$_{-1}[r] = l$。

第三种情况是对称的，因此得到 $x_r \succ x_l$，$\mathsf{lce}(l,\ r) = \mathsf{lce}(k,\ r)$，以及下一个$_{-1}[l] = r$。

输入：逆哨兵模式下的词 $x[1..n]$ **输出：** $next_{-1}$, $prev_{-1}$, $nlce$, $plce$, $\lambda^{-1}$$prev_{-1}[1] \leftarrow 0$**；** $next_{-1}[n] \leftarrow n+1$**；** $plce[1] \leftarrow 0$**对于** $r \leftarrow 2$ **到** n **执行**$\ell \leftarrow r-1$**；** $m \leftarrow$ **Smart-Lce(**ℓ, r**,0) 当** $x[\ell+m] < x[r+m]$ **执行**$next_{-1}[\ell] \leftarrow r$**；** $nlce[\ell] \leftarrow m$**如果** $m = plce[\ell]$ **则**$m \leftarrow$ **Smart-Lce(**$prev_{-1}[\ell], r, m$**) 结束 否则 如果** $m > plce[\ell]$ **则** $m \leftarrow plce[\ell]$**结束** $\ell \leftarrow prev_{-1}[\ell]$**结束** $prev_{-1}[r] \leftarrow \ell$**；** $plce[r] \leftarrow m$**结束 对于** $i \leftarrow 1$ **到** n **执行** $\lambda^{-1}[i] \leftarrow next_{-1}[i] - i + nlce[i]$**结束 算法 1：用于逆 Lyndon 数组的 LCE-NGS**

3.4 LCE-NGS 算法

引理 11 和 12 得出算法 1，该算法沿用了标准 LCE-NSS 算法 [13]的结构。关键区别在于第 8 行比较的方向：在逆向情形中，我们检验 $x[\ell+m] < x[r+m]$, 因为我们要寻找右侧第一个字典序更大的后缀。

过程 Smart-Lce 与标准设定 [13], 中使用的摊还原语相同，但在此处仅针对 NGS/PGS 候选项进行查询。为使内容自足，我们重述定理 2 证明中用到的两个不变量。首先，每次显式扫描后，rhs 都是迄今检查过的最右文本位置。其次，只要某个边值已确定，$nlce$ 或$plce$ 中对应存储的量就恰好是该边所需的最长公共扩展。因此，如果查询 (ℓ, r, m) 满足 $r+m < rhs$，那么答案完全位于先前已认证的窗口内，可以根据引理 12 选取的边信息，在 O(1)时间内返回。否则，Smart-Lce 会显式扩展、增大 rhs，并存储由此得到的最长公共扩展值。只有在此处才会进行新的字符比较。

定理 2。 *算法 1 在一般有序字母表上以* $\lambda^{-1}[1..n]$ *in* $O(n)$ *时间和* $O(n)$ *空间计算逆 Lyndon 数组。*

证明。 正确性由引理 7、8、10、11 和 12 得出。还需说明的一点是，边界修正不会造成额外的重新扫描。

根据引理 8，每当$next_{-1}[\ell] = r$ 确定后，同一个值$nlce[\ell] =$ lce(ℓ, r)既是算法所需的边缘最长公共扩展，也是之后针对 $\lambda^{-1}[\ell]$所需的边界修正。因此，算法从不显式计算边界，也不会为了获取边界信息而单独进行一遍扫描。

此外，如果调用 Smart-Lce 时传入 $r + m < rhs$，那么查询完全落在先前显式扫描已确认的区域内。在这种情况下，引理 12 和已存储的$nlce$ 或$plce$ 值足以回答查询，无需重新扫描该窗口内的字符。因此，边界不会在此处引发隐性的重复扫描。每次显式字符比较都会严格推进全局前沿rhs；该前沿单调递增，且不会超过 n。因此，显式比较的总次数为 $O(n)$。

外层循环执行 $n - 1$ 次。每个索引的最终 $next_{-1}$ 值只赋值一次，每个已存储的 LCE 值只写入一次，最终恢复公式

$$\lambda^{-1}[i] = next_{-1}[i] - i + nlce[i]$$

对每个位置应用一次。因此，总运行时间为 $O(n)$，空间用量为 $O(n)$。

4 实验评估

我们用 C++17 实现了适用于 λ 的 LCE-NSS，以及适用于 λ^{-1} 的 LCE-NGS。本文报告的计时结果由基准测试流程在 g++-std=c++17 -O2 下生成。我们测量了实际构建时间，并在相关情况下单独报告恢复阶段的耗时。所有实验均在运行 Ubuntu Linux 的 Dell Precision 7960 Tower 上进行，配备 Intel Xeon w5-3425 CPU 和 128GB ECC 内存。我们通过在小型和中型输入以及精心构造的边缘案例上与蛮力法进行比较来验证正确性。

我们使用了字母表大小为 $\sigma \in \{2, 4, 26\}$ 的随机字符串、结构化合成数据族，以及来自 Pizza&Chili 和 Large Canterbury Corpus [21,22,23] 的真实文本。启用性能分析的实现还统计了显式字符比较次数、经认证的 Smart-Lce 窗口内 LCE 的复用次数，以及显式扩展调用次数。

表 1 报告了随机输入的运行时间。在 $n \geq 5 \times 10^4$ 的实例上，LCE-NGS/LCE-NSS 的平均比率为 1.0060，中位数比率为 0.9990，观测范围为 $[0.9882, 1.0939]$，这表明在所有字母表大小下，逆向构造与标准构造的表现都非常接近。

表 2 报告了在真实文件上取得的结果。在五个语料库中，平均比率为 1.0019，中位数比率为 1.0024，观测范围为 $[0.9665, 1.0325]$。结构化合成数据族表现出相同的实际特性。对于所有满足 $n \geq 5 \times 10^4$ 的结构化实例，平均比率为 0.9852，中位数比率为 0.9981，观测范围为 $[0.8848, 1.0874]$。因此，在随机、结构化和真实输入上，逆向内核的表现始终与标准内核接近。

为突出边界校正路径的表现，我们还对长边界数据族进行了剖析，同时记录 Smart-Lce 中的显式字符比较次数、最长公共扩展复用命中次数，以及显式扩展调用次数。每个实例都包含一个重复的前后缀边界，其长度占全文的 25% 或 40%。这些边界密集型输入上的运行时间比仍接近 1，平均值为 1.0013，中位数为 1.0031，范围为 $[0.9680, 1.0257]$。表 3 显示，所有计数仍呈线性增长

表 1。随机字符串上的核心构建时间（μ秒）。

	$\sigma = 2$		$\sigma = 4$		$\sigma = 26$	
n	LCE-NSS	LCE-NGS	LCE-NSS	LCE-NGS	LCE-NSS	LCE-NGS
10^3	15.4	15.0	15.8	15.4	15.6	16.0
5×10^3	79.6	84.2	95.6	101.2	94.0	92.6
10^4	171.4	179.0	205.4	207.0	194.0	190.6
5×10^4	948.2	1 037.2	1 062.0	1 059.2	1 006.6	1 000.6
10^5	1 974.2	1 973.4	2 187.0	2 162.4	2 060.6	2 053.8
5×10^5	9 197.8	9 416.2	10 503.2	10 543.2	10 087.8	9 981.4
10^6	18 060.0	18 320.2	20 972.4	20 955.0	20 121.8	19 919.2
2×10^6	35 778.4	36 154.0	41 893.6	41 840.6	40 219.0	39 744.4
5×10^6	89 178.6	90 033.6	102 314.0	104 479.2	100 569.0	99 436.6

表 2。真实语料库上的核心构建时间（μ 秒）。

File	n	LCE-NSS	LCE-NGS	比率
english.txt	5000 000	98 850.6	98 846.6	1.0000
dna.txt	5 000 000	104 074.2	104 328.6	1.0024
bible.txt	4 047 392	75 372.2	77 819.2	1.0325
大肠杆菌	4 638 690	96 704.0	93 460.0	0.9665
world192.txt	2 473 400	46 763.6	47 144.6	1.0081

，结果见 n。尤其是，从 10^5 到 10^6, ，两类数据族中的显式比较次数和显式扩展次数都增长了约 $10\times$ ，这与定理 2 相符。深层边界确实会影响常数，但不会导致超线性重复扫描。

5 结论与后缀排序视角

我们介绍了逆 Lyndon 数组，并通过下一个更大后缀数组和边界校正项对其进行了刻画。我们还表明，NSS/PSS 框架背后的组合要素可以迁移到 NGS/PGS 设置，从而在一般有序字母表上实现线性时间构造。核心技术要点是，边界不会产生隐藏的重新扫描开销：所需的校正项已经编码在最近较大后缀边上的最长公共扩展值中，并且可以由与标准情形相同的摊还机制吸收。在实验中，逆向构造保持了与标准构造相同的实际线性时间特性。

一种自然的视角是借助标准 Lyndon 数组和逆Lyndon数组进行后缀排序。众所周知，标准 Lyndon 数组通过兼容性支持后缀排序

$$w_\lambda(i) \prec w_\lambda(j) \Longrightarrow x_i \prec x_j,$$

其中， $w_\lambda(i)$ 表示从 i [9,15] 开始的最大 Lyndon 子词。

相比之下，逆向情形不存在直接对应的类比。

表 3. 边界密集型性能分析计数器。数值为三次运行的平均值。

类别	n	NSS 比较	NGS 比较	NSS 重用	NGS 重用	NSS 扩展	NGS 扩展
25%边界	100 000	213 984	213 427	29 347	29 258	148 890	148 233
40%边界	100 000	200 510	208 599	44 719	35 317	131 483	141 882
25%边界	500 000	945 829	1 039 872	294 036	179 692	582 948	705 698
40%边界	500 000	947 176	863 646	292 529	393 730	584 150	475 202
25%边界	1000 000	2 134 164	2 134 660	293 205	293 234	1 481797	1 483 100
40%边界	1000 000	2 135 385	2 123 841	292 240	306 666	1 483741	1 468 012

命题 1（不存在直接逆兼容性）。极大逆 *Lyndon* 因子的标准兼容性不存在直接逆向类比。具体而言，以下说法并非普遍成立：

$$w_{\lambda^{-1}}(i) \prec w_{\lambda^{-1}}(j) \Longrightarrow x_i \prec x_j.$$

证明。取 $x = babacbabaa$，其中 $i = 1$ 且 $j = 6$。于是 $w_{\lambda^{-1}}(1) = baba$ 且 $w_{\lambda^{-1}}(6) = babaa$，因此 $w_{\lambda^{-1}}(1) \prec w_{\lambda^{-1}}(6)$，但 $x_1 = babacbabaa \succ x_6 = babaa$。障碍在于前缀情形：如果一个逆 Lyndon 词的形式为 $z = z'\alpha$，则可能出现 $z \succ \alpha$，而只有 $z' \succeq \alpha$，因此仅凭最大的逆因子无法确定后缀的延续。

尽管如此，逆 Lyndon 数组仍能为相关后缀对提供常数时间比较规则。

命题 2. 对于位置 $i < j$，以下每个条件都可以在常数时间内确定 x_i 和 x_j 的顺序：

1. 若 $j < i + \lambda[i]$，则 $x_i \prec x_j$；*2.* 若 $j = \mathit{next}_{-1}[i]$，等价地，若 $j = i + \lambda^{-1}[i]$ –*nlce*$[i]$，则 $x_i \prec x_j$；*3.* 若 $j < \mathit{next}_{-1}[i]$，等价地，若 $j < i + \lambda^{-1}[i]$ –*nlce*$[i]$，则 $x_i \succ x_j$；*4.* 若 $i =$*prev* $[j]$ 或 $i =$*prev*$_{-1}[j]$，则可直接根据定义边得出比较结果。

证明. 每一项都可直接由定义得出。

将本文结果与 ICFL 的其他性质结合起来，或有助于形成高效的后缀排序方法；这一方向也与 [27] 中讨论的更广泛的形式语言视角相契合。

利益披露。作者声明，没有需要披露且与本文内容相关的竞争性利益。

参考文献

1. Lyndon, R.C.：关于 Burnside问题。II。《美国数学会会刊》**78**(2)，329–332（1955 8 2 10 1090 0002 9947 1955 0067884）。https://doi.org/ . /S - - - - . Duval, J.P.：有序字母表上的词分解。《算法杂志》**4**(4)，363–381（1983 2 3 10 1016 0196 6774 83 90017）。https://doi.org/ . / - ( ) - . Manber, U., Myers, G.：后缀数组：一种在线词搜索的新方法。《SIAM计算杂志》**22**0222058 4 5 935 948 1993 10 1137 ( )，– ( )。https://doi.org/ . / . Manacher, G.K.：一种用于寻找词的最小初始回文的全新线性时间“在线”算法。《ACM会刊》**22**(3)，346–351（1975）5. Alstrup, S., Gavoille, C., Kaplan, H., Rauhe, T.：最近公共祖先：综述及一种新的分布式算法。载于：SPAA 2002论文集，第258–264页。ACM（2002 564914 6 10 1145 564870）。https://doi.org/ . / . . Franek, F., Islam, A.S.M.S., Rahman, M.S., Smyth, W.F.：计算 Lyndon 数组的算法。载于：2016年布拉格字符串学会议论文集，第172–184页（2016）7. Franek, F., Liut, M.：重新审视计算Lyndon 数组的算法。载于：2019年布拉格字符串学会议论文集，第16–28页（2019）8. Bannai, H., I, T., Inenaga, S., Nakashima, Y., Takeda, M., Tsuruta, K.：“游程”定理。《SIAM计算杂志》**46**(5)，1501–1514（2017）。https://doi.org/10.1137/ 15M1011032 9. Baier, U.：线性时间后缀排序：一种新的后缀数组构造方法。载于：CPM 2016，信息学讲义，第54卷，第23:1–23:12页。Schloss Dagstuhl（2016）。https: //doi.org/10 23 10 4230 2016 . /LIPIcs.CPM. . . Franek, F., Paracha, A., Smyth, W.F.：后缀数组与部分排序的Lyndon数组的线性等价性。载于：2017年布拉格字符串学会议论文集，第77–84页（2017）11. Bonizzoni, P., De Felice, C., Zaccagnino, R., Zizza, R.：逆 Lyndon 词及词的逆Lyndon分解。《应用数学进展》**101**，281–319（2018）。https://doi.org/10 005 12 1016 2018 08 . /j. aam. . . Bonizzoni, P., De Felice, C., Riccardi, B., Zaccagnino, R., Zizza, R.：通过边界性质揭示Lyndon分解与规范逆 Lyndon 分解之间的联系。载于：MFCS 2024，信息学讲义，第306卷，第31:1–31:14页。Schloss Dagstuhl（2024 31 13 10 4230 2024 2022, 244, 48 1）。https://doi.org/ . /LIPIcs.MFCS. . . Ellert, J.：简化的Lyndon数组。载于：ESA 信息学讲义，第卷，第页：–48:14。Schloss Dagstuhl（2022 48 14 10 4230 2022）。https://doi.org/ . /LIPIcs.ESA. . . Bille, P., Ellert, J., Fischer, J., Gørtz, I.L., Kurpicz, F., Munro, I., Rotenberg, E.：线性时间空间高效构造Lyndon 数组。CoRR abs/1911.03542（2019）15. Bertram, N., Ellert, J., Fischer, J.：Lyndon 词加速后缀排序。载于：ESA 2021，信息学讲义，第204卷，第15:1–15:13页。Schloss Dagstuhl（2021）。https://doi.org/ 10.4230/LIPIcs.ESA.2021.15 16. Bannai, H., Ellert, J.：亚线性时间构造Lyndon 数组。载于：ESA 2023，信息学讲义，第274卷，第14:1–14:16页。Schloss Dagstuhl（2023）。https://doi.org/10.4230/LIPIcs . ESA.2023.14 17. Badkobeh, G., Crochemore, M., Ellert, J., Nicaud, C.：从后向前在线构造Lyndon 森林。载于：CPM 2022，信息学讲义，第223卷，第13:1–13:23页。Schloss Dagstuhl（2022 13 18 10 4230 2022）。https://doi.org/ . /LIPIcs.CPM. . . Bonizzoni, P., 等：用于机器学习方法的测序读段基于数值Lyndon的特征嵌入。《信息科学》**607**，458–476（2022）。https: //doi.org/10.1016/j.ins.2022.06.005

19. Bauer, M.J., Cox, A.J., Rosone, G.：用于构建和反转词集合 BWT 的轻量级算法。《理论计算机科学》**483**，134–148（2013）。https://doi.org/10 002 20 1016 2012 02 . /j.tcs. . . . Fischer, J., Kurpicz, F.：拆解 DivSufSort。CoRR abs/1710.01896（2017） 21. Ferragina, P., Navarro, G.：Pizza&Chili 语料库。压缩索引的数据集和基准测试集合（2007）。https://pizzachili.dcc.uchile.cl/ 22. Arnold, R., Bell, T.：用于评估无损压缩算法的语料库。载于：数据压缩会议论文集，第 201–210 页。IEEE 计算机学会（1997） 23. 坎特伯雷语料库：语料库描述。坎特伯雷大学。https://corpus.canterbury.ac.nz/descriptions/ 24. Crochemore, M., Iliopoulos, C.S., Kociumaka, T., Kundu, R., Pissis, S.P., Ra- doszewski, J., Rytter, W., Waleń, T.：通过非交叉最长公共扩展查询，在一般字母表上近乎最优地计算游程。载于：字符串处理与信息检索（SPIRE 2016），计算机科学讲义，第 9954 卷，第 22–34 页。施普林格，尚（ 2016 10 1007 978 3 319 46049 9_3 ）。https://doi.org/ . / 25 - - - - . Ellert, J., Fischer, J.：在一般有序字母表上以线性时间计算游程。载于：第 48 届自动机、语言与编程国际研讨会（ICALP 2021），信息学讲义，第 198 卷，第 63:1–63:16 页。达格斯图尔城堡—莱布尼茨信息学中心（2021 63 26 10 4230 2021 ）。https://doi.org/ . /LIPIcs.ICALP. . . Bonizzoni, P., De Felice, C., Zaccagnino, R., Zizza, R.：逆 Lyndon 分解中后缀的最长公共前缀及其他性质。《理论计算机科学》**862**，24 034 27 41 2021 10 1016 2020 10 –（）。https://doi.org/ . /j.tcs. . . . Bonizzoni, P., De Felice, C., Pirola, Y., Rizzi, R., Zaccagnino, R., Zizza, R.：形式语言能否助力泛基因组学表示和分析多个基因组？载于：Diekert, V., Volkov, M.V.（编），语言理论发展，DLT 2022。计算机科学讲义，第 13257 卷，第 3–12 页。施普林格，尚（2022）。https://doi.org/10.1007/ 978-3-031-05578-2_1