arXiv:2301.05338v1 [cs.DS] 13 Jan 2023

在 Wheeler 确定有限自动机上计算匹配统计

Alessio Conte[1],Nicola Cotumaccio[2,3], Travis Gagie[3], Giovanni Manzini[1],
Nicola Prezza[4] 和 Marinella Sciortino[5]

[1] 意大利比萨，比萨大学，alessio.conte@unipi.it，giovanni.manzini@unipi.it[2] 意大利拉奎拉，GSSI，nicola.cotumaccio@gssi.it[3] 加拿大哈利法克斯，达尔豪斯大学，nicola.cotumaccio@dal.ca，travis.gagie@dal.ca[4] 意大利威尼斯，Ca' Foscari 大学，nicola.prezza@unive.it[5] 意大利巴勒莫大学，marinella.sciortino@unipa.it

摘要

匹配统计最初是为解决近似字符串匹配问题而提出的，而该问题是生物信息学应用中反复使用的子程序。2010年，Ohlebusch 等人 [SPIRE 2010]提出了一种时间和空间效率较高的算法，用于计算匹配统计；该算法依赖于压缩后缀树的若干组成部分，尤其是最长公共前缀（LCP）数组。本文展示了如何将他们的算法从字符串推广到 Wheeler 确定性有限自动机。最重要的是，我们为 Wheeler 自动机引入了最长公共前缀数组的概念，从而迈出了将（压缩）后缀树功能扩展到带标号图的第一步。

引言

给定字符串 T 和模式 π，模式匹配问题的经典表述要求判断模式 π 是否出现在字符串 T 中，并且可能还要统计出现次数并报告出现位置。FM 索引 [1], 的发明以 Burrows-Wheeler 变换 [2], 为基础，为模式匹配领域开辟了新的研究方向。FM 索引背后的索引与压缩技术深深依赖于后缀排序的思想，多年来已从字符串推广到树 [3], De Brujin 图 [4,5], Wheeler 图 [6,7] 和任意图 [8,9]。特别是，Wheeler 图这一类图可能以最简单的方式捕捉了 FM 索引背后的直觉，而 Wheeler 序的概念在自动机理论 [7,10] 中确实有重要影响。

然而，在生物信息学中，我们感兴趣的不仅是精确模式匹配，还有模式匹配问题的诸多变体 [11]。尤其是，引入匹配统计是为了解决近似模式匹配问题 [12]。后缀树是一种功能强大的数据结构，能够一次处理模式匹配问题的各种变体 [13]。后缀树的主要缺点是空间占用较大，无论在理论上还是实践中都不可忽视。因此，后缀树已被后缀数组取代 [14]。虽然后缀数组并不具备后缀树的所有功能，但已有研究表明，可以为其增添一些数据结构——尤其是最长公共前缀（LCP）数组——从而实现后缀树的全部功能 [15]。所有这些组件都可以有效压缩，由此得到所谓的压缩后缀树 [16]。

自然会想到一个问题：能否将后缀树的功能不仅应用于字符串，也应用于图，尤其是 Wheeler 图？本文通过研究计算匹配统计的问题，首次给出了部分肯定的答案。2010 年，Ohlebusch 等人 [17] 提出了一种时间和空间效率俱佳的算法，用于计算匹配统计，该算法依赖于压缩后缀树的一些组件。本文展示了如何将他们的算法从字符串推广到 Wheeler 确定性有限自动机。最重要的是，我们为 Wheeler 自动机引入了最长公共前缀（LCP）数组的概念，从而为将（压缩）后缀树的功能扩展到带标号图迈出了重要一步。

符号与初步定义

全文中，我们考虑一个字母表 Σ，以及 Σ上的一个固定全序 $\preceq$。我们用 Σ^* 表示 Σ 上所有有限字符串的集合，用 Σ^ω 表示 Σ上所有（可数）无限字符串的集合。空字是 ϵ。若 $\alpha \in \Sigma^*$，则 α^R 是 α的反向字符串。我们将全序 $\preceq$ 从 Σ 扩展到 $\Sigma^* \cup \Sigma^\omega$，并按字典序排列。若 i 和 j 是整数，且 $i \leq j$，则定义 $[i, j] = \{i, i+1, \ldots, j-1, j\}$。若 T是一个字符串，则其第 i个字符为 $T[i]$，且$T[i..j] = T[i]..T[j]$。

我们考虑确定性自动机 $\mathcal{A} = (Q, E, s_0, F)$，其中 Q 是状态集合，$E \subseteq Q \times Q \times \Sigma$ 是带标记边的集合，$s_0 \in Q$ 是初始状态，$F \subseteq Q$ 是终态集合。根据定义，对于每个 $u \in Q$ 和每个 $a \in \Sigma$，从 u出发、标记为 a 的边至多有一条。按照 [7, 10], ，我们假设 s_0 没有入边，且每个状态都可从初始状态到达；此外，所有进入同一状态的边都带有相同标记（输入一致性），因此对于每个 $u \in Q \setminus \{s_0\}$，我们可以令 $\lambda(u)$ 表示所有进入 u的边的标记。我们定义 $\lambda(s_0) = \#$，其中 $\# \notin \Sigma$ 是一个特殊字符，并假设对于每个 $a \in \Sigma$ 都有 $\# \prec a$ （字符 # 类似于后缀树和后缀数组中使用的终止字符 \$）。因此，边 (u', u, a)可以简写为 (u', u)，因为它必定是 $a = \lambda(u)$。

我们假设读者熟悉后缀数组（SA）、Burrows–Wheeler 变换（BWT）、FM 索引和反向搜索的概念 [1]。

模式 $\pi = \pi[1..m]$ 相对于字符串 $T = T[1..n]$ 的匹配统计量定义如下。假设 $T[n] = \$ \notin \Sigma$，其中 $\$ \prec a$ 对每个 $a \in \Sigma$ 都成立。确定 π 相对于 T 的匹配统计量，意味着要对 $1 \leq i \leq m$ 确定：(i) $\pi[i..m]$ 中在 T 里出现的最长前缀 π'；以及 (ii) 在所有按字典序排序的后缀列表中，以 π' 开头的所有字符串所对应的区间。我们可以用三个值来描述 (i) 和 (ii)：π' 的长度 ℓ_i，以及 (ii) 中所考虑区间的端点 l_i 和 r_i。例如，令 $T = mississippi\$$（见图 1），且 $\pi = stpissi$。对于 $i = 1$，我们有 $\pi' = s$，因此 $\ell_1 = 1$ 且 $[l_1, r_1] = [9, 12]$（以 s 开头的后缀）。对于 $i = 2$，我们有 $\pi' = \epsilon$，因此 $\ell_2 = 0$ 且 $[l_2, r_2] = [1, n] = [1, 12]$（所有后缀都以空字符串开头）。对于 $i = 3$，我们有 $\pi' = pi$，因此 $\ell_3 = 2$，且 $[l_3, r_3] = [7, 7]$（以 pi 开头的后缀）。对于 $i = 4$，我们有 $\pi' = issi$，因此 $\ell_4 = 4$，且 $[l_4, r_4] = [4, 5]$（以 $issi$ 开头的后缀）。对于 $i = 5$，可以依此类推，6、7。

i	排序后的后缀	最长公共前缀	后缀数组	BWT
1	$		12	i
2	i$	0	11	p
3	ippi$	1	8	s
4	issippi$	1	5	s
5	ississippi$	4	2	m
6	mississippi$	0	1	$
7	pi$	0	10	p
8	ppi$	1	9	i
9	sippi$	0	7	s
10	sissippi$	2	4	s
11	ssippi$	1	6	i
12	ssissippi$	3	3	i

图1：“mississippi$”的排序后缀以及最长公共前缀、后缀数组和 Burrows–Wheeler变换数组。

计算字符串的匹配统计

我们首先介绍 Ohlebusch 等人提出的算法 [17], ，并强调在转向 Wheeler 确定有限自动机时将推广的思路。该算法通过反向搜索，利用与 m 成线性关系的迭代次数来计算匹配统计。我们从 π 的末尾开始，并使用反向搜索（从区间 $[1, n]$ 开始，该区间对应于以空字符串为前缀的后缀集合），以查找 π 在 T 中所有出现位置对应的区间（如果存在）。然后，从新的区间开始，我们使用反向搜索来查找 π 的长度为 2 的后缀在 T 中所有出现位置对应的区间（如果存在），依此类推。在某个时刻，对于某个 $i \leq m+1$，可能会出现这样的情况： $\pi[i..m]$ 在 T 中出现，但下一次反向搜索返回空区间，因此 $\pi[i-1..m]$ 并未出现在 T 中（情况 $i = m + 1$ 对应于初始设置，此时 $\pi[i..m]$ 为空字符串）。我们区分两种情况：

- (情况 1) 如果 $l_i = 1$ 且 $r_i = n$，这意味着 T 的所有后缀都以 $\pi[i..m]$为前缀。特别是，当 $i = m + 1$时可能会出现这种情况：这意味着第一次反向搜索未成功。我们立即得出结论，字符 $\pi[i-1]$不出现在 T中，因此 $\ell_{i-1} = 0$和 $[l_{i-1}, r_{i-1}] = [1, n]$ （因为所有后缀都以空字符串开头）。在这种情况下，在算法接下来的迭代中，我们可以直接舍弃 $\pi[i-1, m]$：当我们要为 $i' \leq i-2$ 搜索 $\pi[i', m]$中出现的最长前缀时，只需搜索 $\pi[i', i-2]$ 中出现的最长前缀即可。

- (情况 2) 如果 $l_i > 1$ 或 $r_i < n$，这意味着以$\pi[i..m]$ 开头的 T 后缀数量少于 n。现在，每个以 $\pi[i..m]$ 开头的后缀也以 $\pi[i..m-1]$开头。如果以 $\pi[i..m-1]$ 开头的后缀数量等于以 $\pi[i..m]$开头的后缀数量，那么 $\pi[i-1..m-1]$ 也不会出现在 T中。更一般地，对于 $j \leq m-1$，只有当以 $\pi[i..j]$ 开头的后缀数量大于以 $\pi[i..m]$开头的后缀数量时， $\pi[i-1..j]$ 才会出现在 T中。由于我们关注的是极大匹配，我们希望 j 尽可能大：稍后我们将说明

如何计算最大的整数 j，使得以$\pi[i..j]$ 开头的后缀数量大于以 $\pi[i..m]$开头的后缀数量。注意，j 始终存在，因为所有 n 后缀都以空字符串开头，但以 $\pi[i..m]$开头的后缀少于 n 个。确定 j 后，我们舍弃 $\pi[j+1..m]$（因此，在算法接下来的迭代中，我们只需考虑 $\pi[1..j]$），并从与 $\pi[i..j]$ 出现位置关联的区间开始，递归应用反向搜索——我们还将说明如何计算这个区间。

让我们将上述算法应用于 $T = mississippi\$$ 和 $\pi = stpissi$。我们从区间 $[1,n] = [1,12]$ 开始，它对应于空模式，以及字符 $\pi[7] = i$。执行反向步骤后得到区间 $[l_7, r_7] = [2,5]$（以 i 开头的后缀），因此 $\ell_7 = 1$。现在，我们从 $[2,5]$ 和 $\pi[6] = s$ 执行反向步骤，得到 $[l_6, r_6] = [9,10]$（以 si 开头的后缀），因此 $\ell_6 = 2$。接着，我们从 $[9,10]$ 和 $\pi[5] = s$ 执行反向步骤，得到$[l_5, r_5] = [11,12]$（以 ssi 开头的后缀），因此 $\ell_5 = 3$。再接着，我们从 $[11,12]$ 和 $\pi[4] = i$ 执行反向步骤，得到 $[l_4, r_4] = [4,5]$（以 $issi$ 开头的后缀），因此 $\ell_4 = 4$。现在，我们从 $[4,5]$ 和 $\pi[3] = p$ 执行反向步骤，得到空区间。这意味着没有后缀以 $pissi$ 开头。请注意图1中，以 $issi$ 开头的后缀数量等于以 iss 或 is 开头的后缀数量，但以 i 开头的后缀数量更多。因此，我们考虑所有以 i 开头的后缀所对应的区间——即 $[2,5]$ ——并用 $\pi[3] = p$ 执行反向步骤。这次反向步骤成功了，我们得到 $[l_3, r_3] = [7,7]$（以 pi 开头的后缀），并且$\ell_3 = 2$。现在，我们从 $[7,7]$ 和 $\pi[2] = t$ 执行反向步骤，得到空区间。这意味着没有后缀以 tpi 开头。请注意图1中，以 p 开头的后缀数量多于以 pi 开头的后缀数量。对应的区间是 $[7,8],$，但用 $\pi[2] = t$ 执行反向步骤仍不成功（因此没有后缀以 tp 开头）。以 p 开头的后缀数量少于以空字符串开头的后缀数量（后者等于 $n = 12$），因此我们用$[1,12]$ 和 $\pi[2] = t$ 执行反向步骤。由于反向步骤仍未成功，我们得出结论：$\pi[2] = t$ 未在 S 中出现，因此 $[l_2, r_2] = [1,n] = [1,12]$ 和 $\ell_2 = 0$。最后，我们重新从整个区间 $[1,12],$ 开始，用 $\pi[1] = s$ 执行反向步骤，得到 $[l_1, r_1] = [9,12]$（以 s 开头的后缀），因此 $\ell_1 = 1$。

很容易看出，迭代次数关于 m 呈线性增长。事实上，每次执行反向步骤时，我们要么向左跨过 π，以计算新的匹配统计量；要么将 π 的后缀长度至少增加 1，而该后缀之后将被永久丢弃。这意味着迭代次数上限为 $2|\pi| = 2m$。

我们只需说明 (i) 如何计算 j，以及 (ii) 如何计算算法的情况 2 中所有以 $\pi[i..j]$ 开头的后缀所构成的区间。为此，我们引入 T的最长公共前缀（LCP）数组 $\mathrm{LCP} = \mathrm{LCP}[2,n]$。我们将 $\mathrm{LCP}[i]$ 定义为 T 中按字典序第（$i-1$）小的后缀与按字典序第 i小的后缀 T之间最长公共前缀的长度。在图1中，$\mathrm{LCP}[5] = 4$，因为 T 中按字典序第四小的后缀是 $issippi\$$，按字典序第五小的后缀是T $ississippi\$$，而 $issippi\$$与 $ississippi\$$的最长公共前缀是 $issi$，长度为 4。请记住，在这个例子中，从 $[4,5]$（以 $issi$开头的后缀）和 p 开始的反向搜索未能成功，因此计算 j 意味着确定 $issi$ 的最长前缀，使得以该前缀开头的后缀数量大于 2。

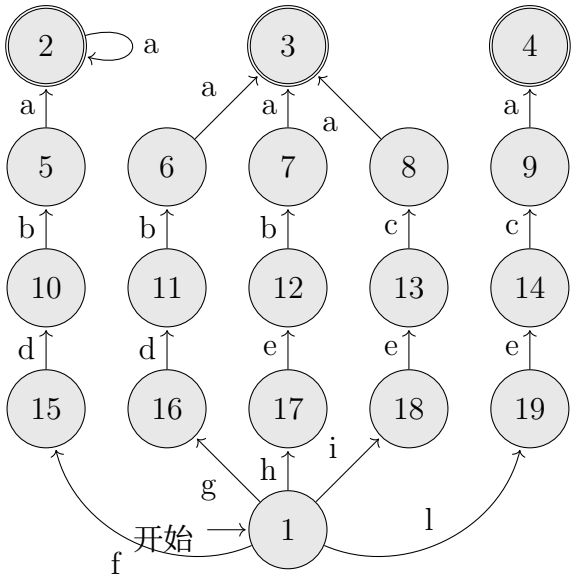


图2：一个 Wheeler 确定有限自动机。状态按照其在 Wheeler 序中的位置编号。

利用最长公共前缀数组即可轻松计算：所需的最长前缀长度为 $\max\{\mathrm{LCP}[4],\mathrm{LCP}[6]\} = \max\{1,0\} = 1$，因此所需前缀为 i。由此，我们只需说明如何计算所有以该前缀 i 开头的后缀所构成的区间，也就是 $[2,5]$。注意，要计算这个区间，只需向两个方向扩展区间 $[4,6]$，直到最长公共前缀值小于 1。由于 $\mathrm{LCP}[4] = 1$、$\mathrm{LCP}[3] = 1$和 $\mathrm{LCP}[2] = 0$，且我们已经知道 $\mathrm{LCP}[6] = 0$，因此可以得出所需区间为 $[2,5]$。换言之，给定位置 t，我们必须能够计算小于 t 的最大整数 k，使得 $\mathrm{LCP}[k] < \mathrm{LCP}[t]$；还要计算大于 t 的最小整数 k，使得 $\mathrm{LCP}[k] < \mathrm{LCP}[t]$（在我们的例子中，$t = 4$）。这类查询称为 PSV（“前一个较小值”）和下一个较小值查询。可以通过增强最长公共前缀数组来高效地解决 PSV 和下一个较小值查询：存在不同的空间—时间权衡，详情请参阅 [17]。

Wheeler确定有限自动机的匹配统计

下面来定义 Wheeler 确定有限自动机 [7]。

定义 1。设 $\mathcal{A} = (Q, E, s_0, F)$ 是一个确定有限自动机。在 $\mathcal{A}$ 上的 Wheeler 序是 Q 上满足以下条件的全序 $\leq$：对于每个 $u \in Q$，均有 $s_0 \leq u$，且：

（公理 1）如果 u、$v \in Q$ 和 $u < v$，那么 $\lambda(u) \preceq \lambda(v)$。（公理 2）如果 (u', u)、$(v', v) \in E$，$\lambda(u) = \lambda(v)$，且 $u < v$，那么 $u' < v'$。

如果确定有限自动机 $\mathcal{A}$ 具有 Wheeler 序，则称其为 Wheeler 自动机。

不难验证，该定义与 [7], 中的定义等价；其中已证明，如果某个 DFA $\mathcal{A}$ 具有 Wheeler 序 $\leq$，那么 $\leq$ 是唯一确定的（即，

$\leq$ 是 $\mathcal{A}$ 上的 Wheeler 序）。下文中，我们固定一个 Wheeler 确定有限自动机 $\mathcal{A} = (Q, E, s_0, F)$，并假设 $Q = \{u_1, \ldots, u_n\}$，其中 $u_1 < u_2 < \cdots < u_n$ 按 Wheeler 序排列，且 u_1 与初始状态 s_0 重合。示例见图2。

现在我们说明，Wheeler 序可以看作所有状态的一个排列，其作用与字符串的后缀数组相同。下文中，为方便起见，我们在概念上假设 s_0 有一条标记为 # 的自环（这与公理 1 一致，因为对于每个 $a \in \Sigma$，# $\prec a$ ）。这意味着每个状态至少有一条入边，因此对于每个状态 u_i，至少存在一个无限字符串 $\alpha \in \Sigma^\omega$，可以从 u_i 开始沿边反向读取（例如，在图2中，对于 u_9，这样的字符串是 $cel\#\#\#\ldots$）。我们用 I_{u_i} 表示所有此类字符串的集合。形式化地说：

定义2。令 $i \in [1, n]$。对于每个状态 $u_i \in Q$，定义：

$$I_{u_i} = \{\alpha \in \Sigma^\omega \mid \textit{there exist integers } f_1, f_2, \ldots \textit{ in } [1, n] \textit{ such that (i) } f_1 = i, \\ \textit{(ii) } (u_{f_{k+1}}, u_{f_k}) \in E \textit{ for every } k \geq 1 \textit{ and (iii) } \alpha = \lambda(u_{f_1})\lambda(u_{f_2})\ldots\}.$$

例如，在图2中，我们有 $I_{u_3} = \{abdg\#\#\#\ldots, abeh\#\#\#\ldots, acei\#\#\#\ldots\}$。

下面的引理表明，由 Wheeler 序定义的状态排列，正是按照字典序排列进入各状态的字符串的排列；这与后缀数组定义的排列按照字典序排列字符串的后缀相同（后缀可视为“离开”文本位置的字符串）。

引理3。设 $i, j \in [1, n]$，其中 $i < j$。设 $\alpha \in I_{u_i}$ 和 $\beta \in I_{u_j}$。则 $\alpha \preceq \beta$。

证明。设 $f_1, f_2, \ldots$ 是 $[1, n]$ 中满足以下条件的元素：(i) $f_1 = i$；(ii) 对每个 $k \geq 1$，$(u_{f_{k+1}}, u_{f_k}) \in E$；(iii) $\alpha = \lambda(u_{f_1})\lambda(u_{f_2})\ldots$。类似地，设 $g_1, g_2, \ldots$ 是 $[1, n]$ 中满足以下条件的元素：(i) $g_1 = j$；(ii) 对每个 $k \geq 1$，$(u_{g_{k+1}}, u_{g_k}) \in E$；(iii) $\beta = \lambda(u_{g_1})\lambda(u_{g_2})\ldots$。设 $\alpha \neq \beta$。我们必须证明 $\alpha \prec \beta$。令 $p \geq 1$ 为最小整数，使得 α 的第 p 个字符与 β 的第 p 个字符不同。换言之，我们知道 $\lambda(u_{f_1}) = \lambda(u_{g_1})$，$\lambda(u_{f_2}) = \lambda(u_{g_2})$，$\ldots$，$\lambda(u_{f_{p-1}}) = \lambda(u_{g_{p-1}})$，但 $\lambda(u_{f_p}) \neq \lambda(u_{g_p})$。我们必须证明 $\lambda(u_{f_p}) \prec \lambda(u_{g_p})$。由于 $\lambda(u_{f_1}) = \lambda(u_{g_1})$ $f_1 = i < j = g_1$，且 (u_{f_2}, u_{f_1})、$(u_{g_2}, u_{g_1}) \in E$，由公理 2 可得 $f_2 < g_2$。由于 $\lambda(u_{f_2}) = \lambda(u_{g_2})$、$f_2 < g_2$，且 (u_{f_3}, u_{f_2})、$(u_{g_3}, u_{g_2}) \in E$，由公理 2 可得 $f_3 < g_3$。重复这一论证，我们得出 $f_p < g_p$。由公理 1 可得 $\lambda(u_{f_p}) \preceq \lambda(u_{g_p})$。由于 $\lambda(u_{f_p}) \neq \lambda(u_{g_p})$，我们得出 $\lambda(u_{f_p}) \prec \lambda(u_{g_p})$。 □

如果把字符串看作一条带标签的路径，那么后缀数组会对从每个位置向前移动时可以读出的字符串（即该字符串的后缀）进行排序，而 Wheeler 序则会对从每个位置向后朝初始状态移动时可以读出的字符串进行排序。其基本思想相同：向前与向后的差别仅仅源于历史原因 [6]。要计算 Wheeler 确定有限自动机上的匹配统计，我们采用上一节中的推理方法，将反向搜索替换为如下定义的前向搜索 [6]：给定 $[1, n]$ 中的一个区间 $[i, j]$ 和 $a \in \Sigma$，求 $[1, n]$ 中（可能为空的）区间 $[i', j']$，使得状态 $v_{k'}$ 可从某个状态 v_k 经由一条标记为 a 的边到达，其中 $i \leq k \leq j$，当且仅当 $i' \leq k' \leq j'$（这可由定义 1 的公理直接推出）。对于大小固定的字母表，给定 $[i, j]$ 和 a，即可在常数时间内确定 $[i', j']$。给定字符串 $\pi \in \Sigma^*$，若从全体状态开始并反复应用前向搜索，则会到达所有满足存在 $\alpha \in I_{u_i}$ 的状态 u_i

以 π^R 为前缀；这是相对于 Wheeler 序的一个区间：下文将此区间称为 $T(\pi)$。

由于前向与反向的差异，匹配统计问题将在 Wheeler 确定有限自动机上以对称的方式定义。给定模式 $\pi = \pi[1..m]$，对于每个$1 \le i \le m$，我们希望确定 (i) $\pi[1..i]$ 中在 Wheeler 确定有限自动机 $\mathcal{A}$ 中出现的最长后缀 π'（也就是说，可以通过拼接边在 $\mathcal{A}$ 上的某处读取该后缀），以及 (ii) 区间 $T(\pi')$ 的端点。

总的来说，我们可以以对称的方式应用字符串算法中的相同思路。我们从 π 的开头开始（而不是从 π的末尾开始），并且最初考虑整个状态集合。我们反复进行前向搜索（而不是反向搜索），直到对于某个 $i \ge 0$，前向搜索返回空区间。这意味着$\pi[1..i+1]$没有出现在 $\mathcal{A}$中。然后，如果 $T(\pi[1..i])$ 是整个状态集合，我们就得出结论：字符 $\pi[i+1]$ 没有标记图中的任何边。否则，我们必须找到最小的 j，使得 $T(\pi[1..i])$ 严格包含于 $T(\pi[j..i])$（也就是说，我们必须确定 $\pi[1..i]$ 中能够到达比 $\pi[1..i]$更多状态的最长后缀 $\pi[j..i]$）。然后，我们必须确定区间 $T(\pi[j..i])$ 的端点，以便继续进行前向搜索。

现在的挑战是找到一种方法，来解决我们在字符串算法的情况 2 中确定的相同子问题。换句话说，我们必须找到一种方法来确定 j 并找出区间 $T(\pi[j..i])$ 的端点。我们将说明，这一解法并不像字符串算法中的解法那么简单。

Wheeler 确定有限自动机的最长公共前缀数组和匹配统计

我们首先注意到，I_{u_i} 可能是一个无限集合。例如，在图2中，我们有

$$I_{u_2} = \{aaaaa\ldots, abdf\#\#\#\ldots, aabdf\#\#\#\ldots, aaabdf\#\#\#\ldots, \ldots\}.$$

一般而言，Σ^ω 中的按字典序排列的无限字符串集合未必有最小值或最大值。例如，集合 $\{baaaa\ldots, abaaa\ldots, aabaa\ldots, aaaba\ldots\}$没有最小值（但有下确界字符串 $aaaaa\ldots$）。尽管如此，引理3意味着每个 I_{u_i} 都有最小值和最大值。例如，最小值可按如下方式求得。令 $f_1 = i$，并对每个 $k \ge 1$，递归地令 f_{k+1} 为满足 $(u_{f_{k+1}}, u_{f_k}) \in E$ 的 $[1, n]$ 中的最小整数。于是，I_{u_i}的最小值为 $\lambda(u_{f_1})\lambda(u_{f_2})\ldots$，最大值也可用类似方法确定。

下文中，我们分别用 min_i 和 max_i 表示 I_{u_i} 的最小值和最大值（例如，在图2中，我们有 $\mathsf{min}_2 = aaaaa\ldots$，以及 $\mathsf{max}_2 = abdf\#\#\#\ldots$）。引理3意味着：

$$\mathrm{min}_1 \preceq \mathrm{max}_1 \preceq \mathrm{min}_2 \preceq \mathrm{max}_2 \preceq \cdots \preceq \mathrm{max}_{n-1} \preceq \mathrm{min}_n \preceq \mathrm{max}_n.$$

这启发我们将最长公共前缀数组推广如下。给定 α、$\beta \in \Sigma^* \cup \Sigma^\omega$，令 $\mathsf{lcp}(\alpha, \beta)$ 表示 α 和 β 的最长公共前缀的长度（如果 $\alpha = \beta \in \Sigma^\omega$，则定义 $\mathsf{lcp}(\alpha, \beta) = \infty$）。

定义 4. Wheeler 自动机 $\mathcal{A}$ 的最长公共前缀数组 $\mathrm{LCP}_{\mathcal{A}} = \mathrm{LCP}_{\mathcal{A}}[2, 2n]$按以下顺序包含这些 $2n-1$ 值：$\mathsf{lcp}(\mathsf{min}_1,\mathsf{max}_1)$、$\mathsf{lcp}(\mathsf{max}_1,\mathsf{min}_2)$、$\mathsf{lcp}(\mathsf{min}_2, \mathsf{max}_2)$、.....、$\mathsf{lcp}(\mathsf{max}_{n-1}, \mathsf{min}_n)$、$\mathsf{lcp}(\mathsf{min}_n, \mathsf{max}_n)$。

根据上述对 $\min_i$ 和 $\max_i$的刻画，可以证明对于每个条目，要么是 $\mathrm{LCP}_{\mathcal{A}}[i] = \infty$，要么是 $\mathrm{LCP}_{\mathcal{A}}[i] < 3n$（由 Fine–Wilf 定理 [18, 19]可得）；并且可以设计一个多项式时间算法来计算 $\mathrm{LCP}_{\mathcal{A}}$。

遗憾的是，仅凭数组最长公共前缀$_{\mathcal{A}}$ 不足以计算匹配统计。假设 $T(\pi) = \{u_r, u_{r+1}, \ldots, u_{s-1}, u_s\}$，并且在通过添加字符 c进行前向搜索后，我们得到 $T(\pi c) = \emptyset$。接着，我们必须确定 π'的最大后缀，使其属于 $T(\pi)$，且 $T(\pi)$严格包含于 $T(\pi')$。假设 I_{u_r} 中的每个字符串都以 π^R为前缀，而 I_{u_s} 中的每个字符串都以 π^R为前缀。特别地，$\min_r$ 和 $\max_s$都以 π^R为前缀。在这种情况下，我们可以像字符串算法那样处理：所需的后缀 π' 长度为 $\max\{\mathsf{lcp}(\max_{r-1}, \min_r)$、$\mathsf{lcp}(\max_s, \min_{s+1})\}$，可利用最长公共前缀$_{\mathcal{A}}$确定。然而，一般来说，即使 I_{u_r} 中的某个字符串必定以 π^R为前缀，字符串 $\min_r$ 也未必以 π^R为前缀；同样，$\max_s$ 也未必以 π^R为前缀。最坏情况出现在 $r = s$时。考虑图2，并假设$\pi = heba$。于是，我们有 $r = s = 3$（注意，$abeh\#\#\#\ldots$是 I_{u_3}中的一个字符串，且以π^R为前缀）。然而，$\min_3 = abdg\#\#\#\ldots$和 $\max_3 = acei\#\#\#\ldots$都不以$\pi^R$为前缀。注意，$\mathsf{lcp}(\max_2, \min_3) = 3$，且 $\mathsf{lcp}(\max_3, \min_4) = 3$，但 π' 并不是 π中长度为 3 的后缀。事实上，由于 $\min_3$ 仅以 π^R 中长度为 2 的前缀为前缀，而 $\max_3$仅以 π^R 中长度为 1 的前缀为前缀，因此我们可以得出 $|\pi'| = 2$。一般而言，所需的后缀 π' 长度为 $|\pi'|$，其定义如下：

$$\max\Big\{\min\{\mathsf{lcp}(\max_{r-1}, \min_r), \mathsf{lcp}(\min_r, \pi^R)\}, \min\{\mathsf{lcp}(\pi^R, \max_s), \mathsf{lcp}(\max_s, \min_{s+1})\}\Big\}. \quad (1)$$

上述公式表明，为了计算 π'，除了最长公共前缀$_{\mathcal{A}}$ 之外，只需知道 $\mathsf{lcp}(\min_r, \pi^R)$ 和 $\mathsf{lcp}(\pi^R, \max_s)$ 的值（π'是 π 的一个后缀，因此可由其长度确定）。现在我们将说明，在计算匹配统计的过程中，我们的算法如何高效地维护当前模式 π、集合 $T(\pi) = \{u_r, u_{r+1}, \ldots, u_{s-1}, u_s\}$，以及 $\mathsf{lcp}(\min_r, \pi^R)$ 和 $\mathsf{lcp}(\pi^R, \max_s)$ 的值。我们假设输入自动机采用秩/选数据结构编码，支持在$O(\log|\Sigma|)$时间内执行一步前向搜索，详情参见 [6] 。此外，我们还将使用以下结果。

引理 5. 设 $A[1, n]$ 是有序字母表 Σ 上的一个值序列。考虑以下查询：(i) 给定 i, $j \in [1..n]$，计算 $S[i..j]$中的最小值；(ii) 给定 $t \in [1..n]$ 和 $c \in \Sigma$，确定满足$A[k] < c$ 的最大 $k < t$ （或最小 $k > t$）。那么，可以为 A 增添一个大小为 $2n + o(n)$ 位的数据结构，使得查询 (i) 可在常数时间内回答，查询 (ii) 可在 $O(\log n)$ 时间内回答。

证明。存在一个占用 $2n + o(n)$ 位的数据结构，能够在常数时间内执行区间最小值查询 [20], ，因此使用 A，我们可以在常数时间内解决查询 (i)。现在，我们来说明如何解决查询 (ii)。设 f_1 是在输入 $i = \lceil t/2 \rceil$ 和 $j = t - 1$ 上执行查询 (i) 的答案。如果 $f_1 < c$，那么我们必须继续在区间 $[\lceil t/2 \rceil, t - 1]$ 中搜索；否则，必须继续在区间 $[1, \lceil t/2 \rceil - 1]$ 中搜索。换言之，我们可以通过对 $[1, t - 1]$ 进行二分查找来回答查询 (ii)，耗时 $O(\log\ t)$（因此耗时 $O(\log\ n)$）。 □

注意，查询 (ii) 可以看作 PSV 和下一个较小值查询的变体。下文中，我们假设数组 $\mathrm{LCP}_{\mathcal{A}}$ 已通过引理 5 中的数据结构进行了增强。

起初我们有 $\pi = \epsilon$，因此 $T(\epsilon) = \{1, 2, \ldots, n\}$，并且显然 $\mathsf{lcp}(\min_r, \pi^R)$ =$\mathsf{lcp}(\pi^R, \max_s) = 0$。每次迭代时，我们执行一步前向搜索，在给定 $T(\pi)$ 的情况下计算 $T(\pi c)$；然后根据 $T(\pi c)$ 是否为空，区分两种情况。

情况 1。 $T(\pi c) = \{u_{r'}, u_{r'+1}, \ldots, u_{s'-1}, u_{s'}\}$ 不为空。在这种情况下， πc 将在下一次迭代中成为模式。由于我们已经有 $T(\pi c)$，因此只需计算 $\mathsf{lcp}(\min_{r'}, c\pi^R)$ 和 $\mathsf{lcp}(c\pi^R, \max_{s'})$。我们仅说明如何计算 $\mathsf{lcp}(\min_{r'}, c\pi^R)$，另一个计算与此类似。设 k 为 $[1, n]$ 中满足 $(u_k, u_{r'}) \in E$的最小整数。注意，利用用于编码 $\mathcal{A}$的紧凑数据结构上的标准 rank/select 操作，我们可以轻松计算 k。由于 $u_{r'} \in T(\pi c)$，必有 $k \leq s$。此外，我们前面描述的 $\min_{r'}$ 的刻画表明，$\min_{r'} = c\min_k$，因此 $\mathsf{lcp}(\min_{r'}, c\pi^R) = \mathsf{lcp}(c\min_k, c\pi^R) = 1 + \mathsf{lcp}(\min_k, \pi^R)$。为计算 $\mathsf{lcp}(\min_k, \pi^R)$，我们区分两种子情况：

a) $k > r$，因此 $r < k \leq s$。由于 $u_r, u_s \in T(\pi)$，存在 $\alpha \in I_{u_r}$和 $\beta \in I_{u_s}$，它们都以 π^R 为前缀。但 $\alpha \preceq \max_r \preceq \min_k \preceq \min_s \preceq \beta$，因此 $\min_k$ 也以 π^R 为前缀，由此可得 $\mathsf{lcp}(\min_k, \pi^R) = |\pi|$。

b) $k \leq r$。在这种情况下，我们有 $\min_k \preceq \max_k \preceq \min_{k+1} \prec \max_{k+1} \preceq \cdots \preceq \min_r \prec \pi^R$，因此 $\mathsf{lcp}(\min_k, \pi^R)$ 等于

$$\min\{\mathsf{lcp}(\min_k, \max_k), \text{最长公共前缀}(\max_k, \min_{k+1}), \text{最长公共前缀}(\min_{k+1}, \max_{k+1}) \ldots, \text{最长公共前缀}(\min_r, \pi^R)\}.$$

利用上述公式，我们可以通过引理 5 的查询 (i)，在范围 $\mathrm{LCP}_{\mathcal{A}}[2k, 2r-1]$ 内并结合值 $\mathsf{lcp}(\min_r, \pi^R)$，计算 $\mathsf{lcp}(\min_k, \pi^R)$。

情况 2。 $T(\pi c)$ 为空。在这种情况下，在下一次迭代中，模式将是 π 的最大后缀π'，使得 $T(\pi)$ 严格包含于 $T(\pi') = \{u_{r''}, \ldots, u_{s''}\}$。我们使用 (1) 计算 $|\pi'|$；如果 $|\pi'| > \mathsf{lcp}(\min_r, \pi^R)$，则设定 $r'' = r$；否则，我们应用引理 5 的查询 (ii)，在 $\mathrm{LCP}_{\mathcal{A}}[2, 2r-1]$ 中找到小于 $|\pi'|$的最右侧条目 r''。计算 s'' 的过程类似。

给定 $T(\pi') = \{u_{r''}, u_{r''+1}, \ldots, u_{s''-1}, u_{s''}\}$，其中 $r'' \leq r$、 $s \leq s''$，且至少一个不等式为严格不等式，我们希望计算 $\mathsf{lcp}(\min_{r''}, (\pi')^R)$ 和 $\mathsf{lcp}((\pi')^R, \max_{s''})$。我们只考虑 $\mathsf{lcp}(\min_{r''}, (\pi')^R)$，另一项计算过程类似。我们区分两种子情况：

a) $r'' = r$. Then $\mathsf{lcp}(\min_{r''}, (\pi')^R) = \mathsf{lcp}(\min_r, (\pi')^R) = \min\{\mathsf{lcp}(\min_r, \pi^R), |\pi'|\}$.

b) $r'' < r$。特别地，由于 $u_{r''}$ 是 $T(\pi')$ 的左端点，且 $|T(\pi')| \geq 2$，我们可以像情况 $1a$) 中那样证明，$\max_{r''}$ 以 $(\pi')^R$ 为前缀。我们立即得出，$\mathsf{lcp}(\min_{r''}, (\pi')^R) = \min\{\mathsf{lcp}(\min_{r''}, \max_{r''}) |\pi'|\}$；由于 $\mathsf{lcp}(\min_{r''}, \max_{r''})$ 是存储在 $\mathrm{LCP}_{\mathcal{A}}$ 中的值，因此可以立即计算。

我们可以将上述讨论总结如下。

定理 6. 给定一个 Wheeler 确定有限自动机 $\mathcal{A}$，存在一种占用 $O(|\mathcal{A}|)$ 个字的 数据结构，可在 $O(|P| \log |\mathcal{A}|)$ 时间内计算模式 P 的模式匹配统计量。

资助 TG 获得了美国国立卫生研究院（NIH）美国国立过敏症和传染病研究所（NIAID）的资助（资助编号 HG011392）、美国国家科学基金会（NSF）IIBR 的资助（资助编号 2029552），以及加拿大自然科学与工程研究委员会（NSERC）发现基金的资助（资助编号 RGPIN-07185-2020）。GM 获得了意大利大学与研究部（PRIN 2017WR7SHH）的资助。MS 获得了 INdAM-GNCS 项目（CUP E55F22000270001）的资助。NP 获得了欧盟（欧洲研究委员会，REGINDEX，101039208）的资助。然而，文中表达的观点和意见仅代表作者本人，并不一定反映欧盟或欧洲研究委员会的立场。欧盟和拨款机构均不对此承担责任。

参考文献

[1] P. Ferragina 和 G. Manzini，“具有应用的机会数据结构”，载于第 41 届计算机科学基础年度研讨会论文集（FOCS’00），2000年，第 390–398 页。[2] M. Burrows 和 D. J. Wheeler，“一种块排序无损数据压缩算法”，技术报告，1994年。[3] P. Ferragina、F. Luccio、G. Manzini 和 S. Muthukrishnan，“构造带标记树以实现最优简洁性及更多应用”，载于第 46 届 IEEE 计算机科学基础年度研讨会论文集（FOCS’05），2005年，第 184–193 页。[4] A. Bowe、T. Onodera、K. Sadakane 和 T. Shibuya，“简洁 De Bruijn 图”，载于《生物信息学算法》，柏林、海德堡，2012年，第 225–235 页，Springer Berlin Heidelberg。[5] V. M¨akinen、N. V¨alim¨aki 和 J. Sir´en，“用于路径查询的图索引及其在基因组研究中的应用”，《IEEE/ACM计算生物学与生物信息学汇刊》，第 11 卷，第 375–388 页，2014年。[6] T. Gagie、G. Manzini 和 J. Sir´en，“Wheeler 图：基于 Burrows–Wheeler变换的数据结构框架”，《理论计算机科学》，第 698 卷，第 67–78 页，2017年。[7] J. Alanko、G. D’Agostino、A. Policriti 和 N. Prezza，“正则语言与前缀排序相遇”，载于第 31 届离散算法研讨会论文集（SODA’20），2020年，第 911–930 页，SIAM。[8] N. Cotumaccio 和 N. Prezza，“有限自动机的索引与压缩”，载于第 32 届离散算法研讨会论文集（SODA’21），2021年，第 2585–2599 页，SIAM。[9] N. Cotumaccio，“图可被简洁地编制索引，以便在 $O(|E|^2 + |V|^{5/2})$ 时间内进行模式匹配”，载于 2022年数据压缩会议（DCC），2022年，第 272–281 页。[10] J. Alanko、G. D’Agostino、A. Policriti 和 N. Prezza，“Wheeler语言”，《信息与计算》，第 281 卷，第 104820 页，2021年。[11] D. Gusfield，《字符串、树和序列算法：计算机科学与计算生物学》，剑桥大学出版社，1997年。[12] W. I. Chang 和 E. L. Lawler，“次线性近似字符串匹配及其生物学应用”，《Algorithmica》，第 12 卷，第 327–344 页，2005年。[13] P. Weiner，“线性模式匹配算法”，载于第 14 届 IEEE 开关与自动机理论年度研讨会论文集，1973年，第 1–11 页。[14] U. Manber 和 G. Myers，“后缀数组：一种在线字符串搜索的新方法”，《SIAM 计算杂志》，第 22 卷，第 5 期，第 935–948 页，1993年。[15] M. I. Abouelhoda、S. Kurtz 和 E. Ohlebusch，“用增强后缀数组替代后缀树”，《离散算法杂志》，第 2 卷，第 1 期，第 53–86 页，2004年。[16] K. Sadakane，“具有完整功能的压缩后缀树”，《理论计算系统》，第 41 卷，第 4 期，第 589–607 页，2007年。[17] E. Ohlebusch、S. Gog 和 A. K¨ugell，“在压缩全文索引上计算匹配统计和最大精确匹配”，载于第 17 届字符串处理与信息检索国际会议论文集（SPIRE’10），柏林、海德堡，2010年，第 347–358 页，施普林格出版社。[18] N. J. Fine 和 H. S. Wilf，“周期函数唯一性定理”，《美国数学会会刊》，第 16 期，第 109–114 页，1965年。[19] S. Mantaci、A. Restivo、G. Rosone 和 M. Sciortino，“Burrows-Wheeler 变换的扩展”，《理论计算机科学》，第 387 卷，第 3 期，第 298–312 页，2007年。[20] Johannes Fischer，“区间最小值查询的最优简洁性”，载于《LATIN 2010：理论信息学》，Alejandro L´opez-Ortiz 编，柏林、海德堡，2010年，第 158–169 页，Springer Berlin Heidelberg。