arXiv:2303.05336v1 [cs.DS] 9 Mar 2023

弹性创始图的改进与增强*

NicolaRizzo Massimo Equi
nicola.rizzo@helsinki.fi massimo.equi@helsinki.fi
Tuukka Norri Veli Mäkinen
tuukka.norri@helsinki.fi veli.makinen@helsinki.fi

芬兰赫尔辛基大学计算机科学系

摘要

用于模式匹配的带标号图索引是泛基因组学的一项核心挑战。Equi 等人（Algorithmica，2022）构建了弹性创始图（EFG），用于表示由字母表 Σ 及特殊间隙字符构成的字母所组成的 m 条长度为 n 的序列的比对：图中的路径可拼写出原始序列或其重组结果。通过强制满足半无重复性质，EFG 可以支持多项式空间索引，实现线性时间模式匹配，从而突破了带标号图索引的条件下界（Equi 等人，SOFSEM 2021）。在本文中，我们改进了 EFG 索引的空间效率，使其能够在线性时间内回答模式匹配查询；所需空间从与三个连续节点标签所拼写的所有字符串总长度成线性关系，降至与边标签总大小成线性关系。接着，我们开发了针对不同指标进行优化的线性时间构造算法：通过在树上解决新提出的排他祖先集问题，将现有的线性对数时间构造算法改进为 $O(mn)$；对于简化的无间隙情形，我们提出了一种 $O(mn)$ 时间的解，以最小化最大块高度，并进一步将块高度替换为前缀感知高度来推广该解。最后，为展示该框架的通用性，我们开发了基于 Burrows–Wheeler 变换的 EFG 索引，并研究如何对图中一组路径进行编码和执行文档列举查询，以报告哪些路径包含给定模式作为子串。我们提出 EFG 框架，作为无间隙情形下框架的改进与增强版本，并提供适用于所有涉及对齐序列分段的情形的构造方法。

关键词 多序列比对、模式匹配、分段算法、后缀树、文档列举

资助 本研究部分由欧盟地平线 2020 研究与创新计划资助，资助依据为玛丽·斯克沃多夫斯卡-居里资助协议第 956229 号（ALPACA）、欧洲研究理事会（ERC）资助协议第 851093 号（SAFEBIO）；此外还获得赫尔辛基信息技术研究所（HIIT）和芬兰科学院（资助编号 351149）的资助。

*本文扩展自 IWOCA 2022 [30]和 CPM 2022 [29]会议论文，并包含 Massimo Equi [11] 和 Tuukka Norri [26]博士论文项目中的部分成果。

1 引言

随着高通量测序技术的发展，在图中搜索字符串已成为一个核心问题。如今已有数千个人类基因组可用，它们共同构成了一个物种的所谓泛基因组 [35]。这种泛基因组可用于改进此前基于单一参考基因组开展的各种分析任务 [23, 31, 33, 17, 20, 10, 27]。泛基因组最常见的表示形式是图，其路径可拼写出输入基因组 [35]。在此类泛基因组图中，需要具备的基本操作是搜索查询字符串（短读段）在图中作为子路径出现的位置。遗憾的是，即使只是在含有 e 条边的图中查找长度为 q 的查询字符串的精确匹配，也无法显著快于 $O(qe)$ 时间；除非正交向量假设（OVH）不成立，否则任何在多项式时间内构建的索引都无法实现亚二次时间的字符串匹配 [13, 12]。因此，实用工具会采用各种启发式方法，或以其他泛基因组表示形式为基础。

由于一般图中的字符串搜索较为困难，Equi 等人 [14]研究了由多序列比对（MSA）得到的图，其中 MSA$[1..m, 1..n]$ 是一个矩阵，由 m 个对齐的行组成，每行都是长度为n的字符串，取自一个字母表 Σ 以及一个特殊的间隙符号 $-$。正如我们在第 2 节所述，对 MSA 的任何分段都会自然地导出一个由带标签节点组成的图，这些节点划分为若干块。边连接相邻的块，而原始序列则被拼写为该图中的路径；不过，该图还允许拼写出由这些序列重组而成的新序列。这种弹性创始图（EFG）如图1所示。关键的观察是，如果生成的节点标签不会作为任何其他路径的前缀出现（从同一个块出发的路径除外），那么就满足所谓的半无重复性质，并且存在一种基于特定长度路径所拼写字符串的图索引结构；该结构可在多项式时间内计算，并支持快速模式匹配。Equi 等人 [14] 还证明，这种可索引性是必要的，因为对于由多序列比对导出的 EFG，一般情况下正交向量假设（OVH）所导出的下界仍然成立。

此外，能够在多项式时间内计算出生成半无重复 EFG 的最优分段。在简化的无间隙情形中，半无重复性质等价于更强的无重复性质，我们有两种线性时间构造算法：M¨akinen 等人 [22] 提出了一种$O(mn)$时间算法，在给定无间隙多序列比对$[1..m, 1..n]$的情况下，构造最大块长度最小化的可索引弹性创始图；Equi 等人 [14] 实现了相同的时间复杂度，用于构造块数最多的最优弹性创始图（在半无重复要求下，这并非一项平凡的任务）。在图1的示例中，这两项指标分别为 5 和 3，因为由长度不超过 5 的片段定义了 3 个块。第三种优化准则，即最小化最大块高度，仍是一个开放问题。该指标定义为划分到单个块中的不同字符串的最大数量；在图1的示例中，该指标为 3，因为每个块最多包含 3 个节点。Equi 等人 [14] 还将这两种算法扩展到含间隙的一般多序列比对，得到了一种 $O(mn\log\ m)$时间预处理算法，该算法支持构造块数最多的半无重复 EFG，或者构造最大块长度最小的半无重复 EFG，所需时间分别为 $O(n)$和 $O(n\log\log\ n)$，

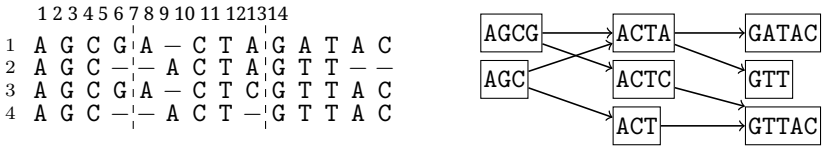

图1：由多序列比对（左）的分段所诱导的弹性创始图（右）。分段以一组垂直切线（虚线）表示。该弹性创始图是半无重复的，也就是说，每个节点标签在图中只会作为从该节点所在块出发的路径的前缀出现。因此，它可用于线性时间模式匹配的索引。

分别。我们将在第4节回顾这些结果；关于 EFG、弹性退化字符串和惠勒图之间的联系，请读者参阅前述论文。

本文将可索引弹性创始图的理论扩展如下：

- 在第3节中，我们为弹性创始图提供一个索引，可在 $O(|Q|)$ 时间内回答给定模式 Q 是否作为子路径出现在图中；该索引的大小与边标签串接所占空间呈线性关系。这在保持查询时间复杂度不变的同时，改进了 Equi 等人提出的索引空间 [14], ；后者与包含三个节点的路径所拼写的字符串大小呈线性关系。

- 在第4节中，我们研究弹性创始图的构建，并在对多序列比对进行通用预处理以找出有效分段后，将最大片段长度最小化的$O(n\log\log n)$时间解改进为 $O(n)$时间。此外，我们还优化了所得弹性创始图的高度 。对于无间隙情形，我们展示了如何将 Norri 等人利用左扩展 [28]提出的从左到右分段算法，与 Equi 等人 [22]计算最小左扩展 的方法相结合，从而在 $|\Sigma| \in O(m)$时得到一个 $O(mn)$算法。由于在一般情形和半无重复情形下无法使用左扩展，我们提出了一种等效的从左到右解法，利用有意义的右扩展，该解法在存在间隙时也适用。然而，这类扩展的数量为 $O(mn^2)$，计算它们需要 $O(mn\alpha \log|\Sigma|)$时间，其中 α 表示多序列比对中最长游程的长度；在该游程中，一行拼写出的字符串前缀是另一行拼写出的字符串的前缀，最坏情况下 $\alpha \in \Theta(n)$。因此，我们进一步采用对无间隙高度的另一种一般化，称为前缀感知高度，其定义为块中不同字符串的最大数量，但不计入作为该块中其他字符串前缀的字符串。在图1的示例中，该度量等于2。

- 在第5节中，我们研究用于分割算法的 MSA 预处理。通过深入分析基于去间隙 MSA 行的广义后缀树 $\mathrm{GST}_{\mathrm{MSA}}$ 的现有解，我们将寻找有效片段的预处理算法的时间复杂度从 $O(mn\log n)$ 降至 $O(mn)$。尽管去除间隙会丢失重要信息，但通过考虑其节点或叶节点的适当子集，可以将这些信息反馈到结构中。接下来，主要步骤是

对 MSA 进行预处理时，需要解决 $\mathsf{GST}_{\mathsf{MSA}}$ 的树结构上的一个新型祖先问题，我们称之为排他祖先集问题；我们的一项贡献是识别出这一问题，并给出一个线性时间的解。对于前缀感知高度，有意义的右扩展数量为 $O(mn)$，我们得到一个用时 $O(mn)$ 的解来计算它们。得益于从 MSA 行构建广义后缀树、与之对称的前缀树，以及 Belazzougui 等人的后缀树数据结构提供的两者间常数时间导航（用于回答加权祖先查询） [3]，我们实现了线性时间。

- 最后，在第6节中，我们将介绍一种基于 BWT 的 EFG 索引变体，并通过一组路径 $\mathcal{P}$ 加以增强，以解决类似于文档列举查询的问题 [25], ，即报告哪些路径包含给定模式作为子串。采用这种表述后，例如，可以将模式匹配限制在原始多序列比对的各行内，使该索引在功能上等同于那些用于重复基因组集合文档列举的索引 [8, 7]。

在第7节中，我们总结并讨论研究结果。

2 定义

我们采用 Equi 等人 [14]的记法。

字符串 我们用 $[x..y]$ 表示整数区间。令 $\Sigma = [1..\sigma]$ 为大小为 $|\Sigma| = \sigma$ 的字母表。字符串 $T[1..n]$ 是由 Σ 中的符号构成的序列，记作 $T \in \Sigma^n$；其中，Σ^n 表示由 Σ 上长度为 n 的字符串组成的集合。本文假设，σ 始终小于或等于我们所处理字符串的长度。T 的反转记作 T^{-1}，即将 T 从右向左读取所得的字符串。我们用 $T[x..y]$ 表示 T 的子串，该子串由其从第 x 个到第 y 个字符串接而成。字符串 $T[1..n]$ 的后缀（前缀）为 $T[x..n]$ （$T[1..y]$），其中 $1 \leq x \leq n$ （$1 \leq y \leq n$）；若 $x > 1$（$y < n$），则称其为真后缀（真前缀）。若字符串 S 是字符串 T 的前缀，则记作 $S \preceq T$；若 S 是 T 的真前缀，则记作$S \prec T$。字符串 T 的长度记作 $|T|$，空字符串 ε 是长度为 0 的字符串。特别地，当 $j < i$ 为空字符串时，子串 $T[i..j]$ 即为空字符串。我们用 Σ^* 和 Σ^+ 分别表示 Σ 上的有限字符串集合和有限非空字符串集合。若对于某个区间 $[x..y]$，有 $Q = T[x..y]$ ，则字符串 Q 在 T 中出现；此时，我们称起始位置 x 是 Q 在 T 中的一次出现，而 $y = x + |Q| - 1$ 是该次出现的结束位置。两个字符串 A 和 B 的字典序由字母表的顺序自然定义：当且仅当 $A[1..y] = B[1..y]$ 且对于某个 $y \geq 0$ 有 $A[y+1] < B[y+1]$ 时， $A < B$ 。若 $y + 1 >$ 是 $|A|$、$|B|$ 中较小者，则较短者视为较小。不过，我们通常通过在字符串末尾添加结束标记 \$ 来避免这种隐式比较；\$ 不会出现在任何字符串中，并且按字典序将 \$ 视为最小字符。字符串 A 和 B 的串接记作 $A \cdot B$，也可简记为 AB。

计算模型 下文中，我们假设采用字 RAM 计算模型 [16]。

弹性创始图 多序列比对可以用弹性创始图紧凑地表示；我们将在本段中对这种顶点标记图进行形式化定义。

一个多序列比对MSA$[1..m, 1..n]$ 是一个矩阵，其行是m 长度为 n 的字符串，取自 $\Sigma \cup \{-\}$。这里，$- \notin \Sigma$ 是间隙符号，我们分别用 $[1..m]$ 和 $[1..n]$ 表示行索引集和列索引集。对于字符串 $T \in (\Sigma \cup \{-\})^*$，我们用 spell($T$) 表示从 T中移除间隙符号后得到的字符串。如果一个 MSA 不含间隙，则称其为无间隙；否则，称其为一般MSA。给定 $I \subseteq [1..m]$，我们用 MSA$[I, 1..n]$ 表示仅考虑满足 $i \in I$的行 MSA$[i, 1..n]$所得到的 MSA。

设$\mathcal{P}$ 是 $[1..n]$的划分，即由子区间组成的序列 $\mathcal{P} = [x_1..y_1], [x_2..y_2], \ldots\ldots, [x_b..y_b]$，其中 $x_1 = 1$、$y_b = n$，且对所有 $j > 2$，$x_j = y_{j-1} + 1$。基于划分$\mathcal{P}$ 对 MSA$[1..m, 1..n]$ 进行的分段 S 是由 b 个集合 $S^k = \{\text{spell(MSA}[i, x_k..y_k]) \mid 1 \le i \le m\}$ 构成的序列，其中 $1 \le k \le b$；此外，我们不允许出现对应于 MSA 中一整段间隙的片段。

假设 1. 为了得到 *MSA*$[1..m, 1..n]$的一个恰当分段 $S_1, \ldots, S_b$，我们要求对于任意 $k \in [1..b]$，$\varepsilon \notin S_k$ 。

我们将集合 S^k 称为一个块，并将 MSA$[1..m, x_k..y_k]$ 或简称 $[x_k..y_k]$ 称为 S的一个片段，因为在本文其余部分中，S 总是默认与某个划分 $\mathcal{P}$ 和 MSA$[1..m, 1..n]$相关联。于是，片段 $[x..y]$的长度就是 $y - x + 1$。块 S^k 的高度为 $H(S^k) := |S^k|$；对于给定的片段 $[x..y]$，我们用 H(MSA$[1..m, x..y]$) 或简称 $H([x..y])$ 表示相应块的高度，即其片段高度。

无间隙多序列比对的分段自然引出通过块图概念定义创始图。

定义 1 (块图)。块图是一个图 $G = (V, E, \ell)$，其中 V是节点集合，$E \subseteq V \times V$ 是边集合，而 $\ell : V \to \Sigma^+$ 是一个函数，为每个节点分配一个非空字符串标签。此外，还必须满足以下性质：

1. 集合 V可以划分为一系列 b块 $V^1, V^2, \ldots, V^b$，也就是说，$V = V^1 \cup V^2 \cup \cdots \cup V^b$且对于所有 $i \neq j$，$V^i \cap V^j = \emptyset$；*2.* 所有边都连接相邻的块，也就是说，如果 $(v, w) \in E$，那么对于某个 $1 \le i \le b - 1$，$v \in V^i$ 且$w \in V^{i+1}$；*3.* 一个块中标签的长度相等，也就是说，如果 v，$w \in V^i$，那么 $|\ell(v)| = |\ell(w)|$；如果 $v \neq w$，那么对于每个 $1 \le i \le b$，$\ell(v) \neq \ell(w)$。

如果只有一个块，也就是说 $b = 1$，因此 $E = \emptyset$，我们称 G 是平凡的。

块 S^k 等于片段多序列比对$[1..m, x_k..y_k]$，而创始图是由分段 S [22]诱导的块图。其思路是构建一张图，其中节点表示 S 中的字符串，而边则保留这些字符串如何重新组合以拼写出原始多序列比对中的任意序列的信息。或者，在第6节中，我们将介绍一种索引数据结构，用于仅针对预先定义的一组路径查询该图。

对于一般多序列比对，我们考虑以下推广。

定义 *2*（弹性块图和创始图）。如果块图的第三个条件放宽为每个 V^i 都可以包含非空的可变长度字符串，我们就称该块图为弹性块图。弹性创始图（*EFG*）是由分段 S 按如下方式诱导的弹性块图 $G(S) = (V, E, \ell)$：对于每个 $1 \le k \le b$，都有 $S^k = \{spell(MSA[i, x_k..y_k]) \mid 1 \le i \le m\} = \{\ell(v) : v \in V^k\}$。当且仅当存在块 $k \in [1..b-1]$ 和行 $i \in [1..m]$ 满足 $v \in V^k$, $w \in V^{k+1}$，且 $spell(MSA[i, x_k..y_{k+1}]) = \ell(v)\ell(w)$ 时，$(v, w) \in E$。

例如，在图1的一般多序列比对 $[1..4, 1..13]$ 中，基于划分 $[1..4]$、$[5..9]$、$[10..14]$ 进行分段后，会得到一个 EFG $G(S) = (V^1 \cup V^2 \cup V^3$、$E$、$\ell)$，其中 V^1、V^2 和 V^3 中的节点标签长度可变。正如 Equi 等人 [14], 所指出的，块图与广义退化字符串 [1] 相关，而弹性创始图与弹性退化字符串 [4]相关。

根据定义，（弹性）创始图和块图都是无环的。按照惯例，我们将边的方向解释为从左向右。考虑 G 中任意两个节点之间的一条路径 $P = v_1 \cdots v_k$，其中我们将长度为 k 的路径定义为由边连接的 k 个顶点组成的序列，即 (v_1, v_2)、$(v_2, v_3) \ldots, (v_{k-1}, v_k) \in E$。$P$ 的标签 $\ell(P) := \ell(v_1) \cdots \ell(v_k)$，是路径上各节点标签的串接。设 Q 为一个查询字符串。如果对于 $G(S)$ 的任意路径 P，Q 都是 $\ell(P)$ 的子串，我们就称查询字符串 Q 出现在 $G(S)$ 中。在这种情况下，为简便起见，我们说 Q 作为 P 的子串出现在 $G(S)$ 中，意即它是 $\ell(P)$ 的子串。

最后，我们可以介绍使 EFGs 索引能够进行模式匹配的关键性质。

定义 3 ([22])。如果每个 $\ell(v)$（$v \in V$）在 $G(S)$ 中仅作为以 v 开始的路径的前缀出现，则 *EFG* $G(S)$ 是无重复的。

定义 4 ([22])。如果每个 $\ell(v)$（$v \in V$）在 $G(S)$ 中仅作为以 $w \in V$ 开始的路径的前缀出现，且 w 与 v 来自同一个块，则 *EFG* $G(S)$ 是半无重复的。

例如，图1中的 EFG 不是无重复的，因为 `AGC` 作为同一个块中两个不同节点标签的前缀出现；但它是半无重复的，因为所有满足 $v \in V^k$ 的节点标签 $\ell(v)$ 在 $G(S)$ 中都只从块 V^k 开始出现，否则就不会在图中的其他位置出现。请注意，在无间隙情形下，或者更一般地说，如果 EFG 是非弹性的，那么无重复和半无重复这两个概念是等价的。

基本工具 集合中字符串的字典树或关键词树 [9] 是一棵有根有向树，其中每个节点的出边都标记有不同的符号，并且集合中的每个字符串都对应一条唯一的根到叶路径，其拼写即为该字符串；两条根到叶路径的公共部分拼写出对应字符串的最长公共前缀。在紧凑字典树中，字典树中最长的非分支路径会变为一条边，其标签是该路径上各标签的串接。$T \in \Sigma^*$ 的后缀树是字符串 $T' = T\mathbf{0}$ 所有后缀的紧凑字典树，其中 $\mathbf{0} \notin \Sigma$ 是一个终止符。在这种情况下，边标签是 T 的子串，可以用一个区间以常数空间表示。假设 $\sigma \le |T|$，这种树占用线性空间，并且可以在线性时间内构建，因此从左到右读取叶节点时，后缀会按字典序排列

顺序 [36, 15]。如果后缀树的两个或更多叶节点在字典序中相邻，我们就称它们相邻。这些叶节点构成后缀数组$\mathsf{SA}_{T'}[1..|T'|]$，其中 $\mathsf{SA}_{T'}[i] = j$ 当且仅当 $T'[j..|T'|]$ 是字典序中第 i 小的后缀 [24]。在一组 m 字符串 [18]上构建的树称为广义后缀树或数组。在这种情况下，上文的字符串 T 是在每个字符串[1], 后附加唯一结束标记 $\$_i$ 后，将这些字符串串接而成的，其中 $1 \le i \le m$。将*Burrows–Wheeler* 变换应用于字符串 T时， [5] 会生成另一个字符串 BWT_T，使得 $\mathsf{BWT}_T[i] = T'[\mathsf{SA}_{T'}[i] - 1]$，其中 T'循环回绕，即 $T'[-1] = T'[|T| + 1] = \$$。

设 $Q[1..m]$ 为一个查询字符串。如果 Q 出现在 T中，那么 Q 在 T的后缀树中的轨迹点 或隐式节点 为 $(v,\ k)$，使得 $Q = XY$，其中 X 是从根节点到 v 的父节点的路径所拼写出的字符串，而 Y 是从 v 的父节点到 v的边上长度为 k 的前缀。以 v为根节点的子树中的叶节点，或被 v覆盖的叶节点，都是共享公共前缀 Q 的后缀。设 aX 和 X 分别是从后缀树的根节点到节点 v 和 w 的路径所拼写出的字符串。于是，可以存储一条从 v 指向 w的后缀链接 。对于后缀树，加权祖先查询 要求给定 x 和 y，计算文本子串 $T[x..y]$ 对应的隐式节点或显式节点。

来自二进制字母表的字符串 $B[1..n]$ 称为位向量。$\mathsf{rank}_1(B,\ i)$ 或简写为 $\mathsf{rank}(B,\ i)$，表示 $B[1..i]$中 1 的个数；类似地，$\mathsf{rank}_0(B,\ i)$ 返回 $B[1..i]$中 0 的个数。操作 $\mathsf{select}(B,\ j)$ 返回索引 i ，该索引包含 B中第 j 个 1。使用一个可在线性时间内构建的索引，可在常数时间内回答这两种查询；除位向量本身 [19]外，该索引所需空间为 $o(n)$ 位。

3 半无重复 EFG 的索引

我们的目标是对 EFG G 进行预处理，使我们能够按照 Equiet 等人以下结果的方式，在 $O(|Q|)$时间内检查查询字符串 Q 是否出现在 G中。

定理 1 ([14, 定理 8])。 半无重复 *EFG* $G = (V, E, \ell)$ 可以在多项式时间内建立索引，形成一种占用 $O(|D|$对数$|D|)$ 比特空间的数据结构，其中 $|D| = (NH^2)$，N 是节点标签的总长度，而 H 是 G的高度。之后，可以在 $O(|Q|)$ 时间内查明给定的查询字符串Q是否出现在 G中。

这一结果基于字符串$D = \sum_{(u,\ v)、(v,\ w)\in E}((\ell(u)\ell(v)\ell(w))^{-1}\mathbf{0}$，而我们现在证明，只需基于 G的边构建索引即可。我们首先将每条边 $(v,\ w) \in E$的边标签 $\ell(v)\ell(w)$ 连接起来，形成字符串

$$D_F = \prod_{(v,w)\in E} (\ell(v)\ell(w))\mathbf{0} \qquad \text{and} \qquad D_R = \prod_{(v,w)\in E} (\ell(v)\ell(w))^{-1}\mathbf{0}\$^{|\ell(v)|}\mathbf{0}$$

其中 $\mathbf{0}$,$\$ \notin \Sigma$ 和 $\$^{|\ell(v)|}$ 是重复出现的美元符号，其数量与字符串 $\ell(v)$ 中的字符数相同。稍后便会明白，为什么要用一元编码来表示字符串的长度。现在，我们构造 D_F 和

[1]就我们的用途而言，拼接后字符串的后缀树在功能上等同于图 5 中所示的“修剪后”广义后缀树。此外，我们只需使用一个唯一终止符$\mathbf{0}$。

D_R。匹配算法考虑三种可能的情形：匹配至多跨越两个节点、恰好跨越三个节点，或至少跨越四个节点。如果可以排除前一种情形，就考虑下一种情形。

3.1 一次匹配最多跨越两个节点

我们在 D_R 的后缀树中，对 Q^{-1} 运行标准的模式匹配查询；如果查询成功，就能找到与查询字符串 Q 匹配的内容，且该匹配最多跨越两个节点。如果查询失败，也就是说，Q^{-1} 不是 D_R 的子串，那么我们知道，如果存在匹配，它就至少跨越三个节点。

要找到这些其他潜在匹配，首先，我们需要知道在到达块边界之前，查询中已匹配的字符数 k：

- 如果查询因不匹配而失败，且存在一条标记为**0**的分支，那么 k 是目前已匹配的 Q^{-1} 前缀的长度；
- 如果查询因不匹配而失败，且不存在标记为 **0** 的分支，那么我们沿着目前已匹配的路径回溯到最后一个内部节点；从该节点出发，存在一条以 **0** 开头的向下路径。在这种情况下，k 是截至该内部节点已匹配的 Q^{-1} 前缀的长度。

换句话说，k 是 Q 的最长后缀的长度，该后缀与 EFG 中至多跨越两个节点、且从某个节点标签的起始位置（即块边界）开始的一条子路径相匹配。

如果不存在这样的节点，也就是说，没有节点标签以 Q的最后一个字符开头，那么我们就知道不存在匹配。此外，如果到目前为止我们还没有匹配到完整的节点标签，也同样不存在匹配。为了判断是否处于这种情况，我们可以在构建时标记所有通过至少包含一个完整节点标签的路径到达的内部节点。另一种方法是，从 D_R 的后缀树中回溯到的节点出发，检查是否能为 $1 \le s \le k$读取任意字符串 $\mathbf{0\$}^s\mathbf{0}$ 。如果此测试结果为否，那么根据 D_R 的构造，我们没有遇到完整的节点标签，并且 Q 不会出现在 G中。在第6节中，我们将提出另一种基于后缀数组的方法，通过关注不以其他节点标签为前缀的节点标签 $\ell(v)$，来判断我们是否读取了完整的节点标签。

令 $j = |Q| - k$。到目前为止，我们已在 D_R 中为 $Q^{-1}[1..k]$ 找到一个匹配，该匹配至少跨越一个完整的节点标签。半无重复性质限制了 $Q[j+1..|Q|]$ 的出现必须从某个块 V^{r-1} 开始。该字符串可以跨越块 V^{r-1} 和 V^r 之间的多条边，甚至可以作为 V^{r-1} 中节点标签的前缀出现，如图2和下一个引理所示。

引理 1. 令 Q 为一个字符串，使得 Q 至少有一个后缀在半重复自由 *EFG* $G = (V, E, \ell)$ 中存在匹配，且该匹配至少跨越三个块。令 V^r 为某次在 G 中出现的 Q 所含匹配字符所在的最右侧块，并考虑后缀 $S = Q[|Q| - j + 1..|Q|]$，它与某个 $v \in V^{r-1}$、$w \in V^r$ 所对应的 $\ell(v)\ell(w)$ 的前缀匹配，其中 $1 \le k \le |Q|$ 和 $2 \le j \le |Q|$。后缀 S 只能作为某个 $v' \in V^{r-1}$、$w' \in V^r$ 所对应的 $\ell(v')\ell(w')$ 的前缀出现。

证明。考虑字符串 $R = \ell(v')\ell(w')$，其中 $v' \in V^{r-1}$、$w' \in V^r$ 为任意值，并用反证法假设 $S = R[k_1...k_2]$，其中 $1 < k_1 < k_2 \le |R|$ 为任意值；也就是说，S匹配的并非 $\ell(v')\ell(w')$ 的前缀。于是，标签 $\ell(v)$ 是$R[k_1...k_2]$ 的前缀，却不是 R 的前缀，这与 G 是半无重复的假设相矛盾。 □

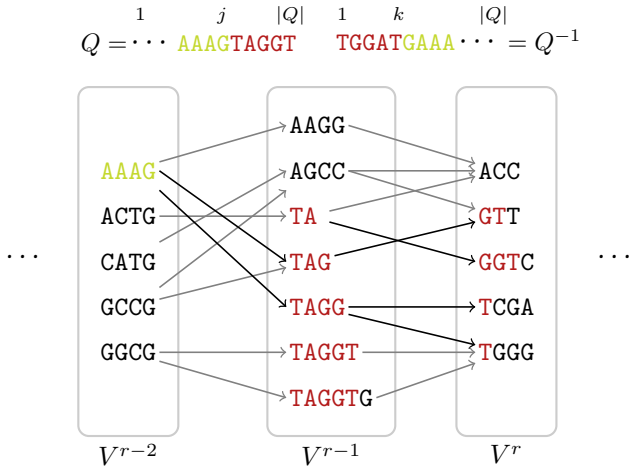

图2：字符串 Q 的后缀匹配的最后三个块。我们用红色突出显示最后 $k = |Q| - j$ 个字符及其匹配项，而青柠色字符则标记了倒数第二个块中的唯一匹配项。

3.2 一次匹配恰好跨越三个节点

我们已在 D_R 中匹配了 $Q^{-1}[1..k]$，并找到了 V^{r-1} 对应的左边界。根据引理 1，我们知道该边界在 EFG 中是唯一的位置。现在，我们只考虑在 V^{r-1} 中完全匹配、在 V^r 中部分匹配的 $Q^{-1}[1..k]$ 的出现位置，并标记 V^{r-1} 中所有被此次匹配命中的节点，也就是所有满足以下条件的 $v_{\mathrm{R}} \in V^{r-1}$：$\ell(v)$ 是 $Q[j+1..|Q|]$ 的真前缀，并且 v_{R} 有一个出邻居，其标签部分拼写了 Q 的剩余后缀。我们需要逐一考虑这些节点，因为正如图2所示，它们可能与 V^{r-2} 中拼写模式其余部分的某个节点相连，也可能不相连。

一方面，注意到这些节点彼此都是前缀，因此它们至多有 $|Q^{-1}[1..k]| - 1 = O(|Q|)$ 个。另一方面，EFG 的拓扑结构限制很少，我们不希望探索所有这些节点的入邻域，因为那样会花费超过 $O(|Q|)$ 的时间。相反，我们可以利用在 D_R 中每条边之后以一元编码的数字。事实上，考虑从 D_R 中对应于 $Q^{-1}[1..k]\mathbf{0}$ 的节点出发，读取 \$ 个字符的路径。不难看出，在分解为 $\ell(v_{\mathrm{R}}) \cdot \ell(w)[1..k - |\ell(v_{\mathrm{R}})|]$ 的 EFG 中，存在 $Q[j+1..|Q|]$ 的一次出现，且满足 $v_{\mathrm{R}} \in V^{r-1}$ 和 $w \in V^r$，当且仅当 $Q^{-1}[1..k]\mathbf{0}\$^{|\ell(v_{\mathrm{R}})|}\mathbf{0}$ 是 D_R 的一个子串。因此，只需检查在 D_R 中的 $Q^{-1}[1..k]\mathbf{0}$ 之后能否读出 $\$^p\mathbf{0}$，即可标记所有节点；对于 $1 \le p \le |Q|$，这只需沿着上述路径向下遍历即可完成。

然后，如果我们想找到一个恰好跨越三个节点的 Q 匹配项，就在 D_F 的后缀树中搜索 Q，并定位节点 $v_{\mathrm{F}} \in V^{r-1}$，使得存在 $u \in V^r$ 满足 $(u, v_{\mathrm{F}}) \in E$, $\ell(u) = Q[1..j]$，且 $\ell(v_{\mathrm{F}}) = Q[j+1..j+|\ell(v_{\mathrm{F}})|]$。利用我们已知的信息—— $Q[j+1..|Q|]$ 从块 V^{r-1} 开始——这次正向搜索比之前的反向搜索更简单：根据半无重复性质，当且仅当 $Q[1..j]\ell(v_{\mathrm{F}})\mathbf{0}$ 出现在 D_F 中时，Q 才会在 G 中匹配，其中 v_{F} 等于先前在反向搜索中标记的某个 v_{R}。

反向搜索。我们可以轻松地预处理 D_F 的后缀树，以定位节点 v_{F}，因为半无重复性质保证，对于 $T \in \Sigma^*$，字符串 $T\ell(v)\mathbf{0}$ 对应子树中的任何叶节点都能唯一确定任意一个 $v \in V$。因此，我们通过简单地沿着 D_F 的后缀树向下遍历，查询 $Q[1..j] \cdot Q[j+1..j+1+s] \cdot \mathbf{0}$ 中的 $0 \le s < k$，并检查对应的节点 v_{F} 是否已标记。

请注意，即使 $Q^{-1}[1..k]$匹配两个完整的节点标签，上述过程也能正确运行。事实上，如果 Q 的一个后缀与 V^{r-1}中的某个完整节点标签匹配，且该节点有一条边连接到 V^{r-2}中另一个完全匹配的节点标签，那么我们在 D_R中首次搜索 Q^{-1} 时，就会在 V^{r-2} 中找到这个节点标签；随后便可执行刚才描述的过程，让 V^{r-1}充当 V^r，并让 V^{r-2} 充当 V^{r-1}。

3.3 匹配至少跨越四个节点

如果到目前为止我们还没有找到完全匹配，就意味着即使存在匹配，它也必定至少跨越四个节点。考虑发现位置 j的那次对 D_R 的首次搜索。在该次搜索结束时，我们确定了 D_R 的后缀树中的一个特定内部节点，该节点对应于 $Q[j+1..|Q|]$。从该内部节点出发，有一条向下路径，起始字符为 0，后面接着一串 \$ 字符。根据前一种情况，我们知道已经标记了节点 $v_{\mathrm{R}} \in V^{r-1}$，它们与 $Q[j+1..|Q|]$的一个前缀匹配，且最多有 O个（$|Q|$）。令 $\hat{}v$ 为标签长度最短的已标记节点 $|\ell(\hat{}v)|$。现在我们在 D_R中搜索 $Q^{-1}[k-|\ell(\hat{}v)|+1..|Q|]$。这样，我们就能找到唯一的节点 u，使得对于某个 $1 \le j' \le j$，$Q[j'..j] = \ell(u)$ 完全匹配，并且 $(u,v) \in E$ 满足 $\ell(\hat{}v)$ 是 $\ell(v)$ 的前缀。请注意，我们已经排除了匹配跨越三个块的可能性，因此 $Q[j'..j]$ 必须与 $\ell(u)$ 完全匹配，而不只是与其某个后缀匹配。此外还要注意，如果 $Q[j+1..|Q|]$ 与 V^{r-1} 中某个节点的前缀完全匹配，且该节点有一条边连接到某个 v_{R}，那么我们在最初进行的搜索中就会找到这个匹配。既然事实并非如此，我们就需要执行以下操作。我们扫描每个已标记节点 $v_{\mathrm{R}} \in V^{r-1}$，并检查边 (u,v) 是否存在：若存在，则后缀 $Q[j'+1..|Q|]$ 有一个跨越块 V^{r-2}、V^{r-1}和 V^r的匹配。为执行这次扫描，我们按照上一段中对节点 v_{F}所采用的相同方式，在 D_F中搜索 $0 \le s < k$对应的 $\ell(u) \cdot Q[j+1..j+1+s] \cdot \mathbf{0}$；每当找到一个匹配时，我们就检查 V^{r-1} 中对应的节点是否已标记。[2]

此时，从该块向左，我们可以像之前一样使用 D_R 的后缀树，在前一个块中继续匹配；始终将当前节点的标签与 Q 中尚待匹配的前缀拼接起来。之所以可行，是因为除最后一个块外，左侧的每个块在 $|Q|$中都只有一个匹配的节点标签，如下一个引理所示。

引理 2 ([14, 引理 9])。*考虑一个半无重复 EFG $G=(V,E)$。字符串$\ell(v)\ell(w)$，其中$(v,w) \in E$，只能在 G 中作为以 v开头的路径的前缀出现。*

[2]请注意，我们明确地识别出节点 u 属于G中 Q 的任意剩余匹配。Equi 等人的原始解会为每个节点 v的出邻域标签构建相应的字典树，也会为每个块的字符串（反转后）构建字典树；我们可以在此处利用它们进行导航。不过，也可以用 D_R 和 D_F 以类似方式导航 G。

证明。 反设 $\ell(v)\ell(w)$ 是一条从某个 $v' \in V, v' \neq v$ 的标签内部开始的路径的前缀。那么 $\ell(v)$ 是这条路径的前缀，而这只有在 v' 与 v 位于同一个块，且 $\ell(v)$ 是 $\ell(v')$ 的真前缀时才可能；否则， G 就不是半无重复的。于是， $|\ell(v)| < |\ell(v')|$ 和 $\ell(w)$ 出现在一条从标签 $\ell(v')$ 内部开始的路径中。这与 G 是半无重复的这一事实矛盾。 □

因此，一旦确定位置 j，根据引理 2，所有以 $Q[j'..|Q|]$ 为前缀的路径都必须以 v 开始，因为 $Q[j'..j] = \ell(v)$；对于在 Q 中完全拼写出来的所有边，同样的论证也成立。在 D_R 中进行的反向搜索能够在最后一个块中完成匹配，甚至可能在到达 D_R 的叶节点之前就匹配完 Q 中的所有字符。

因此，在以线性时间构造出 D_R 的后缀树后，我们的搜索对 $|Q|$ 中的每个位置至多考察三次，所以可以在 $O(|Q|)$ 时间内完成在 G 中匹配 Q。另外请注意，在反向 D_R 搜索和正向 D_F 搜索中，块 V^{r-1} 中标记的节点都可以由一个 Q 范围唯一确定，因此我们可以将后缀树用作黑盒索引，以便在 D_F、 D_R 中匹配模式；对于算法的这部分，我们则可以在大小为 $|Q|$ 的位向量中标记 Q 的位置。

定理 2. 半无重复 *EFG* $G = (V, E, \ell)$ 可以在多项式时间内编入数据结构中，该数据结构占用 $O(|D|\log|D|)$ 位空间，其中 $|D| = O(NH)$， N 是节点标签的总长度， H 是 G的高度。之后，可以在 $O(|Q|)$ 时间内判断给定的查询字符串 Q 是否出现在 G中。

4 最优 EFG 的构造

本节回顾并扩展 EFG 构造算法的理论。回想一下，在没有间隙的情况下，半无重复和无重复这两个概念（定义 3 和 4）是等价的，因为任意块中的字符串长度都相同。事实上，在证明半无重复 EFG 易于建立索引、可实现快速模式匹配之后，Equi 等人 [14] 将先前的结果扩展到了无间隙情形，并证明半无重复 EFG 等价于 MSA 的特定分段：只需根据 MSA 检验半无重复性质，而不必检查最终的 EFG。我们在第 4.1 节回顾这些论证，并介绍由此得到的递推式，用于根据三种评分函数计算最优分段： i。最大化块数； ii。最小化片段的最大长度；以及 iii。最小化块的最大高度。

在无间隙且无重复的情形下，得分 i. 和 ii. 可在 $O(mn)$ 时间内构造可索引创始图，这得益于此前关于创始图和多序列比对分段的研究 [22, 28, 6]。在第 4.4 节中，我们结合这些结果，得到一个同样适用于得分 iii. 的 $O(mn)$ 时间解：通过首先计算有意义的左扩展来找出最优分段，也就是$\cdots$这些位置 $x_1 > > x_k$，在这些位置上，无重复片段 $[x_i..y]$ 的高度会增加，且 $y \in [1..n]$。

在一般情形和半无重复情形中，将片段向左扩展可能会违反半无重复性质，并且高度可能降低。因此，Equi 等人在 [14] 提出了分别针对得分 i 和 ii 的 $O(n)$ 时间和 $O(n \log \log n)$ 时间算法，这些算法利用了公共 {style id='9'>右扩展 之后的半无重复性质

$O(mn\log\ m)$ 时间的多序列比对预处理，我们将在第 4.2 节回顾该方法。在第 4.3 节中，我们将得分 $ii.$ 的构造算法改进至 $O(n)$ 时间；在第5节中，我们将预处理改进至 $O(mn)$，从而达到整体线性时间。

在第4.5节中，我们针对得分 $iii.$，开发了一种类似的算法，用于构造最优的半无重复分段，并处理有意义的右扩展。尽管这些扩展的总数为 $O(n^2)$，我们仍设法给出一种参数化线性时间的解，其上界取决于最长游程的长度；在该游程中，任意两行所拼写的字符串都满足一个是另一个的前缀。另一种高度概念——前缀感知高度——则会生成 $O(mn)$ 个有意义的前缀感知右扩展：处理这些扩展的方式与处理原始高度相同，从而得到最优分段；我们将在第5节介绍如何高效计算它们。

4.1 可索引创始图的分段刻画

在适用时，我们将结合讨论无重复性质和半无重复性质（定义 3 和 4），并称之为（半）无重复性质。不过，这种括号式的表述并非总是适用，因为我们关注含缺口的多序列比对，而且我们的多数结果仅适用于半无重复性质。

考虑一个分段 $S = S^1, S^2, \ldots, S^b$，它根据定义 2 诱导出一个（半）无重复 EFG $G(S) = (V, E, \ell)$。图$G(S)$中出现的字符串是原始多序列比对行中出现的字符串的超集，因为每个节点标签可以表示多行，而每条边 $(v, w) \in E$ 表示在相应的连续片段中存在某一行，其拼写为 $\ell(v)\ell(w)$。例如，字符串 GACTAGT 出现在图1的 EFG 中，但它并未出现在原始多序列比对的任何一行中。

（半）无重复性质涉及图 $G(S)$，但幸运的是，它不依赖于创始图中新增的这些字符串，只需根据多序列比对和分段 S 进行检查即可。直观地说，这是因为新增的字符串涉及 $G(S)$ 的三个或更多顶点。这简化了选择能够生成可索引创始图的分段的过程；这一结论最初由 M¨akinen 等人在无间隙且无重复的情形下证明。

引理3 (刻画，无间隙情形 [22])。 如果某个无间隙 多序列比对$[1..m, 1..n]$ 中的片段 $[x..y]$是无重复的，那么字符串多序列比对$[i, x..y]$ 仅在某一行的 x 位置出现在该多序列比对中，对所有 $1 \le i \le m$均成立。于是， $G(S)$ 是无重复的，当且仅当 S 的所有片段都是无重复的。

Equi 等人在 [14] 中将这一性质推广到含缺口的多序列比对，但没有提供明确的证明。由于这一性质对于构造算法的正确性至关重要，我们在此给出证明。

引理 *4*（刻画 [14]）。如果一般多序列比对$[1..m, 1..n]$ 的片段 $[x..y]$ 满足：对于任意 i， $i' \in [1..m]$ 字符串 ***spell(MSA***$[i, x..y]$***)*** 在去间隙行 ***spell(MSA***$[i', 1..n]$***)*** 中只在位置 $g(i', x)$出现，其中$g(i', x)$等于 x 减去 ***MSA***$[i', 1..x]$中的间隙数，则称该片段为半无重复。同样，如果 ***spell(MSA***$[i, x..y]$***)*** 在行 i' 的位置 $g(i', x)$出现时，也结束于位置 $g(i', y)$，则称 $[x..y]$为无重复。那么， $G(S)$是（半）无重复，当且仅当 S 的所有片段都是（半）无重复。

证明。为方便起见，如果一个片段或创始图是（半）无重复的，我们就称其为有效；否则称其为无效。此外，我们定义标准字符串在 $G(S) = (V, E, \ell)$中出现的如下概念。如果 P使用路径 $P = w_1 \cdots w_k$的所有顶点来拼写 S，则称 $S \in \Sigma^+$为 $G(S)$中路径的标准子串，这意味着

$$S = \ell(w_1)\big[j..|\ell(w_1)|\big] \cdot \ell(w_2) \cdots \ell(w_{k-1}) \cdot \ell(w_k)\big[1..j'\big]$$

其中 $1 \le j \le |\ell(w_1)|$ 和 $1 \le j' \le |\ell(w_k)|$。我们还称，S 在 P 中的出现涉及 $G(S)$ 的 k 个顶点。

我们通过证明两个方向的逆否命题，并利用以下事实来完成证明：

1. 片段 $[x..y]$ 无效，当且仅当存在 $i, i' \in \{1, \ldots, m\}$，使得字符串 spell(MSA$[i, x..y]$) 在行 spell(MSA$[i', 1..n]$) 中某个位置出现，该位置不是 $g(i', x)$；或者字符串 spell(MSA$[i, x..y]$) 是字符串 spell(MSA$[i', x..y]$) 的真前缀（对于半无重复情形，忽略最后这个条件）；

2. 创始图 $G(S)$无效，当且仅当存在节点 $v \in V$，使得 $\ell(v)$ 是图 $G(S)$中某条路径 $P = w_1 \cdots w_k$的标准子串，且满足以下条件之一：w_1 与 v位于不同的块中；$\ell(v)$ 在$\ell(P)$中某个非第 1 的位置出现；或者 $k = 1$ 和 $\ell(v)$ 是 $\ell(P) = \ell(w_1)$的真前缀（对于半无重复情形，忽略最后这个条件）。

显然，根据 $G(S)$的构造，只有无重复情形才有的附加无效性条件（事实 1. 和 2. 中的最后几个条件）是等价的，因此我们着重讨论与半无重复情形共有的条件。

($\Rightarrow$) 设 $[x..y]$ 是 S的一个无效片段，其字符串拼写 spell(MSA$[i, x..y]$) 出现在行 i'的某个位置 j，且该位置不同于 $g(i', x)$，其中 $i, i' \in \{1, \ldots, m\}$；再设 $v \in V$ 是与片段 $[x..y]$ 对应的块中的节点，满足 $\ell(v) =$ spell(MSA$[i, x..y]$)。若 $g(i', x) < j \le g(i', y)$，则 $\ell(v)$出现在$\ell(P)$中的位置 $j - g(i', x) \ne 1$，其中 P是从 v所在的同一个块出发的一条路径；否则，$j < g(i', x)$ 或 $j > g(i', y)$，且 $\ell(v)$出现在 $G(S)$的一条路径中，该路径从一个与 v所在块不同的块中的节点出发。在这两种情况下，$G(S)$都是无效的。

($\Leftarrow$) 若 $G(S)$ 无效，则设 $\ell(v)$ 是某条路径 $P = w_1, \ldots, w_k$ 上的标准子串，该路径属于使创始图无效的 $G(S)$，其中某个 $v \in V$。按照第5节中的相同论证，若 [22, Section 5.1], if $k \le 2$ then $\ell(P)$ 是输入多序列比对中某一行的子串，使 S 无效，因为根据 $G(S)$ 的构造，对于每条边 $(u, u') \in E$，都有 $\ell(u)\ell(u')$ 出现在多序列比对中。否则$k > 2$，这意味着 $\ell(v)$ 经由 P 的出现涉及至少三个顶点，并且对于某个 $w \in \{w_2, \ldots, w_{k-1}\} \subseteq V$, $A, B \in \Sigma^+$，有 $\ell(v) = A\ell(w)B$。但此时，$\ell(w) \in \Sigma^+$ 出现在 $\ell(v)$ 中的位置不是 1，因此存在行索引 $i, i' \in \{1, \ldots, m\}$，使得 $\ell(w) =$ spell(MSA$[i, x..y]$) 出现在 spell(MSA$[i', 1..n]$) 中的位置不是 $g(i', x)$；其中 $[x..y]$ 是 S 中与 w 的块对应的片段，且 spell(MSA$[i', 1..n]$) 包含$\ell(v)$，从而使片段 $[x..y]$ 无效。 □

4.2 EFG 构造算法

与无间隙、无重复情形一样，引理 4 表明，对于多种评分方案，一般多序列比对前缀 MSA$[1..m, 1..j]$ 的（半）无重复分段的最优得分 $s(j)$ 可以递归计算：

$$s(j) = \bigoplus_{\substack{j' : 0 \le j' < j \text{ s.t.} \\ \mathsf{MSA}[1..m, j'+1..j] \text{ is} \\ \text{(semi-)repeat-free}}} E\big(s(j'), j', j\big) \tag{1}$$

其中，算子 $\bigoplus$ 和函数 E 取决于所需的评分方案。具体而言：

i。若要使 $s(j)$ 等于块数最多的分段的最优分数，则设置 $\bigoplus = \max$ 和 $E(s(j'), j', j) = s(j') + 1$；若要正确初始化，则设置 $s(0) = 0$；如果不存在（半）无重复分段，则设置 $s(j) = -\infty$；

ii。对于最大分段长度最小的分段，设置 $\bigoplus = \min$ 和 $E(s(j'), j', j) = \max(s(j'), L([j'+1, j]))$ $= \max(s(j'), j - j')$；设置$s(0) = 0$；如果不存在（半）无重复分段，则设置 $s(j) = +\infty$。

iii。对于最大块高度最小的分段，设置 $\bigoplus = \min$ 和 $g(s(j'), j', j) = \max(s(j'), W([j'+1, j]))$；设置 $s(0) = 0$；如果不存在（半）无重复分段，则设置 $s(j) = +\infty$。

Equi 等人 [14] 研究了半无重复分段的计算，并针对得分 i. 和 ii进行了优化。他们开发的算法——我们将在第 4.3 节和第 5 节对其进行改进——基于对有效的半无重复分段范围进行共同的预处理，该预处理基于以下观察。

观察 1 (半无重复右扩展 [14])。给定字母表为 $\Sigma \cup \{\dashv\}$ 上的多序列比对$[1..m, 1..n]$，对于任意 $x < y$，我们称片段 $[x+1..y]$ 是多序列比对$[1..m, 1..x]$前缀的扩展。如果扩展 $[x+1..y]$ 是半无重复的，那么对于所有 $y < y' \le n$，扩展 $[x+1..y']$ 都是半无重复的。

请注意，在存在间隙的情况下，如果将半无重复概念替换为无重复概念，或者将右扩展替换为对称定义的左扩展，则观察 1 不成立。

为了计算 $s(j)$，公式 (1) 会考虑所有以列 j 结尾的半无重复右扩展 $[j'+1..j]$。Equi 等人发现，可以通过以下方式高效地计算值 $s(j)$：考虑每个半无重复右扩展 $[j'+1..j]$ 都以一个最小的（半无重复）右扩展$[j'+1..f(j')]$，为前缀，其函数 f 定义如下。

定义5 （最小右扩展 [14]）。给定 多序列比对$[1..m, 1..n]$，对于每个$0 \le x \le n-1$，我们将值 $f(x)$ 定义为大于 x 的最小整数，使得片段 $[x+1..f(x)]$ 是半无重复的；换言之，$[x+1..f(x)]$ 是前缀 多序列比对$[1..m, 1..x]$的最小（半无重复）右扩展。如果不存在半无重复扩展，则定义 $f(x) = \infty$。

[3]在无间隙情形下，片段长度与所得 EFG 块中字符串的长度一致。一般情形下则并非如此，片段长度是所得块中最大节点标签长度的上界。

事实上，Equi 等人在 [14] 中开发了一种算法，可在 $O(mn \log m)$ 时间内计算值 $f(x)$。仅使用这些值（由按第二个分量递增排序的对列表 $(x, f(x))$ 描述），他们开发了两种算法来计算最优半无重复分段的得分：计算最大块数得分的算法耗时 $O(n)$，计算最大块长度得分的算法耗时 $O(n \log \log n)$。我们将在第 4.3 节详细说明后者的工作原理，并将其运行时间改进为 $O(n)$。

4.3 最小化最大块长度

我们将在第5节中把一般多序列比对情况下最小右扩展的计算从 $O(nm \log m)$ 改进到 $O(nm)$，这促使我们改进 Equi 等人的 $O(n \log \log n)$ 时间算法 [14, ，该算法用于求解使最大块长度最小化的最优半无重复分段。正如第 4.2 节所述，我们可以通过处理与所有满足 $f(x) \leq j$ 的右扩展（x, $f(x)$）相对应的递归解，来计算 $s(j)$。对于最大块长度，MSA 的最优解在以半无重复片段 $[x+1..j']$ 作为最后一个片段时，有两种递归类型：

非领导者递归： 如果 $j' \leq x + s(x)$，那么 $s(j')$ 的得分等于 $s(x)$，因为片段 $[x+1..j']$ 的长度小于或等于 $s(x)$；在这种情况下，我们称 $[x+1..j']$ 为一个非领导片段；

主导递归： 否则，如果 $j' > x + s(x)$，我们称 $[x+1..j']$ 为一个领导片段，因为在约束最后一个片段必须为它的最优解中，它的得分为 $j' - x$ 。

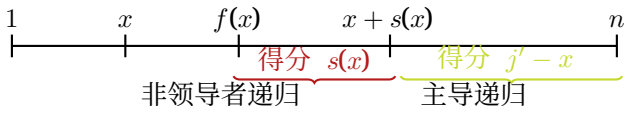

图 3：MSA$[1..m, 1..j']$ 的最优半无重复分段在约束最后一个片段为 $[x+1..j']$ 时的得分示意图。

注意，如果 $x + s(x) < f(x)$，那么对于 $(x, f(x))$，不会发生非领导者递归。于是，很容易看出

$$s(j) = \min \left(\min_{\substack{(x,f(x)):\\ f(x) \leq j \leq x+s(x)}} s(x), \min_{\substack{(x,f(x)):\\ j > f(x) \wedge j > x+s(x)}} j - x \right) \tag{2}$$

因此，Equi 等人通过使用两棵一维搜索树跟踪两类递归，正确地解决了该问题：第一棵树跟踪得分为 $s(x)$ 的范围 $[f(x)..x + s(x)]$；第二棵树跟踪必须使用主导递归的范围 $[x + s(x) + 1..n]$，并且仅保存得分 $j - x$ 的 $-x$ 部分。通过分别针对范围 $[j + 1.. + \infty]$ 和 $[-\infty..j]$ 执行两次半无限区间最小值查询，我们可以通过动态规划计算 $s(j)$，并在 $O(n \log \log n)$ 时间内解决该问题。

相反，借助以下观察，我们可以使用更简单的数据结构达到线性时间复杂度：

- 主导递归的数据结构可以替换为单个变量 S，其值为 $\min\{j-x:\ j>f(x) \wedge j>x+s(x)\}$，这样 S就是以领导片段 $[x+1..j]$结尾的分段的最佳得分；

- 对于非领导者递归，我们可以将 Equi 等人的结构替换为一个等价的数组 $\mathtt{C}[1..n]$，使得 $\mathtt{C}[k]$ 统计使用非领导者递归且得分为 k 的可行解数量，从而变量$K=\min\{k:\ \mathtt{C}[k]>0\}$ 等于以非领导片段 $[x+1..j]$结尾的分段的最佳得分。

最后一个关键观察是，这两种递归类型密切相关：当 $[x+1..j]$ 从非领导片段变为领导片段时，也就是 $j=x+s(x)+1$，我们将 $\mathtt{C}[s(x)]$ 减一，并在必要时将 S 更新为值 $s(x)+1=j-x$。因此，当以这种方式移除 $\mathtt{C}[1..n]$ 的最佳得分时，我们无需将 K 更新为 $\min\{k:\ \mathtt{C}[i]>0\}$，只需将 K 加 1，即可确保 $s(j)=\min(K,S)$，除非对 $\mathtt{C}$ 和 S 的其他更新得出更好的得分。最终的解已在算法 1 中实现。

定理 3. 给定（x、f（x））的多序列比对$[1..m,1..n]$的最小右扩展，我们可以在 O（n）时间内计算出使最大块长度最小化的最优半无重复分段的得分。

证明。 算法 1 的正确性由 [14, 算法 2]的正确性以及以下事实可知：当 $\mathtt{C}[K]=0$ 时，对于 $1\le j'\le K$和 $S\le K+1$，有 $\mathtt{C}[j']=0$。类似地，处理最小右扩展（x，f（x））以及动态管理区间 $[f$（x）$..s$（x）$+j']$ 总共需要 O（n）时间，因此该算法的运行时间为线性时间。 □

可以轻松修改算法 1，使其显式地计算出一个最优分段。事实上，在算法 2 的主循环中，对于每个 $\mathtt{C}[i]$值$\mathtt{backtrack}_{\mathtt{C}}[i]$，我们可以保留最大的 x，使得 $[x+1..j]$ 产生得分为 i 的非领导者递归解。类似地，我们可以维护值 $\mathtt{backtrack}_S$，其等于最大的 x，使得 $[x+1..j]$ 产生得分为 S 的领导者类型最优解。然后，对于每个 s（j），我们可以计算值 $\mathtt{backtrack}[j]$，其等于最大的 x，使得 $[x+1..j]$ 用于得分为最优得分s（j）的解中。此外，结合第 5.3 节的定理 7，我们得到以下结果。

推论 1. 给定多序列比对$[1..m,1..n]$，来自 $\Sigma\cup\{-\}$，且有 $\Sigma=[1..\sigma]$ 和 $\sigma\le mn$，构造最优半无重复分段以最小化最大块长度的过程可在 $O(mn)$时间内完成。

4.4 在无间隙设置下最小化最大高度

对于无间隙多序列比对，Norri 等人 [28] 针对块长度受给定下界 L限制的情况，提出了一种用于构造分段、以最小化最大块高度的 $O(mn)$ 解，而非要求满足无重复性质。该结果基于以下假设： Σ 是大小为 $O(m)$ 的整数字母表。本节将 Norri 等人的算法与 M¨akinen 等人 [22], 计算值 $v(j)$ 的方法相结合——我们将这些值称为最小左扩展——从而得到一个线性时间解

算法 1： 用于求半
无重复分段的最优得分，目标是使最大块长度最小。操作——
K.Update 和 S.Update 在输入值较小时，
会用输入值替换变量的当前值。

输入： 最小右扩展（x_1，$f(x_1)$）$\ldots$，（x_n，$f(x_n)$），按第二个分量从小到大排序

输出： 最优半无重复分段的得分，目标是最小化最大块长度

将数组 $\mathtt{C}[1..n]$ 初始化为取值范围为 $\{0,\ldots,n\}$ 的数组，并将所有值设为 0；
将数组 $\mathtt{L}[1..n]$初始化为空链表，其值为指向（x，$f(x)$）；
$\mathtt{minmaxlength}[0] \leftarrow 0$;
$y \leftarrow 1$；$K \leftarrow 1$；$S \leftarrow \infty$;
对于 $j \leftarrow 1$ **到** n **执行**
  当 $j = f(x_y)$ **时执行** ▷ 处理最小右扩展
    如果 $j \le x_y + \mathtt{minmaxlength}[x_y]$ **则** ▷ 非领导者递归
      $\mathtt{C}[\mathtt{minmaxlength}[x_y]] \leftarrow \mathtt{C}[\mathtt{minmaxlength}[x_y]] + 1$;
      K.更新($\mathtt{minmaxlength}[x_y]$);
      $\mathtt{L}[x_y + \mathtt{minmaxlength}[x_y] + 1]$.add$((x_y,\ f(x_y)))$;
    else 。 主导递归
      S.更新($j - x_y$);
    $y \leftarrow y + 1$;
  对于 $(x, f(x)) \in \mathtt{L}[j]$ **执行** ▷ 将非领导者记录更新为领导者记录。
    $\mathtt{C}[\mathtt{minmaxlength}[x]] \leftarrow \mathtt{C}[\mathtt{minmaxlength}[x]] - 1$;
    S.更新($j - x$); ▷ $j - x = s(x) + 1$
  如果 $\mathtt{C}[K] > 0$ **则**
    $\mathtt{minmaxlength}[j] \leftarrow \min(K, S)$;
  else
    $\mathtt{minmaxlength}[j] \leftarrow S$;
  $S \leftarrow S + 1$； 。 更新数据结构以供下一次迭代使用
  如果 $\mathtt{C}[K] = 0$ **则**
    $K \leftarrow K + 1$;
返回 $\mathtt{minmaxlength}[n]$;

用于构造最大块高度最小的无重复创始图。

观察 2（左扩展的单调性 [28, 12]）。给定一个无间隙多序列比对$[1..m, 1..n]$，对于任意 $1 \le x \le y \le n$，我们称 $[x..y]$ 是后缀多序列比对$[1..m, y+1..n]$的一个左扩展。那么：

- 如果 $[x..y]$ 是无重复的，那么对于所有 $x' < x$，$[x'..y]$ 也是无重复的；
- 对于所有 $x' < x$，$m \ge H([x'..y]) \ge H([x..y])$。

因此，对于每个 $j \in [1..n]$，我们将值 $v(j)$ 定义为不大于 j 且使 $[v(j)..j]$ 无重复的最大列索引，并称 $v(j)$ 或$[v(j)..j]$ 是多序列比对$[1..m, j+1..n]$的最小左扩展。如果不存在有效的左扩展，那么 $v(j) = -\infty$。

定义 6（有意义的左扩展 [28, 12]）。设 多序列比对$[1..m, 1..n]$ 是一个无间隙多序列比对。对于任意 $j \in [1..n]$，我们用 $L_j = \ell_{j,1}, \ldots, \ell_{j,c_j}$ 表示多序列比对$[1..m, j+1..n]$的有意义（无重复）左扩展，即所有小于或等于 j 的位置构成的严格递减序列，满足：

1. $\ell_{j,c_j} < \cdots < \ell_{j,2} < \ell_{j,1} = v(j)$，因此 L_j 包含多序列比对$[1..m, j+1..n]$的所有无重复左扩展；*2.* $H([\ell_{j,k}..j]) > H([\ell_{j,k}+1..j])$ 对于 $2 \le k \le c_j$，因此每个 $\ell_{j,k}$ 都标记一个左扩展高度增加的列；由观察 *2* 可知 $|L_j| = c_j \le m$。

如果多序列比对$[1..m, j+1..n]$ 没有无重复左扩展，我们定义 $L_j = ()$ 和$c_j = 0$。否则，为求完整，我们定义 $\ell_{j,c_j+1} = -1$。

在得分 $iii.$ 下，公式 (1) 可利用 $L_j = \ell_{j,1}, \ldots, \ell_{j,c_j}$ 重写如下：

$$s(j) = \min_{k \in [1..c_j]} \max\left(\min_{j' \in [\ell_{j,k+1}+1..\ell_{j,k}]} s(j'),\quad H\big([\ell_{j,k}..j]\big) \right) \tag{3}$$

以及 $s(j) = +\infty$ 如果 $c_j = 0$，因此，对于 $k \in [1..c_j]$，数值 L_j、$H([\ell_{j,k}..j])$ 和 $\min_{j' \in [\ell_{j,k+1}+1..\ell_{j,k}]} s(j')$ 可用于在 $O(m)$ 时间内计算 $s(j)$。一方面，给定固定长度 L，Norri 等人 [28] 针对定义 6 的一个变体开发了一种算法，该变体考虑长度至少为L的片段（而非无重复片段），并可在总计 $O(mn)$ 时间内计算这些数值。另一方面，M¨akinen 等人 [22] 开发了一种线性时间算法，用于计算无间隙多序列比对的数值 $v(j)$。将后者用于求得数值 $v(j)$，并将这些数值用作最小接受片段长度的动态下界，即可结合这两种解。由于我们在第4.5节中为一般情形开发的算法也能解决这个问题，只需使用对称定义的右扩展，因此本文不再描述这种修改。

定理4。给定一个来自大小为 O 的整数字母表 Σ 的无间隙多序列比对$[1..m, 1..n]$ $m()$，可以在时间 $O(mn)$内计算出使最大块高度最小化的多序列比对$[1..m, 1..n]$ 的最优无重复分段。

4.5 重新审视右扩展的线性时间解

对于含缺口的多序列比对，在半重复自由概念下，左扩展的单调性（观察 2）不成立 [14, 表 1]: 固定 $j \in [1..n]$ 时，从 $x = v(j)$ 向后进行的左扩展多序列比对 $[1..m, x..y]$ 并不总是半无重复的，也就是说，并不总是有效；而且，有效片段在扩展时，其高度可能降低。例如，在图1的多序列比对中，片段 $[5..9]$ 是半无重复的，但片段 $[4..9]$ 不是，并且 $H([5..9]) < H([6..9])$。本节解决这两个问题中的前一个：我们开发一种利用右扩展的算法，以与 [14, 算法 1 和 2] 以及算法 1 相同的方式，从左向右计算最优多序列比对分段。我们将在第4.6节讨论计算这些右扩展的复杂度。

回顾有效右扩展的性质以及值 $f(x)$的定义（观察 1 和定义5）。

定义 7 (有意义的右扩展)。 给定一般的 多序列比对 $[1..m, 1..n]$，对于任意 $x \in [0..n-1]$，我们用 $R_x = r_{x,1}, \ldots r_{x,d_x}$ 表示多序列比对 $[1..m, 1..x]$的有意义（半无重复）右扩展，即所有大于 x 的位置构成的严格递增序列，使得：

- $f(x) = r_{x,\,1} < r_{x,\,2} < \cdots < r_{x,\,d_x}$，从而 R_x 涵盖多序列比对 $[1..m, 1..x]$的所有半无重复右扩展；
- $H([x+1..r_{x,k}]) \neq H([x+1..r_{x,k}-1])$ 对于 $2 \le k \le d_x$，因此每个 $r_{x,\,k}$都标记一个右扩展高度发生变化的列。

如果多序列比对 $[1..m, 1..x]$ 没有半无重复右扩展，那么 $R_x = ()$ 且 $d_x = 0$。否则，为完整起见，我们定义值 $r_{x,d_x+1} = n+1$。

由于我们将把所有 $R_0, \ldots, R_{n-1}$ 一并处理，因此我们为每个值 $r_{x,k}$ 补充列 x 以及对应多序列比对片段的高度，从而得到三元组 $(x, r_{x,k}, H([x+1..r_{x,k}]))$。

因此，在得分为 iii 的条件下，公式 (1) 可改写如下：

$$s(j) = \min_{\substack{x \in [0..j-1],\, k \in [1..d_x]\,: \\ r_{x,k} \le j < r_{x,k+1}}} \max\Big(s(x),\ H\big([x+1..r_{x,k}]\big)\Big). \tag{4}$$

由于每个 R_x 都定义了关于 $[1..n]$ 的不重叠区间 $[r_{x,k}..r_{x,k+1}-1]$，在计算 $s(j)$ 时，每个 R_x 至多涉及一个满足 $x < j$ 的区间 $[r_{x,k}..r_{x,k+1}-1]$，相应的得分取决于 j 所在的区间。此外，请注意，公式（4）比公式（3）更简单。最后，计算使最大块高度最小化的最优半无重复分段得分的算法，在算法 2 中进行了描述；该算法通过处理 $R_0, \ldots, R_{n-1}$ 中所有有意义的右扩展来实现，这些扩展表示为三元组 (x, r, h)，并按第二个分量从小到大排序。

主要策略是在每次迭代中，保留以右扩展 $[1..j]$、$[2..j]$、.....或 $[j..j]$ 结尾的 MSA 半无重复分段的最佳得分 j；这些分段由 R_0、R_1、.....或 R_{j-1} 中的范围描述。逐一检查当前有效的每个范围会导致二次时间解法，因此我们需要用其他形式表示这些范围。事实上，通过使用数组 $\mathtt{C}[1..m]$ 统计这些得分，使得 $\mathtt{C}[i]$ 等于得分为 i 的可用解的数量，就可以通过找出最小的 i 来计算得分 $s(j)$

算法 2：用于求解使最大块高度最小化的半无重复分段的最优分数的主算法。若输入为空集，min 操作返回 $+\infty$。输入：有意义的右扩展 $(x_1, r_1, h_1), \ldots, (x_k, r_k, h_k)$，按第二个分量从小到大排序。输出：使最大块高度最小化的最优半无重复分段的得分。1 初始化数组 $\mathtt{R}[0..n-1]$，其值为 $[0..m] \cup \{\perp\}$，并将所有值设为 $\perp$；2 初始化数组 $\mathtt{C}[1..m]$，其值为 $[0..m]$，并将所有值设为 0；3 $y \leftarrow 1$；4 $\mathtt{minmaxheight}[0] \leftarrow 0$；5 对 $j \leftarrow 1$ 到 n 执行 6 当 $j = r_y$ 时执行 7 如果 $\mathtt{R}[x_y] \neq \perp$，则 8 $\mathtt{C}[\mathtt{R}[x_y]] \leftarrow \mathtt{C}[\mathtt{R}[x_y]] - 1$；. 移除 R_{x_y} 9 $s \leftarrow$ 的最后一个解 $\max(\mathtt{minmaxheight}[x_y], h_y)$；10 $\mathtt{R}[x_y] \leftarrow s$；. 保存对应于 (x_y, r_y, h_y) 的得分 11 $\mathtt{C}[s] \leftarrow \mathtt{C}[s] + 1$；. 添加对应于 (x_y, r_y, h_y) 的解 12 $y \leftarrow y + 1$；13 $\mathtt{minmaxheight}[j] \leftarrow \min\{i \in [1..m] : \mathtt{C}[i] > 0\}$；

14 **返回** $\mathtt{minmaxheight}[n]$;

使得 $\mathtt{C}[i]$ 大于零。只有当 j 到达某个 r_x、k 时，才需要更新数组 C；换言之，当对于某个 (x, r, h)，$j = r$ 时，必须将以 $[x+1..j]$ 结尾的最优分段的得分 $\max(s(x), h)$ 加到 C，并从中移除相对于 R_x 的前一个范围的旧得分。我们可以用数组 $\mathtt{R}[0..n-1]$ 跟踪这些得分，使得 $\mathtt{R}[x]$ 等于与当前有效扩展 R_x 相关联的得分。要计算实际分段而非仅计算其得分，可以使用两个回溯数组 $\mathtt{backtrack}[1..n]$ 和 $\mathtt{backtrack_C}[1..m]$：$\mathtt{backtrack_C}[i]$ 等于某个 x，其中 $[r_{x,k}..r_{x,k+1}-1]$ 是当前有效且得分为 i、并具有最大 $r_{x,k+1}$ 的范围；$\mathtt{backtrack_C}$ 的值可用于计算 $\mathtt{backtrack}[j]$，它等于某个 x，其中 $[x+1..j]$ 是 $\mathrm{MSA}[1..m, 1..j]$ 的某个最优解的最后一个片段，且具有最大 $r_{x,k+1}$。然后，backtrack 可重建一个最优分段。只要给每个有意义的右扩展 $(x, r_{x,k}, H([x+1..r_{x,k}])$ 附加值 $r_{x,k+1} - 1$，就可以轻松修改算法 2 来更新这些数组。

引理 5. 给定多序列比对 $[1..m, 1..n]$ 及其有意义的右扩展 $R_0, R_1, \ldots, R_{n-1}$，我们可以在时间 $O(mn + R)$ 内计算使最大块高度最小的最优半无重复分段，其中 $R := \sum_{x=0}^{n-1} |R_x|$。

证明。正确性由公式（4）及上述论证得出。按第二个分量对有意义的右扩展（x，r，h）排序，可在 $O(n + R)$ 时间内完成，因为有意义的右扩展的取值范围为 $[1..n]$。此外，数组 R 和 C 的管理对每个有意义的右扩展只需常数时间，而从 C 计算每个 $s(j)$ 需要

$O(m)$ 时间，达到 $O(mn + R)$ 的时间复杂度。 □

对于无间隙情形，这是第 4.4 节所述方法的一种替代解法，因为可以使用 $R \in O(mn)$ 以及 Norri 等人和 Equi 等人的算法来计算有意义的右扩展。

4.6 一般情形下最小化最大块高度

如引理 5 所述，我们可以处理定义 7 中的有意义的右扩展，以计算使最大块高度最小化的最优分段得分。不幸的是，在存在间隙的一般情形下，有意义的右扩展总数为 $O(n^2)$：如图 4 所示，如果从同一列 x 开始的任意两个行后缀拼写出相同的字符串，但其拼写被间隙交错打断，那么片段 $[x..y]$ 的高度可能在任意列 y 发生变化；这种模式可能涉及一般多序列比对 $[1..m, 1..n]$ 中任意片段里的任意两行，因此在最坏情况下，二次上界是紧的。

	1	2	3	4	5	6	7		$n-2$	n
1	T	A	–	A	–	A	–	...	A	– C
2	T	–	A	–	A	–	A		–	A C
$H([1..y])$	1	2	1	2	1	2	1	$\cdots$	2	1 1

图 4：多序列比对 $[1..2, 1..n]$ 的示例，使得 $|R_0| \in \Theta(n)$。

观察 3。 给定字母表为 $\Sigma \cup \{–\}$ 的多序列比对$[1..m, 1..n]$，只有当存在行 i、$i' \in [1..m]$，使得 多序列比对$[i, y{+}1] = –$、多序列比对$[i', y+1] = c$，且 拼写(多序列比对$[i, x..y]$) =拼写(多序列比对$[i', x..y+1]$) $= S \cdot c$，其中 $c \in \Sigma$ 且 $S \in \Sigma^+$ 时，才有 $H([x..y]) > H([x..y{+}1])$。

图 4 的示例以及观察 3 所描述的情境直观上似乎不切实际，因为高分多序列比对会尝试对齐各行，以避免出现这种情况。尽管如此，在不对多序列比对中读取相同字符串的部分所含间隙作进一步假设的情况下，我们仍需计算所有有意义的右扩展。具体而言，令 $K_{x,y}$ 为字符串集合$S_{x..y} :=$ {拼写(多序列比对$[i, x..y]$)：$1 \le i \le m$}的关键词树；由于 $|S_{x..y}| = H([x..y])$，$[x..y]$ 的高度等于 $K_{x..y}$ 中对应于 $S_{x..y}$ 内字符串的不同节点数。注意到如果$K_{x..y}$ 有 m 个叶节点，那么 $S_{x..y}$ 中任意两个字符串都不是彼此的前缀，并且对于所有 $y' > y$，$H([x..y']) = m$，由此可得一个参数化解。

引理 6. 给定一个基于大小为$\sigma \in O(mn)$的整数字母表 $\Sigma \cup \{–\}$ 的一般多序列比对 $[1..m, 1..n]$，我们用 α 表示满足以下条件的任意片段$[x..y]$ 的最大长度 $y - x + 1$：对于某些i、$i' \in [1..m]$，拼写(多序列比对$[i, x..y]$) 是拼写(多序列比对$[i', x..y]$) 的前缀。于是，我们可以在$O(mn\alpha \log \sigma)$时间内计算出所有有意义的右扩展。

证明。 对于每个 $x \in [0..n-1]$，我们可以通过增量计算树 $K_{x+1..x+1}$、$K_{x+1..x+2}$、.....、$K_{x+1..n}$ 来找到 R_x，使用支持从根节点到叶节点的遍历以及向任意节点 c 插入一个子节点的动态关键词树 $\mathcal{T}$

节点 v，其中 $c \in \Sigma$。算法 3 描述了该过程的一种可能实现。在计算过程中，数组 $\mathrm{v}[1..m]$ 跟踪与字符串 spell(MSA$[i, x+1..r]$) 对应的节点，每个变量 $v.$count 统计读取相应字符串的行数，而变量 h统计满足 v.count 大于零的不同节点 v 的数量：如果 h 发生变化，那么相应的有意义的 $[x+1..r]$ 的右扩展为 $(x,\ r,\ h)$。对于每个 R_x，如果 $\mathcal{T}$的叶节点数量为 m，算法就会停止；也就是说，它最多增量计算 α 棵关键词树。得益于观察 3 和上述论证，不会遗漏任何有意义的右扩展。我们可以在 $O(mn\alpha \log \sigma)$ 时间内计算 R_0、.....、R_{n-1}，因为 $\mathcal{T}$的遍历和插入操作可以在 $O(\log \sigma)$[4] 时间内实现，而其他操作则可以在常数时间内完成。

□

由于有意义的右扩展总数为 $O(mn\alpha)$，引理 5 和 6 给出了我们分段问题的以下解。

定理 5. 给定字母表大小为O的整数字母表 Σ 上的一般多序列比对多序列比对 $[1..m, 1..n]$，我们可以在 $O(mn\alpha\log\ \sigma)$时间内计算使最大块高度最小化的最优分段得分，其中 α 是最长多序列比对片段的长度；在该片段中，任意两行拼写出的字符串分别为 S和 S'，且 S 是S'的前缀。

最坏情况下，有意义的右扩展数量为 $O(mn^2)$,$\alpha \in \Theta(n)$，而引理 6 的时间复杂度为 $\Theta(mn^2 \log \sigma)$。因此，我们将块高度从无间隙情形推广到一般情形时，引入另一种不同的定义。

定义 8 (前缀感知高度)。给定多序列比对$[1..m, 1..n]$，我们将片段 $[x..y]$ 的前缀感知高度定义为集合 $\{$spell(多序列比对$[i, x..y]$) : $1 \le i \le m\}$ 中不同字符串 S 的数量，记作 $\overline{H}$(多序列比对$[1..m, x..y]$)，或简写为 $H([x..y])$；其中满足条件的字符串是指 S 不是该集合中其他字符串的前缀。

由于 $\overline{H}([x..y])$ 等于 $H([x..y])$ 减去在 $[x..y]$ 中拼写出来、且是该片段中其他字符串真前缀的字符串数量，因此这种细化高度始终小于或等于原始高度：相对最优分段为原始情形下的最大高度提供了下界。此外，观察 3 中所述的高度降低必要条件不再成立，而且很容易看出，前缀感知右扩展的单调性依然成立（见观察 2）。事实上，如果我们按照定义 7 定义有意义的前缀感知右扩展 R_0、.....、R_{n-1}，很容易看出，对于所有 $x \in [0..n-1]$，都有 $|R_x| \le m+1$，因此这些扩展的总数为 $O(mn)$。

定义 9（有意义的前缀感知右扩展）。给定字母表 $\Sigma \cup \{\$\}$ 上的多序列比对$[1..m, 1..n]$，对于任意 $x \in [0..n-1]$，我们用 $R_x = r_{x,\ 1},\ \cdots r_{x,\ d_x}$表示多序列比对$[1..m, 1..x]$ 的有意义的前缀感知右扩展，其定义同定义 7，只是将片段高度 H 替换为前缀感知高度 $\overline{H}$。

最后，给定 R_0、...、R_{n-1} 作为输入，算法 2 能正确计算最优分段的得分（依据我们定义的细化高度），因为公式（4）

[4]由于只需要插入，且字母表 Σ 固定，因此可以用高度至多为 $\lceil \log_2 \sigma \rceil$ 的动态二叉树来实现为每个节点 v 添加一个子节点；随着子节点数量增加，这棵树会向下生长。

算法 3： 计算所有有意义右扩展的算法

$R_0, \ldots, R_{n-1}$。为高效计算 $y \in [x+$ 的关键词树 $K_{x+1..r}$ $1..n]$，我们需要一种支持导航的动态树数据结构 $\mathcal{T}$ 并能在 $O(\log \sigma)$ 时间内执行插入操作。

输入： 来自大小为 $\Sigma \cup$ 的整数字母表 $[1..m, 1..n]$ 的多序列比对$\{-\}$ $\sigma \in O(mn)$，针对 $x \in [0..n-1]$ 的最小右扩展 $f(x)$。

输出： 有意义的前缀感知右扩展 R_0，……，R_{n-1} 表示为三元组 $(x, r_{x,\,k},\ H([x+1..r_{x,k}]))$。

```
1 对 x ← 0 到 n − 1 执行
2     初始化空关键词树 T，其中仅包含根节点;
3     初始化数组 V[1..m]，其中的值为指向 T 中节点的指针，并将所有
        值设为根节点;
4     h ← 0;
5     r ← x;
6     当 T 叶节点 < m ∧ r ≤ m 时
7         h′ ← h;
8         对于 i ← 1 到 m，执行
9             如果 多序列比对[i, r] ≠ −，则
10                V[i].计数 ← V[i].计数 − 1;
11                如果 V[i].计数 = 0 则
12                    h ← h − 1;
13                如果 V[i] 有一个(多序列比对[i, r])子项 v 则
14                    V[i] ← v;
15                else
16                    将新的 (多序列比对[i, r])子项 v 添加到 V[i];
17                    V[i] ← v;
18                如果 V[i].计数 = 0 那么
19                    h ← h + 1;
20                V[i].计数 ← V[i].计数 + 1;
21        如果 r = f(x) 那么
22            输出 (x, r, h);
23        否则如果 r ≥ f(x) ∧ h ≠ h′ 则
24            输出 (x, r, h);
25        r ← r + 1;
```

仍然成立。在第5节中，我们将提供一种基于去间隙 MSA 行的广义后缀树的算法，可在线性于 MSA 大小的时间内计算前缀感知扩展，从而得到以下结果。

定理6. 给定字母表大小为 $\sigma \leq mn$ 的整数字母表上的 多序列比对$[1..m, 1..n]$ *{–}*，计算使最大前缀感知块高度最小的半无重复分段需要 $O(mn)$ 时间。

5 MSA 预处理以进行分段

本节研究最小右扩展 $f(x)$（定义5）和有意义的前缀感知右扩展 $R_{\underline{x}}$（定义9）对 $0 \leq x \leq n-1$ 的计算。我们的目标是分别在 $O(n)$ 和 $O(mn)$ 时间内计算它们。对于前者，Equi 等人在 [14] 中提出了一种 $O(nm\log\ m)$ 时间的解，使用以下由去间隙 MSA 行构建的数据结构。

定义 10. 给定字母表 $\Sigma \cup$*{–}* 上的 多序列比对$[1..m, 1..n]$，我们将 $\mathsf{GST}_{\mathsf{MSA}}$定义为字符串集合 *{*拼写(多序列比对$[i, 1..n]) \cdot \$_i$: $1 \leq i \leq m$*}* 的广义后缀树，其中 $\$_1$、.....、$\$_m$ m 是 Σ 中不存在的、彼此不同的新终止符号。[5]

图 5 给出了 $\mathsf{GST}_{\mathsf{MSA}}$ 的一个示例。根据后缀树的性质，对于任意去间隙行 S_i := spell(MSA$[i, 1..n])\$_i$，其中 $1 \leq i \leq m$：每个后缀 $S_i[x..|S_i|]$ 都与 $\mathsf{GST}_{\mathsf{MSA}}$ 的唯一一个叶节点 $_{,x}$ 相对应，反之亦然，其中 $1 \leq x \leq |S_i|$；每个子串 $S_i[x..y]$ 都与 $\mathsf{GST}_{\mathsf{MSA}}$ 中根节点到 ℓ_i 的路径上的一个显式或隐式节点相对应$_{,x}$；每个显式或隐式节点也都与一个或多个这样的子串相对应，这些子串可根据该节点所覆盖的叶节点唯一确定。此外，请注意，$\mathsf{GST}_{\mathsf{MSA}}$ 不包含任何关于多序列比对中的间隙符号的信息，因为这些信息将借助所考虑的叶节点和节点集合重新添加到结构中。

在第5.1节中，我们对 $\mathsf{GST}_{\mathsf{MSA}}$ 进行了与 Equi 等人类似的分析，表明多序列比对中的半无重复片段对应于 $\mathsf{GST}_{\mathsf{MSA}}$ 中恰好覆盖 m 个叶节点的一组特定节点。随后，在第5.2节中，我们证明了 $\mathsf{GST}_{\mathsf{MSA}}$树结构上由此产生的新问题——我们称之为独占祖先集问题——可以高效求解，从而得到一种能够在线性时间内计算最小右扩展的算法；我们将在第5.3节介绍这一解。此外，在第5.4节中，我们将这些技术扩展，证明由专属祖先确定的 $\mathsf{GST}_{\mathsf{MSA}}$ 内部森林能够描述有意义的前缀感知右扩展：通过为这些森林中的每个节点计算一个位置，指示相关字符串的首次多序列比对出现结束的位置，并对这些位置排序，我们就能在 $O(m^2n)$ 全局时间内计算有意义的右扩展。最后，在第5.5节中，我们将介绍如何借助去间隙行的广义前缀树 以及后缀树节点与前缀树节点之间的映射，高效地计算这些位置。该映射可在线性时间内计算，方法是应用词缀树或词缀数组 [21, 34], ，或者使用 Belazzougui 等人的加权祖先查询数据结构

[5]为简化起见，我们添加了 m 新的不同终止符；而 Equi 等人使用了所有去除间隙的行串接而成的字符串的后缀树，并在每行之间插入一个新的符号 $。如果在该字符串末尾串接第二个唯一终止符 #，则该字符串的后缀树就与 $\mathsf{GST}_{\mathsf{MSA}}$ 在我们的用途上等价。

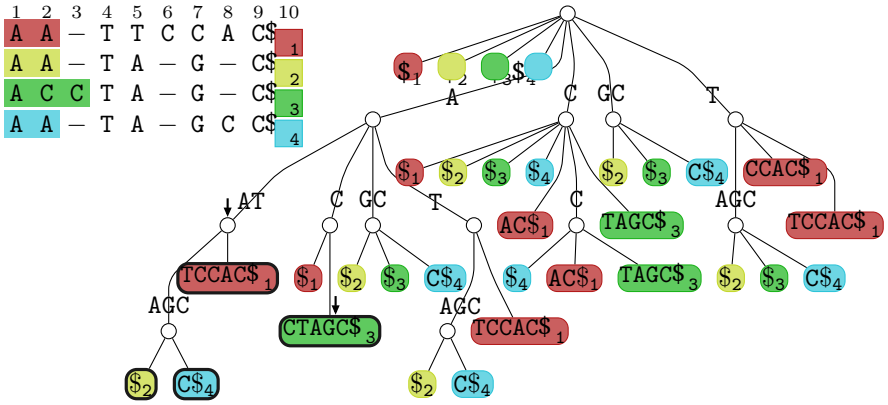

图 5：一个多序列比对$[1..4, 1..10]$ 及其广义后缀树$_{\mathsf{MSA}}$示例，其中每个叶节点的标签都已移至叶节点内部。我们还突出显示了与后缀拼写(MSA$[i, 1..n]$对应的叶节点（黑色轮廓）及其专属祖先节点（箭头）。

，使得我们能够在常数时间内从后缀树导航至前缀树，从而达到全局 $O(mn)$ 时间。

5.1 广义后缀树中的半无重复片段

以下内容已在 [14] 中陈述并加以利用。

定义 11 (半无重复子串)。 回顾半无重复片段的定义（引理 *4*）。给定 MSA $[1..m, 1..n]$ 中的子串 MSA$[i, x..y]$，使得 拼写(MSA$[i, x..y]) \in \Sigma^+$，若对所有 $1 \le i' \le m$ 字符串 MSA$[i, x..y]$，拼写() 仅在去间隙行 i' 的位置 $g(i', x)$ 出现（或完全不出现），则称 MSA$[i, x..y]$ 为一个半无重复子串 。

观察 4. 片段 $[x..y]$ 是半无重复的，当且仅当其所有子串MSA$[i, x..y]$ 都是半无重复的，其中 $1 \le i \le m$。如果MSA$[i, x..y]$ 是半无重复的，那么对于所有 $y < y' \le n$，MSA$[i, x..y']$ 都是半无重复的。回顾最小右扩展 $f(x)$的定义，对于 $0 \le x \le n-1$ （定义*5*）。令 $f^i(x)$为大于 x 的最小整数，使得子串MSA $[i, x+1..f^i(x)]$ 是半无重复的：容易看出， $f(x) =$最大$_{i=1}^{m} f^i(x)$。

这对应于 GST$_{\mathsf{MSA}}$ 中一组特定的隐式节点或显式节点。我们在每一行添加一个唯一终止符号，这等价于在位置 $n+1$ 添加一列 MSA 列，其拼写为 $\$_1 \cdots \$_m$，这意味着 $[x+1..n+1]$ 始终是半无重复的，并且使 $f(x) = \infty$ 变为 $f(x) = n+1$ 的最小右扩展。

引理 7. 给定 m 多序列比对多序列比对$[i, x..y_i]$ 的多序列比对$[1..m, 1..n]$ 行子串，使得拼写(多序列比对$[i, x..y_i]) \in \Sigma^+$ 对于 $1 \le i \le m$，令 $W = \{w_1, \ldots, w_k\}$ 为 GST$_{\mathsf{MSA}}$ 中与字符串 { 拼写(多序列比对$[i, x..y_i]) : 1 \le i \le m\}$ 对应的隐式或显式节点集合。那么，对于所有 $1 \le i \le m$， W 恰好覆盖 GST$_{\mathsf{MSA}}$ 中的叶节点，当且仅当 $[i, x..y_i]$ 多序列比对是半无重复的。

证明。 根据 $\mathsf{GST_{MSA}}$ 的构造， W 覆盖了 m 叶节点 $\ell_{1,z_1},\ldots,\ell_{m,z_m}$，其中 $z_i = g(i,x)$，因此我们只需证明：如果某个 $\mathsf{MSA}[i,x..y_i]$ 不是半无重复的，或者无效，那么 W 覆盖的叶节点多于 m ，反之亦然。

(⇐) 设 $\mathsf{MSA}[i,x..y_i]$ 无效，即对于某一行 $1 \le i' \le m$，$\mathsf{spell}(\mathsf{MSA}[i,x..y_i])$ 在 $S_{i'}$ 中某个位置 z 出现，且该位置不等于 $z_{i'}$。那么，$\mathsf{GST_{MSA}}$ 中与字符串 $\mathsf{spell}(\mathsf{MSA}[i,x..y_i])$ 对应的节点覆盖叶节点 $\ell_{i',\hat{z}} \neq \ell_{i',z_{i'}}$，因此 W 覆盖的叶节点不止 m 个。

(⇒) 设 $\ell_{i',\hat{z}}$ 是 $\mathsf{GST_{MSA}}$ 中的一个叶节点，且不是某个节点 $w \in W$ 所覆盖的叶节点 $\ell_{1,z_1},\ldots,\ell_{m,z_m}$。根据构造， w 对应于某个 $1 \le i \le m$ 的 $\mathsf{spell}(\mathsf{MSA}[i,x..y_i])$，因此 $\mathsf{spell}(\mathsf{MSA}[i,x..y_i])$ 在 $S_{i'}$ 中某个不同于 $g(i',x)$ 的位置出现，因为 $\ell_{i',\hat{z}} \neq \ell_{i',z_{i'}}$。因此，$\mathsf{MSA}[i',x..y_i]$ 无效。 □

请注意，如果将半无重复的概念替换为无重复的概念，引理 7 的正确性便不再成立。

引理 7 与观察 4 结合表明，可以通过分析 $\mathsf{GST_{MSA}}$ 相对于多序列比对后缀的树结构，解决对所有 $i \in [1..m]$ 计算值 $f^i(x)$ 的问题。具体而言，设 $\mathrm{L}_x := \{\ell_{i,z_i} : 1 \le i \le m, z_i = g(i,x+1)\}$ 是 $\mathsf{GST_{MSA}}$ 中与后缀 $\mathsf{spell}(\mathsf{MSA}[i,x+1..n])$ 对应的叶节点。对于每一行 $1 \le i \le m$，$\mathsf{spell}(\mathsf{MSA}[i,x+1..n])$ 的第一个半无重复前缀，对应于 $\mathsf{GST_{MSA}}$ 中从根到 ℓ_{i,z_i} 的路径上第一个隐式或显式节点v ，使得 v 仅覆盖 L_x 中的叶节点。$\mathsf{GST_{MSA}}$ 是一棵压缩字典树，这一点并不会造成问题：后缀字典树中 v 的父节点是分支节点，因为它覆盖的叶节点比 v 更多。因此，$\mathsf{GST_{MSA}}$ 中从根到 ℓ_{i,z_i} 的路径上仅覆盖 L_x 中叶节点的第一个显式节点，就是 v 的第一个显式后代 w ；由此，我们可以通过查找 w 来确定 v 。最后，通过检索最小列索引 y 来计算 $f^i(x)$，使得 $\mathsf{spell}(\mathsf{MSA}[i,x+1..y]) = \mathsf{string}(\mathsf{parent}(w)) \cdot \mathsf{char}(w)$；其中 $\mathsf{string}(u)$ 是根到 u 的路径上边标签的串接，而 $\mathsf{char}(u)$ 是从 $\mathsf{parent}(u)$ 到 u 的边标签中的第一个字符。换言之， y 对应于 MSA 行 i 的第 k 个非间隙符号，且 $k = \mathsf{rank}(\mathsf{MSA}[i,1..n],x) + \mathsf{stringdepth}(\mathsf{parent}(w)) + 1$；其中 $\mathsf{rank}(\mathsf{MSA}[i,1..n],x)$ 是 $\mathsf{MSA}[i,1..x]$ 中非间隙符号的数量，而 $\mathsf{stringdepth}(u) := |\mathsf{string}(u)|$。例如，在图 5 中，$\mathrm{L}_0$ 的叶节点已被标记，其仅覆盖 L_0 中叶节点的最浅祖先也已标记。

5.2 排他祖先集

上一节的结果表明，我们可以通过在 $\mathsf{GST_{MSA}}$ 的树结构上求解以下问题的多个实例，来计算最小右扩展。

问题1 (排他祖先集)。 设 $T = (V, E,$ 根节点) 是一棵有根有序树，其中 $L^T \subseteq V$ 为其叶节点集合。给定 T 以及叶节点的一个子集 $L \subseteq L^T$，求 L 在 T中的专属祖先的最小集合 W ，即满足以下条件的最小集合$W \subseteq V$ ： W 覆盖 L 中的所有叶节点，且仅覆盖 L中的叶节点。能否对 T 进行预处理，以支持高效求解该问题的多个实例?

与 $\mathsf{GST_{MSA}}$ 的情况一样，我们可以假设 T 的每个内部节点至少有两个子节点；否则，可以通过线性时间处理 T 来压缩其单子路径。事实上，对 T 进行线性时间预处理后，任何由 L 和 T 定义的问题1，都可以在 $O(|L|)$ 时间内通过以下方式求解:

按照以下过程仔细遍历这棵树；我们对此作非正式的描述：

1. 将 L 划分为 T 有序遍历中叶节点连续的 k 个极大集合 $L_1, \ldots, L_k$，以便分别处理（如果两个叶节点属于不同的连续集合，则它们的任何共同祖先都不能成为解的一部分）；2. 对于每个 L_i，在 $1 \leq i \leq k$ 的情况下，从最左侧叶节点 ℓ_i 开始，沿树向上遍历，直到找到 ℓ_i 中覆盖某个不属于 L_i 的叶节点的最近祖先；3. 如果步骤 2 失败，则将最后一个安全祖先加入解 W，如果 L_i 中仍有未覆盖叶节点，则从最左侧的未覆盖叶节点开始，重复步骤 2 和 3。

图6展示了该过程的一个示例。步骤 2 的失败条件可以通过检查候选替换节点的子树中最左侧和最右侧的叶节点是否仍属于集合 L_i 来判定。假设 L 是一个非平凡实例，步骤 2 总会终止：如果 $L \subset L^T$，那么 T 的根节点就不是该问题的解。

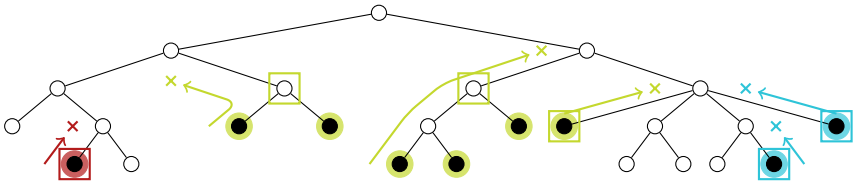

图6：独占祖先集问题的一个实例，其中叶节点集合 L 对应黑色叶节点：算法将 L 划分为若干连续叶节点集合（分别以红色、黄绿色和蓝色叶节点表示），并为每个集合找出其专属祖先（以矩形标记）。每个箭头都表示步骤 2 沿树向上移动，直到到达对应失败条件的节点，该节点以叉号标记。

假设 T 的叶节点已排序，则步骤 1 可以高效实现：我们可以通过对 L 中的叶节点着色，并找出前一个叶节点不在 L 中的所有叶节点，将 L 划分为若干连续叶节点集合。我们可以轻松地对 T 进行预处理，以支持常数时间内执行所需操作；由于构建在叶节点 L 之上的任何森林都包含 $O(|L|)$ 个节点，因此时间复杂度为 $O(|L|)$。

引理8。在有根有序树 $T = (V, E, \text{根})$ 上，给定其叶节点的一个子集 L，可以在 $O(|L|)$ 时间内解决独占祖先集问题；为支持对任意节点 $v \in V$ 执行操作 v.最左叶节点、v.最右叶节点，以及对任意叶节点 $\ell \in L^T$ 执行操作 ℓ.前一叶节点、ℓ.后一叶节点和二元着色，需先进行 $O(|V|)$ 时间的预处理。

5.3 计算最小右扩展

回到计算值 $f(x)$ 的问题，GST_{MSA}的表示需要支持引理8所描述的树结构操作，以及 v.stringdepth 操作。后者返回与显式节点对应的

$\mathsf{GST}_{\mathsf{MSA}}$ 中从根节点到 v 的路径所对应字符串的长度；还需要支持 ℓ.suffixlink 操作，以实现叶节点的后缀链接。最终算法见算法 4，它计算分别对应于从第 1、2、. . . 、 n 列开始的多序列比对后缀的叶集合 $\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_{n-1}$；并对每个满足 $0 \le x < n$ 的 $\mathcal{L}_x$：

1. 它标记$\mathcal{L}_x$ 中的叶节点，并通过找出所有左边界 ℓ （使得 ℓ.prevleaf未被标记），将它们划分为若干连续叶节点集合；2. 它针对每个连续叶节点集合求解独占祖先集问题，并在找到一个覆盖叶节点 $\ell_{i_1}, \ldots, \ell_{i_k}$的专属祖先时，计算 $i \in \{i_1, \ldots, i_k\}$ 的值 $f^i(x)$（参见第5.1节的结论）；3. 处理完所有叶节点后，最后计算 $f(x) = \max_{i=1}^{m} f^i(x)$，并通过仅对满足MSA$[i, x+1] \neq -$ 的叶节点 ℓ_i 取后缀链接[6]，将$\mathcal{L}_x$ 转换为$\mathcal{L}_{x+1}$ 。

定理 7. 给定多序列比对$[1..m, 1..n]$，我们可以在时间(mn)内计算 $0 \le x < n$ *(*定义 *5)*的最小右扩展 $f(x)$ 。

证明。 正确性由观察 4 以及引理 7 和 8 得出。$\mathsf{GST}_{\mathsf{MSA}}$ 的构造等价于构建一个长度小于或等于 $(m+1)\cdot n$的字符串的后缀树：由于我们假设$|\Sigma| \le mn$，因此可以在 $O(mn)$ 时间内构建一棵支持所需操作且操作时间为常数时间的后缀树。此外，我们可以对多序列比对行进行预处理，以便在常数时间内回答关于间隙符号和非间隙符号位置的 rank 和 select 查询。因此，计算每个 $f(x)$ 需要 $O(|\mathcal{L}_x| + m) = O(m)$ 时间，所以总时间为 $O(mn)$。

□

推论 2。 给定来自 $\Sigma \cup \{-\}$ 的多序列比对$[1..m, 1..n]$，其中 $\Sigma = [1..\sigma]$ 且 $\sigma \le mn$，可以在 $O(mn)$ 时间内构造一个使块的最大数量最小化的最优半无重复分段。

证明。 算法 [14, Algorithm 1] 由 Equi 等人提出，可在 $O(n)$ 时间内解决该问题，前提是已按第二个分量递增排序给出最小右扩展（x, $f(x)$)。借助定理 7，我们现在可以在 $O(mn)$ 时间内计算并排序这些扩展。

□

5.4 计算前缀感知右扩展

给定 $\mathsf{GST}_{\mathsf{MSA}}$ 及其叶节点 $\mathcal{L}_x$ （对应于从列 $x+1$ 开始的后缀），由 m 个叶节点构成、且这些叶节点由$\mathcal{L}_x$ 的专属祖先 W_x 标识的森林，也可用于研究有意义的前缀感知右扩展 R_x（定义 9）。

定义 12 (首个结束位置)。 给定 $\mathsf{GST}_{\mathsf{MSA}}$，令 F_x 为$\mathsf{GST}_{\mathsf{MSA}}$ 中属于某个专属祖先 $w \in W_x$所根植子树的所有显式节点的集合。然后，对每个 $v \in F_x$ ，我们将值 $\mathrm{pos}(v)$ 定义为首个

[6]正如本文会议版本的匿名审稿人所指出的，本节对应内容中 [29],，由于我们只遍历叶节点，因此严格来说并不需要支持后缀链接。事实上，只需遍历树即可轻松填充一个包含$\mathcal{L}_0, \ldots \mathcal{L}_{n-1}$的表，然后我们必须将其存储起来。

算法 4： 计算 $0 \le x \le n-1$ 的最小右扩展 $f(x)$ 的算法。为简单起见，我们使用全序 $\le$ 来检查独占祖先集问题的失败条件，但容易看出，可以用叶节点的额外二元着色来替代 $\le$，以检查某个叶节点是否包含在感兴趣的连续叶节点集合中。

输入： 来自字母表 $\Sigma \cup$ 的多序列比对 $[1..m, 1..n]\,\{-\}$

输出： 对于 $x = 0$、.....、$n-1$，输出对 $(x,\ f(x))$

预处理多序列比对行，以支持选择和秩查询；
构建 $\mathrm{GST}_{\mathrm{MSA}}$，即下列内容的广义后缀树
  {拼写(MSA$[i, 1..n]$)\$$_i$: $1 \le i \le m$};
将 L$[i]$ 初始化为对应于 spell(MSA$[i, 1..n]$) 的 $\mathrm{GST}_{\mathrm{MSA}}$ 叶节点；
对于 $x \leftarrow 0$ **到** $n-1$ **执行**
    对于 $i \leftarrow 1$**到** m **执行** ▷ 标记 $\mathcal{L}$ 的每个叶节点 x
        L$[i]$.已标记 $\leftarrow$**真**;
    ▷ 处理每组连续叶节点
    对于i $\in \{1, \ldots, m\}$: L$[i]$.prevleaf.已标记 $=$**false do**
        lb $\leftarrow$ L$[i]$; ▷ 查找左右边界
        rb $\leftarrow$ lb;
        当rb.nextleaf.marked $=$**true do**
            rb $\leftarrow$ rb。下一个叶节点;
        $w \leftarrow$lb; 左叶节点 $\leftarrow$lb; 右叶节点 $\leftarrow$ lb; ▷ 查找专属祖先
        当 rleaf $\le$rb **执行**
            $w' \leftarrow w$.父节点; 左叶节点$' \leftarrow w'$.最左叶节点; 右叶节点$' \leftarrow$ w'.最右叶节点;
            如果 lb $\le$lleaf$' \wedge$ rleaf$' \le$rb **那么** ▷ w' 是一个正确的替换节点
                $w \leftarrow w'$; 左叶节点 $\leftarrow$ 左叶节点$'$; 右叶节点 $\leftarrow$ 右叶节点$'$;
            else ▷ w' 失败，因此 w 是专属祖先
                $g \leftarrow (w$.父节点)。字符串深度 $+ 1$;
                $\ell \leftarrow$ 左叶节点;
                当 $\ell \le$右叶节点 **时**
                    $i' \leftarrow \ell$.获取行;
                    f$[i'] \leftarrow$ 多序列比对$[i', 1..n]$.选择(多序列比对$[i', 1..n]$.秩$(x) + g)$;
                    $\ell \leftarrow \ell$.下一个叶节点;
                $w \leftarrow \ell$; 左叶节点 $\leftarrow \ell$; 右叶节点 $\leftarrow \ell$;
    输出 $(x$,最大值$_{i=\ 1}^{m}$ f$[i])$;
    对于 $i \leftarrow 1$**到** m **执行** ▷ 清理并计算 $\mathcal{L}_{x+1}$
        L$[i]$.标记 $\leftarrow$**false**;
        如果 多序列比对$[i, x+1] \ne -$ **则**
            L$[i] \leftarrow$ L$[i]$.后缀链接;

字符串的结束位置在多序列比对中的位置，其中 字符串(parent(v))· 字符(v)，其中 字符(v)是从 v 的父节点到 v 的标签中的第一个字符。换句话说，如果 S =字符串(parent(v)) $\in \Sigma^+$ 且 c = 字符(v) $\in \Sigma$，那么 pos(v) 就是满足以下条件的最小列索引 $y \in [1..n]$：对于某个 $1 \le i \le m$， Sc = 拼写(多序列比对$[i, x+1..y]$)。

集合 W_x、 F_x 以及 pos(v) 的值示例见图7和图8。将这些值绘制在一条水平线上后，很容易发现， H 的所有增量都对应某个 pos(v)，但反过来不成立：那些不会影响 H的 pos 值（因为它们不对应两行或更多行读取彼此之间不存在前缀关系的字符串），是分支节点在 pos 值意义下的最先出生的子节点。

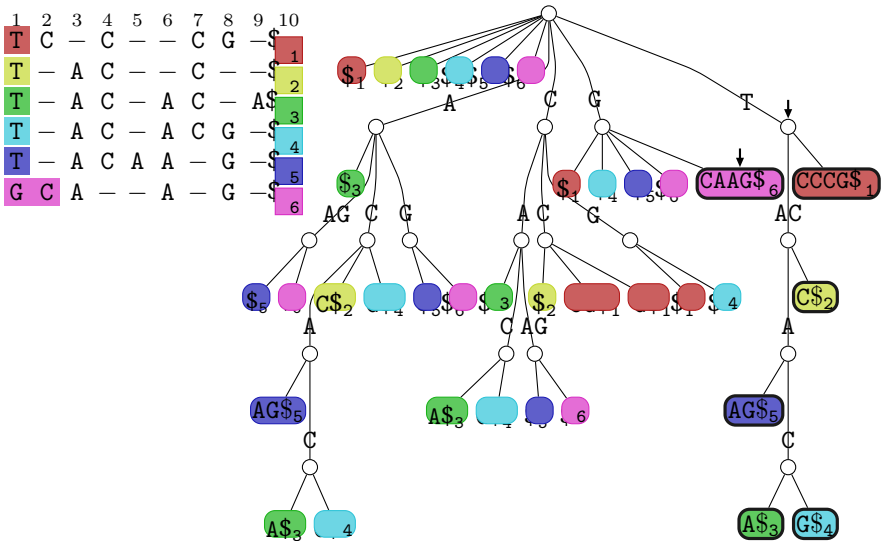

图7：另一个多序列比对$[1..6, 1..9]$ 及其 GST_{MSA}的示例。我们还用黑色轮廓标出了与后缀 spell(MSA$[i, 1..n]$) 对应的叶节点$\mathcal{L}_0$；它们的专属祖先 W_0（即对应于G· CAAG$\$_6$ 和 T 的节点）用箭头标记。

定义13（长子节点）。给定GST_{MSA} 及其森林 F_x，该森林对应于从列 $x+1$ 开始的所有半无重复字符串，其中 $0 \le x < n$。对于每个内部节点 $v \in F_x$，我们任意选择一个 pos 值最小的子节点，作为 F_x 的长子节点。然后，令 F_x 为 F_x 中非首生节点的子集，该子集由从 F_x 中移除这些节点得到。

引理 9。给定GST_{MSA} 以及与列 x 相关联的非首生节点集合 F_x，对于任意 $y \in [f(x)..n]$，片段 $[x+1..y]$ 的前缀感知高度等于 F_x 中 pos 值小于或等于 y 的节点数，记作 $H([x+1..y]) = \left|\{\hat{v} \in \hat{F}_x : \text{pos}(\hat{v}) \le y\}\right|$.

证明。对于任意 $v \in F_x$，令 I_v 为满足以下条件的行索引集合： v 覆盖 GST_{MSA} 中与 spell(MSA$[i, x+1..n]$) 对应的叶节点。根据 GST_{MSA} 的性质可知，若任意 v_1， $v_2 \in F_x$ 互不为对方的祖先，则对应的字符串 string(parent(v_1))· char(v_1) 和 string(parent(v_2)) ·

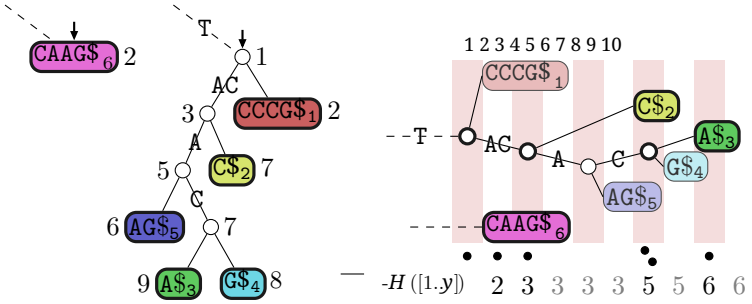

图 8：左侧是图7中示例多序列比对的森林 F_0 ，并标注了 pos(v) 的值。右侧将同一森林与多序列比对^的列对应绘制，其中仅突出显示了 F_0 的非首生节点。注意， $f(0) = f^6(0) = 2$。

char(v_2) 互不为对方的前缀：若 $y \geq$pos(v_1) 且 $y \geq$pos(v_2)，则H(MSA $[I_{v_1} \cup I_{v_2}, x+1..y]) = H(\text{MSA}[I_{v_1}, x+1..y]) + H(\text{MSA}[I_{v_2}, x+1..y])$。我们将这一关键性质称为旁系亲属的独立性。[7] 特别地，对于某个节点 $v \in F_x$ 的任意子节点子集 U ，只要满足$y \geq \max_{u \in U}$pos(u)，该性质便成立；对于专属祖先 $W_x \subseteq F_x$，该性质也成立，因为根据假设，$y \geq f(x) \geq f^i(x) > \max_{w \in W_x}$ pos(w)：

$$\overline{H}\big(\text{MSA}[1..m, x+1..y]\big) = \sum_{w \in W_x} \overline{H}\big(\text{MSA}[I_w, x+1..y]\big). \tag{5}$$

现在，我们可以证明对任意节点 $v \in F_x$的行 I_v 所作的论题修正。为此，我们引入最后一个记号：用$\widehat{\ }F_x^v$表示集合 $(F_x \cap \text{GST}_{\text{MSA}}(v)) \cup \{v\}$，它也涵盖了 v 为长子节点的情况。于是，对于任意 $v \in F_x$ 以及 $y \in [\max_{i \in I_v} f^i(x)..n]$ ，我们有：

$$\overline{H}\big(\text{MSA}[I_v, x+1..y]\big) = \begin{cases} 1 & \text{if } y < \text{pos}(v), \\ \big|\{\hat{v} \in \widehat{F}_x^v : \text{pos}(\hat{v}) \leq y\}\big| & \text{otherwise.} \end{cases} \tag{6}$$

公式（6）的证明通过对以 v 为根节点的子树高度进行归纳来完成。

^**基本情形：** 如果 v 是叶节点，那么 $F_j^v = \{v\}$， $I_v = \{i\}$ 对于某个 $i \in [1..m]$ 成立，并且 H （多序列比对 $[\{i\}\ x+1..y]$） $= 1$，因此公式（6）很容易验证。

归纳假设： 对于所有满足以 v 为根节点的子树高度小于或等于 $h \geq 0$ 的节点 v ，公式（6）均成立。

归纳步骤： 设以 v 为根的子树高度等于 $h+1$，并设 $u_1, \ldots, u_p$ 是 v 的 $p \geq 2$ 个子节点，其中 u_1 是最先出生的子节点。若 $y <$ pos(v)，则多序列比对中所有 $Sc =$ string(parent(v))· char(v) 的出现位置都在列 y 之后结束，因此所有满足 $i \in I_v$ 的字符串 spell(MSA$[i, x+1..y]$) 都是 Sc 的前缀，且 $H(\text{MSA}[I_v, x+1..y]) = 1$。使用相同的

[7]在谱系学术语中，祖先关系被称为直系；与之相对，非直系亲属之间的关系则称为旁系。

论证可知，若 $y <$pos(u_1)，公式（6）同样成立，因此我们可以假设 $y \geq$ pos(u_1) $>$ pos(v)。考虑满足 $y <$pos(u_k) 的 v 的子节点 u_k，其中 $2 \leq k \leq p$；对应行 I_{u_k} 中拼写出的字符串都是 string(parent(u_1)) 的前缀，因此在计算前缀感知高度时会忽略它们。若 $U_{\leq} \coloneqq \{u_k \colon 1 \leq k \leq p \wedge \text{pos}(u_k) \leq y\}$，则

$$\begin{aligned}
\overline{H}\big(\mathsf{MSA}[I_v, x+1..y]\big) &= \overline{H}\left(\mathsf{MSA}\left[\bigcup_{u \in U_{\leq}} I_u\right]\left[x+1..y\right]\right) \\
&= \sum_{u \in U_{\leq}} \overline{H}\big(\mathsf{MSA}[I_u, x+1..y]\big) && \text{独立性的旁系亲属} \\
&= \sum_{u \in U_{\leq}} \big|\{\widehat{u} \in \widehat{F}^u_x : \text{pos}(\widehat{u}) \leq y\}\big| && \text{归纳假设} \\
&= \big|\{\hat{v} \in \widehat{F}^v_x : \text{pos}(\hat{v}) \leq y\}\big|.
\end{aligned}$$

注意，最后一个等式成立是因为 $F^{u_1}_x$ 的 pos(u_1) 被替换为 F^v_x 的 pos(v)。

该论题由公式 (5) 和 (6) 得出，因为专属祖先将行 $[1..m]$ 划分为 $|W_x|$ 个集合。另外，注意到 $|F_x| = m$。 □

集合 F_x 和 F_x 的示例见图 8。遗憾的是，若在本地进行朴素计算，需要耗时 $O(m^2)$，因为 $\mathsf{GST}_{\mathsf{MSA}}$ 不包含 MSA 子串结束位置的信息，而且也无法轻易对其进行扩充以包含这些信息。

引理 10. 给定一般的 多序列比对$[1..m, 1..n]$、$\mathsf{GST}_{\mathsf{MSA}}$以及专属祖先 W_x，我们可以在时间 $O(m^2)$内计算有意义的前缀感知右扩展 R_x 。

证明。 对于每个 $v \in F_x$，我们可以通过对每一行 $i \in I_v$查找 MSA$[i, 1..n]$ 中出现的字符串 $Sc =$ string(parent(v)) ·char(v) 的结束位置 y_i，来计算 pos(v)（Sc 是一个半无重复子串，因此每行至多出现一次）。换言之，位置 y_i 对应于行 i中的第 $k =$ 个非空字符，其中 $+$rank(MSA$[i, 1..n], x$) $+$ 1stringdepth(parent(v))。然后，pos(v) $=$ min$_{i \in I_v} y_i$。选择其子节点中 pos 值最小的一个，即可找到 v 的最先出生的子节点；移除最先出生的节点后，得到的 pos 值 F_x。给定 $f(x)$和按序排列的 pos 值，一个简单的算法（如算法 5）会检查所有包含 pos 值的列，并将相对前缀感知右扩展输出为三元组 $(x, y,H([x+1..y])$。

由于我们可以在线性时间内预处理多序列比对行，从而在常数时间内回答 rank 和 select 查询，因此每个 pos(v) 的计算需要 $O(|I_v|)$ 时间。森林 F_x 由压缩树组成，共有 m 个叶节点，因此包含 $O(m)$ 个节点： F_x 的子树可能不平衡，所以总时间为 $O(m^2)$。然后，可以在 $O(m\log\ m)$ 时间内对这些值进行排序，再用 $O(m)$ 时间处理。 □

5.5 使用广义前缀树加速

根据引理 10，我们可以在 $O(m^2n)$ 时间内计算有意义的前缀感知右扩展，其中瓶颈是计算 pos(v) 的值

算法 5： 给定 F_x 的排序后的 pos 值，计算有意义的前缀感知右扩展的算法。通过从 F_x 中移除长子节点得到集合 F_x 。**输入：** 值 $f(x)$，节点在 $\hat{}$ F_x 中的值 pos(w_1) . . . , pos(w_m)，按从小到大的顺序排列。**输出：** 有意义的前缀感知右扩展 R_x.**1** $h \leftarrow 1$;**2 当** pos(w_h) $< f(x)$ **时执行** **3** $h \leftarrow h + 1$;**4 当** $h \leq m$ **时执行5 如果** $h = m \vee$ pos(w_{h+1}) $\neq$ pos(w_h) **则 6 输出** (x,pos(w_h) h);**7** $h \leftarrow h + 1$;

（定义 12）。其他任务可通过将解改编为全局计算，在 $O(mn)$ 时间内执行：所有 F_0，.....， F_{n-1} 的 pos 值总计为 $O(mn)$；由于这些值取自 $[1..n]$，将它们一起排序需要 $O(mn)$ 时间；之后再分别处理它们，总计只需线性时间。$\mathsf{GST}_{\mathsf{MSA}}$ 不包含 MSA 字符串的结束位置的信息，但包含第一个起始位置（即出现位置）的信息：事实上，叶节点集合 $\mathcal{L}_0, ...、\mathcal{L}_{n-1}$ 分别考察从某个 MSA 列开始的每一个后缀。这为我们提供了关键思路：利用对称性，高效计算 pos(v) 的值。

定义 14。 给定一个一般的 $\mathsf{MSA}[1..m, 1..n]$，我们将 $\mathsf{GPT}_{\mathsf{MSA}}$ 定义为字符串集合 $\{\$_i \cdot$拼写$(\mathsf{MSA}[i, 1..n]) : 1 \leq i \leq m\}$ 的广义前缀树，其中$\$_1$，.....，$\$_m$ m 是 Σ 中没有的新互异终止符号。或者，$\mathsf{GPT}_{\mathsf{MSA}}$ 也可以构造为集合 $\{$拼写$(\mathsf{MSA}[i, 1..n])^{-1} \cdot \$_i : 1 \leq i \leq m\}$ 的广义后缀树。

观察 5。 请注意，在$\mathsf{GPT}_{\mathsf{MSA}}$ 字符串中，字符是从右向左读取的。对于$\mathsf{GPT}_{\mathsf{MSA}}$ 中的每个节点 u ，令$\mathrm{pos}^{-1}(u)$ 表示某个 *MSA* 行中$\mathrm{string}(u)$ 的首个结束位置：

- 对于$\mathsf{GPT}_{\mathsf{MSA}}$ 中与行 $i \in [1..m]$对应的任意叶节点 ℓ ，有$\mathrm{pos}^{-1}(\ell)$ 等于 $\mathsf{MSA}[i, 1..n]$中的第 k 个非空字符，其中 $k = |\mathrm{string}(\ell)|$;
- 对于$\mathsf{GPT}_{\mathsf{MSA}}$中的任意内部节点 u，令 v_1、...、 v_p 为其子节点；则$\mathrm{pos}^{-1}(u) = \min_{k=1}^{p} \mathrm{pos}^{-1}(v_k)$;
- 给定$_{\mathsf{MSA}}$GST中的一个节点 v ，令 v^{-1} 为$\mathsf{GPT}_{\mathsf{MSA}}$ 中与字符串 $\mathrm{string}(\mathrm{parent}(v)) \cdot \mathrm{char}(v)$ 从右向左读取后对应的节点；若这是一个隐式节点，则将$\mathsf{GPT}_{\mathsf{MSA}}$中的首个显式祖先定义为 v^{-1} ；那么$\mathrm{pos}(v) = \mathrm{pos}^{-1}(v^{-1})$。

例如，如果 v 是图7和图8中对应于字符串 $\texttt{TAC} \cdot \texttt{A}$的 $\mathsf{GST}_{\mathsf{MSA}}$ 节点，那么 v^{-1} 对应于图9的 $\mathsf{GPT}_{\mathsf{MSA}}$ 中的 ACAT ，且 $\mathrm{pos}^{-1}(v^{-1}) = 5$。

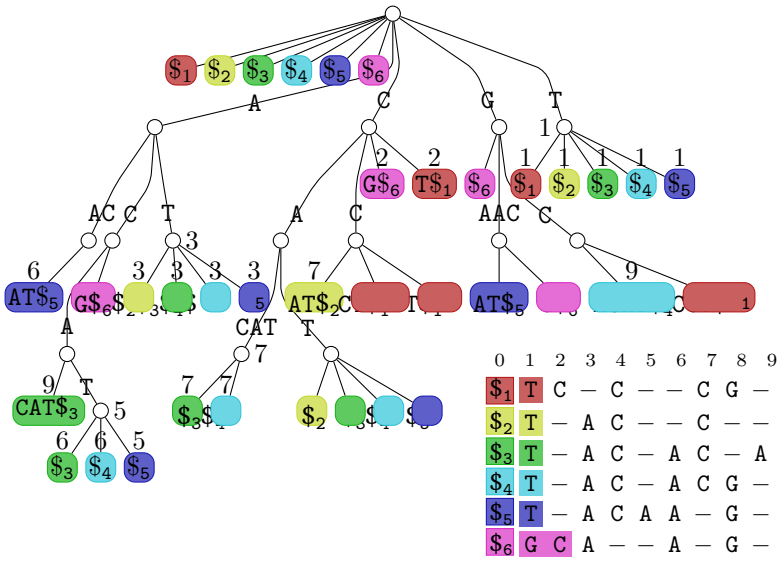

图9：由图7的多序列比对构建的 $\mathrm{GPT}_{\mathrm{MSA}}$ 示例，其中标注了与计算 R_0 （图8）相关的 pos^{-1} 值，以及从第 1 列开始的有意义的右扩展。

引理11。 给定一个一般的 多序列比对$[1..m, 1..n]$、$\mathrm{GST}_{\mathrm{MSA}}$和$\mathrm{GPT}_{\mathrm{MSA}}$，$\mathrm{pos}(v)$ 可在 $O(mn)$ 时间内计算出 $\mathrm{GST}_{\mathrm{MSA}}$ 中任意节点 v 的值。

证明。 如观察 5 所示，$\mathrm{GPT}_{\mathrm{MSA}}$ 的树结构使得能够递归计算 $\mathrm{pos}^{-1}(v)$：与在$\mathrm{GST}_{\mathrm{MSA}}$中计算 $\mathcal{L}_0, ...\mathcal{L}_{n-1}$ 类似，这需要 $O(mn)$时间。接下来需要证明，给定 $\mathrm{GST}_{\mathrm{MSA}}$ 的节点 v，我们能够在 $O(1)$时间内找到 $\mathrm{GPT}_{\mathrm{MSA}}$ 的节点 v^{-1}：一种表示字符串的同步后缀树和前缀树的数据结构，已通过词缀树（或相应的词缀数组）的形式构建出来，并支持线性时间构造算法 [21, 34]。不过，这些结果都假设字母表 Σ的大小为常数。另一种方法是直接在多序列比对中定位字符串 $\mathrm{parent}(v) \cdot \mathrm{char}(v)$ 的一次出现，从而通过在 $\mathrm{GPT}_{\mathrm{MSA}}$ 中回答相应的加权祖先查询找到 v^{-1}。Belazzougui 等人最近证明，我们可以在线性时间内对后缀树进行预处理，从而能够在常数时间内回答加权祖先查询，且无需对字母表大小作任何假设 [3]。

□

定理6的证明至此结束：正如我们在第 4.6 节中已经展示的，有意义的前缀感知右扩展可以直接替代原有有意义右扩展算法 2，因此引理11表明，最小化最大前缀感知高度的最优分段可在线性时间内计算。

6 在半无重复 EFG 中为一组预定义路径建立索引

作为原始多序列比对序列的一种表示，EFG 是一种有损数据结构：该图既能拼写出原始序列，也能拼写出它们的重组，却没有直接的机制来判断某个模式是否确实出现在原始序列中。为增强 EFG 框架在实际应用中的能力，我们着重研究基于BWT的索引，以便在半重复自由 EFG 中查找给定模式的精确匹配，同时仅考虑图中预先指定的一组路径。

问题2（EFG路径列举问题的索引）。给定一个半无重复 *EFG* $G = (V, E, \ell)$ 以及 G 中的一组路径 P，建立 G 的索引，以回答以下查询：给定一个模式 Q，查找并报告 P 中包含模式 Q 作为子串的路径 P。

因此，EFG 可用于确定生成该 EFG 的多序列比对序列中，哪些序列包含给定模式。实现这一目标的整体策略是构造一对可索引文本，方法与第3节中的做法类似；然后生成一个 *Burrows–Wheeler* 变换索引 并用额外的数据结构加以增强。为此，我们借鉴了 Norri [26] 提出的一些思路。

回顾第2节中文本 T 的广义后缀数组和 Burrows–Wheeler 变换$_T$ 的定义。

定义15。Burrows–Wheeler 变换索引 是一种由文本 T 生成的数据结构，需要 $O(|T|s_{\mathsf{LF}})$ 位空间，并支持以下操作。

- 在Burrows–Wheeler 变换$_T$ 中以常数时间访问第 i 个字符。
- 给定一个字典序范围 $[l..r]$，可以在 $O(t_{\mathsf{LF}})$ 时间内对字符 c 进行向后搜索 。该操作会得到 T 中以 c 为前导字符的后缀在 $[l..r]$ 中对应的字典序范围。

在本节中，t_{LF} 和 s_{LF} 是变量，其取值取决于具体的 BWT 索引。

引理*12.* *[2,* 定理*6.2]* 存在一种 *BWT* 索引，使得对于任意正数 k，有 $t_{\mathsf{LF}} = \log\log\sigma + k$，前提是 $\sigma \le |T|$ 且 $\lceil \log\sigma \rceil \le \mu$，其中 μ 是以位为单位的机器字长。该索引数据结构可以在 $O(|T|)$ 时间内构建，使用$O(|T|\log\sigma)$ 位工作空间，并且需要 $|T|\log\sigma(1+\frac{1}{k}) + O(|T|\log\log\sigma)$ 位空间。通过选择 $k = 1$，可得 $t_{\mathsf{LF}} = \log\log\sigma$ 且 $s_{\mathsf{LF}} = \log\sigma$。

该搜索算法基于半无重复 EFG 的一项性质：任意两个相连节点的节点标签串接后，其某些前缀各不相同，如下列引理所示。

引理*13.* *[26,* 引理 *4.4]* 假设 $G = (V, E, \ell)$ 是一个半重复自由 *EFG*，且 $v_1, v_2, w_2 \in V$ 是节点，使得 $(v_1, w_2) \in E$，v_2 与 w_2 位于同一个块中，并且 $\ell(v_2) \preceq \ell(w_2)$。不存在一条 G 的路径 $W = u_1 \cdots u_t$，使得 $u_1 \neq v_1$ 且 $\ell(v_1)\ell(v_2) \preceq \ell(u_1) \cdots \ell(u_t)$。

情形	条件	Node	节点标签		
			$i=1$	$i=2$	$i=3,\ldots,t$
1, 2	$\ell(v_1)\ell(v_2) \preceq \ell(u_1)\ell(u_2)$	u_i	$\ell(v_1)\cdot Q$	$\ell(u_2)$	
	$\ell(u_1)\ell(u_2) \preceq \ell(v_1)\ell(v_2)$	v_i	$\ell(v_1)$	$\ell(v_2)$	
3	$\ell(v_1)\ell(v_2) \preceq \ell(u_1)\ell(u_2)$	u_i	$\ell(u_1)$	$\ell(u_2)$	
		v_i	$\ell(u_1)\cdot Q$	$\ell(v_2)$	
4	$\ell(u_1)\ell(u_2) \preceq \ell(v_1)\ell(v_2)$	u_i	$\ell(u_1)$	$\ell(u_2)$	
		v_i	$\ell(u_1)\cdot Q$	$\ell(v_2)$	
4.1	S 可能是 ε	u_i	$\ell(u_1)$	$Q\cdot\ R\cdot\ S\quad T$	
		v_i	$\ell(u_1)\cdot Q$	$Q\cdot\ R$	
4.2	$S \neq \varepsilon$	u_i	$\ell(u_1)$	$Q\cdot\ R$	$S\cdot T$
		v_i	$\ell(u_1)\cdot Q$	$Q\cdot\ R\cdot\ S$	

表1：引理13中的节点标签与分解，其中 W 由至少两个节点组成。

证明. 我们证明，如果存在 W，那么 G 就不是半重复自由的。注意，如果 $\ell(v_1)\ell(v_2) \preceq \ell(u_1)\cdots\ell(u_t)$，那么对于任意 $k \le |\ell(v_1)\ell(v_2)|$，都有 $(\ell(v_1)\ell(v_2))[1,k] = (\ell(u_1)\cdots\ell(u_t))[1,k]$，反之亦然。

假设 W 只由一个节点 u_1组成。那么 $\ell(v_2)$是 $\ell(u_1)$的非前缀子串，这与 G 是半无重复的相矛盾。

现在假设 W 由至少两个节点组成。节点标签的分解方式有四种可能，如表1所示，其中 Q和S 是一些非空字符串：

1. $\ell(u_1) = \ell(v_1)\cdot Q$ 以及 $\ell(v_1)\ell(v_2) \preceq \ell(u_1)\ell(u_2)$。如果 W存在，那么 $\ell(v_2)$将是 $Q\cdot\ \ell(u_2)$的前缀，因此也是 $\ell(u_1)\ell(u_2)$的非前缀子串，这与 G 是半无重复的相矛盾。2. $\ell(u_1) = \ell(v_1)\cdot Q$ 以及 $\ell(u_1)\ell(u_2) \preceq \ell(v_1)\ell(v_2)$。由此可知，$Q\cdot\ \ell(u_2)$ 是 $\ell(v_2)$ 的前缀，因此 $\ell(u_2)$ 是 $\ell(v_2)$ 的非前缀子串，这再次与 G 是半无重复的相矛盾。3. $\ell(v_1) = \ell(u_1)\cdot Q$ 以及 $\ell(v_1)\ell(v_2) \preceq \ell(u_1)\ell(u_2)$。与前一种情况类似，$Q\cdot\ell(v_2) \preceq \ell(u_2)$，因此 $\ell(v_2)$是$\ell(u_2)$的非前缀子串，矛盾。
4. $\ell(v_1) = \ell(u_1)\cdot Q$ 以及 $\ell(u_1)\ell(u_2) \preceq \ell(v_1)\ell(v_2)$。由于 G 是半无重复的，且 Q 是 $\ell(v_1)$的后缀，Q 也必须是 $\ell(u_2)$和 $\ell(v_2)$的前缀。此外，或者是 $\ell(u_2) = Q\cdot\ R$，或者是 $\ell(v_2) = Q\cdot\ R$；其中 R是非空字符串，因为否则，作为某个节点标签的 Q 就会成为 $\ell(v_1)$的真后缀，从而 G 就不是半无重复的。现在，分解 $\ell(u_2)$和 $\ell(v_2)$有两种可能：

(a) $\ell(u_2) = Q\cdot\ R\cdot S$ 和 $\ell(v_2) = Q\cdot R$。若 $S = \varepsilon$, $u_2 = v_2$。由于$\ell(v_1)\ell(v_2) = \ell(u_1)\cdot Q\cdot\ Q\cdot\ R$，必有 $Q\cdot\ Q\cdot\ R\cdot\ S \preceq \ell(u_2)\cdots\ell(u_t)$。现在可知，$\ell(v_2) = Q\cdot\ R$是一个非前缀子

字符串 $\ell(u_2)\cdots\ell(u_t)$:

$$Q\cdot\underbrace{Q\cdot R}_{\ell(v_2)} \preceq \underbrace{Q\cdot R\cdot S}_{\ell(u_2)}\cdot\underbrace{T.}_{\ell(u_3)\cdots\ell(u_t)}$$

因此 G 将不是半无重复的。

(b) $\ell(u_2)=Q\cdot R$ 和 $\ell(v_2)=Q\cdot R\cdot S$ 对于某个 $S\neq\varepsilon$。现在我们有

$$Q\cdot\underbrace{Q\cdot R\cdot S}_{\ell(v_2)} \preceq \underbrace{Q\cdot R}_{\ell(u_2)}\cdot\underbrace{S\cdot T.}_{\ell(u_3)\cdots\ell(u_t)}$$

由于 $Q\cdot Q\cdot R\cdot S=Q\cdot\ell(v_2)$、$Q\cdot R=\ell(u_2)$、$S\cdot T=\ell(u_3)\cdots\ell(u_t)$，并且 Q, R, S 和 T 均不等于 ε，$\ell(v_2)$ 现在是 $\ell(u_2)\cdots\ell(u_t)$ 的非前缀子串，这就产生了矛盾，因为 G 是半无重复的。 □

利用这一结果，我们可以构造一个类似于 D_F 和 D_R 的可索引文本，该文本来自第3节，由 EFG 的边所拼写的字符串串接而成。为了让每个节点标签在文本后缀的字典序中处于特定位置，我们还加入最后一个EFG块中的节点标签，也就是只有入边的节点。

$$D'_F=\mathbf{0}\cdot\prod_{v\in V^-}\ell(v)\mathbf{0}\cdot\prod_{(v,w)\in E}\ell(v)\ell(w)\mathbf{0}\quad\text{where } V^-=\{v\in V\mid(v,\cdot)\notin E\}$$

判断给定模式是否作为子串出现在 EFG 中某条路径上，可以按以下方式完成。

引理14. 设 $G=(V,E,\ell)$ 是一个半重复自由 *EFG*，且 $[l..r]$ 和 $[l'..r']$ 是 $\mathsf{SA}^{D'_F}$ 中某个 $\ell(v)$ 和 $\ell(w)$ 的字典序范围（顺序不一定相同），其中 $v,w\in V$。那么，$l'\le l$ 和 $r\le r'$ 当且仅当 v 和 w 位于 G 的同一个块中。

证明。 我们通过证明各个条件相互蕴含来证明该引理。

(⇒) 由于字典序范围彼此嵌套，因此 $\ell(v)$ 必定是 $\ell(w)$ 的前缀，反之亦然。由于 G 是半重复自由的，v 和 w 必定位于同一个块中。

(⇐) 由于 v 和 w 位于 G 的同一个块中，且 $\ell(v)\preceq\ell(w)$，反之亦然，因此它们在 $\mathsf{SA}^{D'_F}$ 中的字典序范围必定嵌套。 □

引理*15*（扩展反向搜索）。 *[26,* 引理 *4。6]* 假设 $G=(V,E,\ell)$ 满足 $\ell\colon V\to\{1,\ldots,\sigma\}$ 是一个非平凡的半无重复 *EFG*。通过在 $O(L(|E|+|V|t_{\mathsf{LF}}))$ 时间内对 G 进行预处理，生成一个大小为 $O(L|E|s_{\mathsf{LF}})$ 位的数据结构，我们可以针对任意模式 Q，确定一个节点编码 $\{v_1,v_2,\ldots,v_k\}\subset V$，使得对于所有 $1<i\le k$，都存在 $(v_{i-1},w_i)\in E$，满足 w_i 与 v_i 位于同一个块中，且 $\ell(w_i)\preceq\ell(v_i)$，并且 Q 是 $\ell(v_1)\ell(v_2)\cdots\ell(v_k)$ 的子串。这里 $L=\max^{(v,w)\in E}(|\ell(v)|+|\ell(w)|)$。如果 Q 跨越 G 中至少两个节点标签的串接，该操作的时间开销为 $O(|Q|t_{\mathsf{LF}})$。否则，确定所有包含 Q 的边标签需要 $O((|Q|+L\cdot occ)\cdot t_{\mathsf{LF}})$ 时间，其中 occ 是 Q 在 D'_F 中的出现次数。

证明。首先，我们以 D'_F 为输入构建一个 BWT索引。这可以在 $O(|D'_F|) = O(L(|E|+|V|)) = O(L|E|)$ 时间内完成。如果 Q 是任意 $\ell(u)\ell(v)$ 的子串，其中 $(u, v) \in E$，那么利用常规反向搜索，只需 $O(|Q|t_{\mathsf{LF}})$ 时间。

为了能够处理跨越更多节点的模式，作为预处理的一部分，我们还会填充两个各含 $|D_F|$ 个元素的位向量 $\mathcal{B}$ 和 $\mathcal{E}$。当且仅当 $[l..r]$ 是某个$\ell(v)$ 的字典序范围时，我们才设置 $\mathcal{B}[l] = 1$ 和 $\mathcal{E}[r] = 1$，其中 $v \in V$，且 v 是 G 的所有节点标签中最短的前缀。形式化地说，这类节点的集合为 $B := \{\text{argmin}_v \, |\ell(v) \, | \, \ell(v) \preceq \ell(w) \text{ and } v, w \in V\}$。由于 B 中没有任何节点的标签是另一节点标签的子串，因此对应的字典序范围互不相交。此外，$\mathcal{B}$ 已为秩查询做好准备，而 $\mathcal{B}$ 和 $\mathcal{E}$ 均已为 select 查询做好准备。预处理可在 $O(L|V|t_{\mathsf{LF}})$ 时间内完成。

位向量 B 和 E 在搜索过程中按如下方式使用。假设Q 已处理完毕，得到的字典序范围为 $[l..r]$，且 $\text{BWT}_{D'_F}[l] = \mathbf{0}$。然后，我们通过确定一对索引 l' 和 r' 来扩展该字典序范围，使得 $\text{B}[l'] = \text{E}[r'] = 1$、$l' \le l$ 和 $r \le r'$，以及 l'和 r' 对应于同一个字典序范围。首先计算 $r'' := \text{rank(B } l)$、$l' = \text{select(B } r'')$ 和 $r' = \text{select(E } r'')$，然后检查 $l' \le l$ 和 $r \le r'$ 是否确实成立，即可完成此操作。假设 v 和 w 是 G 中的节点，其中$[l..r]$ 是 $\ell(v)$ 的字典序范围，而 $[l'..r']$ 是 $\ell(w)$ 的字典序范围。根据引理14，必定存在一个节点 $w' \in V$，使得 $(w', v) \in E$ 且 $\ell(w) \preceq \ell(w')$。

继续进行反向搜索，并根据 $\text{BWT}^{D'_F}$确定前一个字符又是 **0** 后，根据引理13，可以从字典序范围明确确定当前节点。随后，可以通过再次扩展字典序范围，从同一个块中的另一个节点恢复搜索。如果模式较短，即字典序范围从未扩展，则可以采用传统的分支方法，根据 $\text{BWT}^{D'_F}$ 确定前一个字符并继续搜索，直到每个分支中都找到一个 **0**，从而达到所述时间复杂度 $O((|Q| + L \cdot \text{occ}) \cdot t_{\mathsf{LF}})$。

□

为了滤除既不出现在 EFG 中、也不出现在构建 EFG 所用文本中的假阳性，我们先证明 EFG 的一项性质，然后说明如何利用一些额外的数据结构。

引理16。设 $G = (V, E, \ell)$ 是一个半重复自由 *EFG*，且 $(v, w) \in E$。$\ell(v)\ell(w)$ **0** 在 $\text{SA}^{D'_F}$ 中的（共）字典序范围是单元素集合。

证明。由于 G 是半无重复的，且 **0** 不属于节点标签字母表，因此 $\ell(v)\ell(w)\mathbf{0}$ 不可能在 D'_F 中的其他位置出现，尤其不能作为另一个节点标签串接的后缀或子串。

□

引理 *17.* *[26,* 引理 *4.8]*假设 $G = (V, E, \ell)$ 是一个非平凡的半无重复创始图。边 E 可以赋予不同的编号，其映射为 $\alpha\colon E \to \{1, \ldots, |E|\}$ 和 $\tilde{\alpha}\colon E \to \{1, \ldots, M_E\}$，其中 $M_E = 2 + H' + |E|$，且 H' 是 G第一个块的高度。给定 $SA_{D'_F}\$$ 中任意一个 $\ell(v)\ell(w)0$ 的共字典序秩，其中 $(v, w) \in E$，可以在常数时间内确定 $\tilde{\alpha}((v, w))$ 的值。类似地，给定上述节点标签串接的字典序排名，可以利用一个 $O(|D'_F|)$ 位的数据结构，在常数时间内检索 $\alpha((v, w))$ 的值；该数据结构可在 $O(L|E|t_{\mathsf{LF}})$ 时间内预处理，其中 $L = max_{(v,w)\in E} |\ell(v)| + |\ell(w)|$。

证明。 由于 \$ 和0 是 $\mathrm{BWT}^{D'_F}$ 字母表中字典序最小的字符，因此 $\mathrm{SA}_{D'_F\$}$ 中所讨论的共字典序秩的上界为 $2+H'+|E| = M_E$，最小的两个秩分别对应于 $D'_F\$$ 和初始的 0。由于分段并非平凡的， E 中的边数必定至少与第一个块中的节点数相同，因此 $M_E = O(|E|)$。

为计算 α 的值，我们准备一个含有 D'_F 个元素的位向量 $\mathcal{D}$。当且仅当 l 是 $\mathrm{SA}_{D'_F\$}$ 中某个 $\ell(v)\ell(w)0$ 的字典序排名（见引理16），且 $(v,w) \in E$ 时，置 $\mathcal{D}[l] = 1$。准备好 $\mathcal{D}$ 后，即可在常数时间内进行秩查询，并通过 $\mathrm{rank}(\mathcal{D}\ l)$ 求得 $\alpha((v,w))$ 的值。□

引理 18. *[26, 引理 4.10][8]* 假设 $G = (V, E, \ell)$ 是一个含有 b 个块的非平凡半无重复 *EFG*，且 P 是一组路径。引理15中的索引数据结构可以扩充，以列出包含任意给定模式的路径 P。为此，可以在 $O(L|E|t_{\mathsf{LF}} + |V|)$ 时间内对 G 进行预处理，从而生成一个占用 $O(|V|(|P| + \log b) + |E|(Ls_{\mathsf{LF}} + \log H))$ 位的索引数据结构，其中 b 是 G 中的块数， H 是最大块高度，且 $L = \max_{(v,w)\in E} |\ell(v)| + |\ell(w)|$。如果 Q 跨越 G 中至少两个节点标签的串接，则随后可以在 $O(|Q|t_{\mathsf{LF}} + (|Q| + H^2)d^{|\mathcal{P}|}_{\mu}e)$ 时间内列出包含 Q 的路径；否则，时间为 $O((|Q| + L \cdot occ) \cdot t_{\mathsf{LF}} + occ d^{|\mathcal{P}|}_{\mu}e)$。这里， μ 是以位为单位的机器字长，occ 是 D'_F 中 Q 的出现次数。

证明。 我们最终希望获得 EFG 的压缩表示。为此，我们为 G 中的节点定义一种保留块顺序的顺序。以下定义了这样一种顺序，它还会使节点按照节点标签的字典序排列。设 V_i 和V_j 是 G 中的块，且 $v \in V_i$ 和 $w \in V_j$。

$$v <_\varrho w \Leftrightarrow i < j \text{ or } (i = j \text{ and } \ell(v) < \ell(w))$$

我们用 ϱ表示节点按此顺序排列所对应的置换： $V \to \{1, \ldots, |V|\}$。如有需要，可以使用计数排序 [32] 按块编号对节点进行排序，时间为 $O(|V|)$。若要按节点标签的字典序对节点排序，则可以花费$O(L|V|t_{\mathsf{LF}})$时间，从由特殊字符分隔的节点标签串接而成的序列构建 BWT索引，并搜索每个节点标签。

对于压缩表示，我们存储 EFG 的各种属性。在整数向量 N 中，我们按照相同顺序存储那些在 $\mathcal{B}$ 中存储了 1 的节点的块编号，耗时 $O(L|V|t_{\mathsf{LF}})$，并使用 $O(|V|\log b)$ 位空间。为了在常数时间内计算块高度的累积和，我们将块高度以一元编码存储在位向量 B 中，并对其进行预处理以支持秩和选择$_0$ 查询。预处理耗时 $O(|V|)$，所需位空间与前述相同。为了确定由一条边连接的节点˜编号，我们还填充四个整数向量： A、 A、$\mathcal{L}$、˜和$\mathcal{R}$。在 A （分别为 A）中，对于每个（v, w） $\in E$，我们按照 $<_\varrho$ 定义的顺序，以 α（（v, w））（分别为 $\tilde{\alpha}$（v, w））为键，存储 v （w）在其所在块中的秩。此过程耗时 $O(L|E|t_{\mathsf{LF}})$，并使用 $O(|E|\log h)$ 位空间。

[8]虽然原始版本的 [26, 引理 4.10] 在反向搜索过程中还维护共字典序范围，但后来发现，只需将模式 Q 处理两次（这相当平凡），实际上并不需要这样做。因此，定理 8 中所证明的时间复杂度里的乘数从 $\log \sigma$ 降至 $\log\log\sigma$。

最后，对于每个（v, w） $\in E$，我们设置 $\mathcal{L}[\tilde{\alpha}((v,w))] = \mathcal{R}[\alpha((v,w))] = \varrho(w) - \varrho(v)$。这需要 $O(|E| \log H)$ 位空间，且可按照引理17所述，在 $O(L|E|t_{\mathsf{LF}})$ 时间内完成。为了计算 $\tilde{\alpha}$ 的值，我们还为 ${D'_F}^{-1}$ 构建一个 BWT索引。

为了确定一个节点出现在哪些预定义路径上，我们为每个节点存储一个位向量，从而建立倒排索引。假设 $f: \mathcal{P} \to \{1, \ldots, |\mathcal{P}|\}$ 是一个双射；那么我们设置 $U_{\varrho(v)}[f(P)] = 1$，其中 $U_{\varrho(v)}$ 是这样的位向量，而 $P \in \mathcal{P}$。

作为最后的准备步骤，将每个 $|\ell(v)|$ 以一元编码存储在位向量 X 中，并按照 ϱ 的顺序排列；其中 $v \in V$ 是一个在 $\mathcal{B}$ 中存储了 1 的节点。换言之，对于每个这样的节点，我们设置 $X[1] = 1$，后接一串 0 和一个 1。因此，在为常数时间秩和 select 查询准备好 X 之后，就可以在常数时间内确定节点标签的长度。此准备步骤可在 $O(|V|L) = O(|E|L)$ 时间内完成，并需要 $O(|V|L)$ 位空间。

现在，我们可以确定在 G 中，作为扩展反向搜索的一部分，被遍历的节点所属的最短路径子集。首先，假设 Q 至多跨越一个节点的标签。在这种情况下，可以采用引理15中所示的常规搜索继续方法。给定某个分支，假设在找到第一个 0 字符之前，其字典序排名为 l。块编号 b' 可以通过 $N[\mathsf{rank}(\mathcal{B}\ l)]$, 确定；而 $\varrho(v)$ 的值可先通过 $a := \mathsf{rank}(\mathcal{D}\ l)$ 确定，再通过 $\mathsf{rank}(B, \mathsf{select}_0(B, b' + 1)) + A[a]$ 确定，两者均为常数时间。$\varrho(w)$ 的值可以通过将 $\mathcal{R}[a]$ 加到 $\varrho(v)$ 上计算得到，同样只需常数时间。最后，可以通过按位运算 $U_{\varrho(v)} \wedge U_{\varrho(w)}$ 确定 Q 出现于哪些路径中。

现在，假设 Q 跨越了多个节点的标签。那么存在一种分解：$Q = P \cdot \ell(v_{j_1}) \cdot \ell(v_{j_2}) \cdots \ell(v_{j_k}) \cdot S$，其中 $v_{j_1} . v_{j_2} \ldots v_{j_k}$ 是 G 中至少包含一个节点的路径，并且 P 和 S（分别表示前缀和后缀）是至少有一个非空的字符串，如图10所示。注意，S 的前缀包含一个完整的节点标签，而 P 不包含任何完整标签。当字典序范围至少扩展两次后，如引理15所示，可以按如下方式确定边的左侧节点编号。假设在扩展之前，边 $(v, w) \in E$ 的字典序排名为 l；扩展之后，某个节点标签 $\ell(v')$ 的字典序范围为 $[l' .. r']$。v 和 v' 的块编号 b' 可通过 $N[\mathsf{rank}(\mathcal{B}\ l')]$, 确定；$\varrho(v)$ 的值也可像前一种情况那样确定，两者均为常数时间。

考虑 P 和 S，我们希望确定集合 $\mathcal{H} = \{u \mid (u, v_{j_1}) \in E$ 和 P 是 $\ell(u)\ell(v_{j_1})$的后缀$\}$ 以及 $\mathcal{T} = \{(v, w) \mid (v, w) \in E$ 和 $S \preceq \ell(v)\ell(w)\}$中的节点。

我们记作 $H' = P \cdot \ell(v_{j_1})$。假设 $[l..r]$ 是$H' \cdot 0$ 的共字典序范围。现在，值 $\tilde{\alpha}(\{(u, v_{j_1}) \mid u \in \mathsf{H}\})$为 $l, l+1, \ldots, r$，可将它们用作 L 的键，从 $\varrho(v_{j_1})$计算 $\varrho(\mathsf{H})$。H 中至多有 H 个节点。对于 S，假设$(u, w) \in \mathsf{T}$，且 r' 是扩展前的字典序排名。与之前一样，可以根据扩展后的字典序范围和 $\varrho(w) = \varrho(u) + \mathsf{R}[r']$确定 $\varrho(u)$的值。成立的是 $|\mathsf{T}| \leq H \cdot \min\{H, |Q|\}$。因此，可以在 $O(h \cdot \min\{h, |Q|\})$ 时间内确定 T 中各条边对应的节点编号。可以使用按位运算确定找到的路径所处的预定义路径，使用$U_{\mathcal{H}} \wedge U_{\varrho(v_{j_1})} \wedge U_{\varrho(v_{j_2})} \wedge \ldots \wedge U_{\varrho(v_{j_k})}) \wedge U_{\mathcal{T}}$where

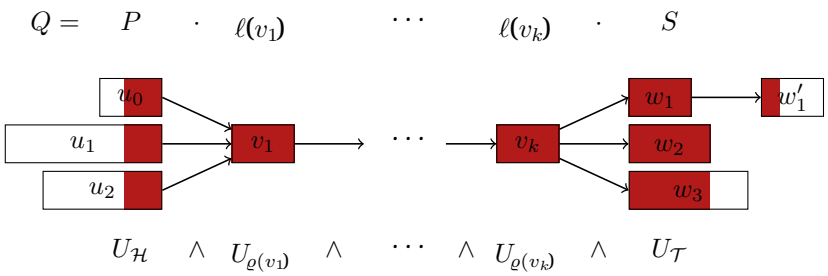

图10：半无重复 EFG 的子图 G，展示了 Q 在 G 中分解为 $P \cdot \ell(u_1) \cdots \ell(u_k) \cdot S$ 时可能出现的不同情况（以蓝色标记）。下方是对应的位向量，表示索引路径中出现 Q 对应子串的位置。它们按位与的结果给出包含 Q 的所有路径 $\mathcal{P}$。

$$U_{\mathcal{H}} := \bigvee_{(u,v_{j_1})\in\mathcal{H}} (U_{\varrho(u)} \wedge U_{\varrho(v_{j_1})}) \quad \text{and}$$
$$U_{\mathcal{T}} := \bigvee_{(v,w)\in\mathcal{T}} (U_{\varrho(v)} \wedge U_{\varrho(w)}).$$

对于边界情况：若某个 v 满足 $S \preceq \ell(v)$，且位于G的最后一个块中，则我们不使用$\mathcal{T}$，而按如下方式处理。假设$\mathcal{T}' = \{u \in V \mid S \preceq \ell(u)\}$.~此外，假设扩展前后的字典序范围分别为 $[l..\tilde{r}]$ 和 $[l'..r']$。根据 $\tilde{l}$，我们可以确定节点~ ~$u' \in B$ 的编号 $\varrho(u')$，使得 $\ell(u') \preceq S$。现在，$\mathcal{T}'$ 中的节点编号为 $\varrho(u')+l'-l$,~ ~ ~

$$U_{\mathcal{H}'(v_{j_0})} := \bigvee_{(u,v_{j_0})\in\mathcal{H}'(v_{j_0})} (U_{\varrho(u)} \wedge U_{\varrho(v_{j_0})}) \quad \text{and}$$
$$U_{\mathcal{T}(v_{j_0})} := \bigvee_{(v_{j_0},w)\in\mathcal{T}(v_{j_0})} (U_{\varrho(v_{j_0})} \wedge U_{\varrho(w)}),$$

最后 $U_{\mathcal{H}'(v_{j_0})} \wedge U_{\mathcal{T}(v_{j_0})}$。由于 $\mathcal{H}'$ 和 $\mathcal{T}$ 中的边总数至多为 $2H^2$，因此可以按公共节点对边进行划分，从而在 $O(H^2\lceil \frac{|\mathcal{P}|}{\mu} \rceil)$ 时间内完成按位运算。

□

结合引理 18 的结果与引理12中的BWT索引，我们得到以下结果。

定理 8. *[26, 定理4.11]* 假设 $G = (V, E, \ell)$ 是一个包含 b 个块的非平凡半无重复 EFG，且 P 是一组路径。可以在 $O(L|E|\log\log\sigma + |V||P|)$ 时间内对图 G 进行预处理，以构建一个大小为 $O(|V|(|P| + \log b) + |E|(L\log\sigma + \log h))$ 位的数据结构，假设 $\sigma \le L|E|$ 且 $\log\sigma \le \mu$。随后，给定模式 Q，若 Q 跨越了 G 中至少两个节点标签的串接，则可在 $O(|Q|\log\log_{2\sigma}+(|Q| + H^2)d\frac{|\mathcal{P}|}{\mu}e)$ 时间内解决路径列举问题；否则，可在 $O((|Q| + L\cdot occ)\log\log\sigma + occd\frac{|\mathcal{P}|}{\mu}e))$ 时间内解决。这里，$L = \max_{(v,w)\in E}|\ell(v)| + |\ell(w)|$，$H$ 是最大块高度，μ 是以位为单位的机器字长，occ 是 Q 在 D'_F 中的出现次数。

7 讨论

我们改进并扩展了弹性创始图的理论，克服了构建和索引表示带间隙的多序列比对的图所面临的困难：我们将支持在弹性块图上进行线性时间模式匹配的索引所占空间从 $O(NH^2)$ 改进至 $O(NH)$，其中 N 是总标签长度，H 是图的高度（定理 2）；通过在树上提出并解决独占祖先集问题（问题1和引理8），我们将对多序列比对进行分段的构造算法改进至线性时间，同时最大化块数或最小化最大分段长度，其时间复杂度分别为 $O(mn\log m)$ 和 $O(mn\log m + n\log\log n)$（推论1和推论2）；我们开发了一种 $O(mn)$ 时间的分段算法，在无间隙情形下最小化最大块高度（定理4），并将该解扩展至一般情形，使其在字母表大小为常数时可在参数化线性时间内运行，但最终为二次时间（定理 5）；我们研究了精细化指标前缀感知高度，并获得了一种 $O(mn)$ 时间的构造算法，可最小化该高度的最大值（定理6）；最后，我们展示了弹性块图索引的一种扩展，可在一组已索引的弹性创始图路径上执行路径枚举查询，进而也可在原始多序列比对序列上执行此类查询（问题2和定理 8）。

我们开发的分段算法（算法 1 和 2）接受有效分段的基于范围的描述，以及每个选定片段对应的得分。我们认为，这些算法可应用于任何涉及对齐序列最优受约束分段的场景，并可作为进一步推广到不同评分标准的基础。

如果所选的半无重复片段包含在多序列比对中拼写相同字符串的比对子串，那么弹性块图自然会对这些片段进行压缩。我们研究的优化指标——块数、片段长度和块高度——针对弹性块图的特定特征，可视为压缩图索引的一种启发式解：事实上，定理 2 的空间上界 $O(NH)$ 等价于 $O(L|E|)$，其中 $L = \max_{v \in V} |\ell(v)|$ 是最大标签长度。不过，索引的确切大小（即第3节中的字符串 D_R 和 D_F ）本身也可视为一种压缩指标，因此我们提出以下分段问题。

问题 3（总边长分段问题）。 求 *MSA* 的半无重复分段 S ，使生成的创始图 $G(S) = (V, E, \ell)$ 的边标签总长度 $\sum_{(u,\ v) \in E} |\ell(u)| + |\ell(v)|$ 最小。

此外，EFG 框架还存在一些限制：

- 如果某个片段包含一个字符串 T，而该字符串也在 MSA 的其他位置拼写出来，那么 MSA 片段的半无重复性质就不成立。具体来说，考虑在 MSA 的列 o_1 和 o_2处出现的字符串 $T[1..t]$：任何满足 $o_1 \le x < y \le o_1 + t$ 或 $o_2 \le x < y \le o_2 + t$的片段 $[x..y]$ 都不可能是半无重复的，这会限制任何可索引分段的重组与压缩。第 3 节的索引结果表明，边标签 $\ell(u)\ell(v)$可以有效模拟 EFG 的导航，因此，未来的研究方向可以探讨如何利用用于索引重复文本集合（在此处为字符串$\ell(u)\ell(v)$）的方法和数据结构，进一步压缩 EFG；

- MSA 的分段不允许包含空字符串的块（假设 1），这限制了半无重复片段在 MSA 长间隙游程中的作用。沿此方向扩展我们的结果——允许空字符串，并在 EFG 中将其剥除，同时仍保留对应于原始序列的路径——自然会得到一个创始有向无环图，而非弹性块图。这也是我们正在进行的研究主题。

我们对上述想法的初步实现可见于 `https://github.com/tsnorri/founder-graphs-semi-repeat-free`。我们正在将其与其他适用于此目的的算法实现进行比较。

参考文献

[1] Mai Alzamel、Lorraine A. K. Ayad、Giulia Bernardini、Roberto Grossi、Costas S. Iliopoulos、Nadia Pisanti、Solon P. Pissis 和 Giovanna Rosone。《退化字符串比较》。*Fundamenta Informaticae*，175(1-4)：41–58，2020。

[2] Djamal Belazzougui、Fabio Cunial、Juha K¨arkk¨ainen 和 Veli M¨akinen。小空间中的线性时间字符串索引与分析。*ACM* 算法汇刊，16(2)，2020 年 3 月。`doi:10.1145/3381417`。

[3] Djamal Belazzougui、Dmitry Kosolobov、Simon J. Puglisi 和 Rajeev Ra- man。后缀树中的加权祖先问题再探。载于 Pawel Gawrychowski 和 Tatiana Starikovskaya 编，第 *32* 届年度组合模式匹配研讨会，*CPM 2021*，*2021* 年 *7* 月 *5* 日至 *7* 日，波兰弗罗茨瓦夫，LIPIcs 第 191 卷，第 8:1–8:15 页。达格施图尔—莱布尼茨信息学中心，2021 年。 doi:10.4230/LIPIcs.CPM.2021.8。

[4] Giulia Bernardini、Pawel Gawrychowski、Nadia Pisanti、Solon P. Pissis 和 GiovannaRosone。通过快速矩阵乘法实现更快的弹性简并字符串匹配。载于 Christel Baier、Ioannis Chatzigiannakis、Paola Flocchini 和 Stefano Leonardi 编，第*46*届国际自动机、语言与程序设计研讨会（*ICALP 2019*），莱布尼茨国际信息学论文集（*LIPIcs*）第132卷，页码 21:1– 21:15，德国达格施图尔，2019年。Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik。URL: http://drops.dagstuhl.de/opus/volltexte/2019/10597, doi:10.4230/LIPIcs.ICALP.2019.21。

[5] M. Burrows 和 D. Wheeler。块排序无损数据压缩算法。技术报告 124，数字设备公司，1994年。

[6] Bastien Cazaux、Dmitry Kosolobov、Veli M¨akinen 和 Tuukka Norri。《列流模型中的线性时间最大分段问题》。载于 Nieves R. Brisaboa 和 Simon J. Puglisi 编，《字符串处理与信息检索——第 26 届国际研讨会，SPIRE 2019，西班牙塞哥维亚，2019 年 10 月 7—9 日，论文集》，《计算机科学讲义》第 11811 卷，第 322–336 页。施普林格，2019年。

[7] Dustin Cobas、Veli M¨akinen 和 Massimiliano Rossi。《面向宏基因组学应用的文档列举 r-index 定制》。载于 Christina Boucher 和 Sharma V. Thankachan 编，《字符串处理与信息检索——第 27 届国际研讨会，SPIRE 2020，美国佛罗里达州奥兰多，2020 年 10 月 13—15 日，论文集》，《计算机科学讲义》第 12303 卷，第 291–306 页。施普林格，2020年。 doi:10.1007/978-3-030-59212-7_21。

[8] Dustin Cobas 和 Gonzalo Navarro。在重复文本集合上进行快速、小巧且简单的文档列举。载于 Nieves R. Brisaboa 和 Simon J. Puglisi（编），字符串处理与信息检索——第*26*届国际研讨会，*SPIRE 2019*，西班牙塞哥维亚，*2019*年*10*月*7*日至*9*日，论文集，《计算机科学讲义》第11811卷，第482–498页。施普林格，2019年。 doi:10.1007/978-3-030-32686-9_34。

[9] Rene De La Briandais。使用可变长度键进行文件搜索。载于*1959*年*3*月*3*日至*5*日西部联合计算机会议上发表的论文，IRE-AIEE-ACM ’59（西部），第295–298页，美国纽约州纽约市，1959年。计算机协会。 doi:10.1145/1457838.1457895。

[10] Hannes P. Eggertsson、Snaedis Kristmundsdottir、Doruk Beyter、Hakon Jonsson、Astros Skuladottir、Marteinn T. Hardarson、Daniel F. Gud- bjartsson、Kari Stefansson、Bjarni V. Halldorsson 和 Pall Melsted。Graphtyper2 利用泛基因组图实现群体规模的结构变异基因分型。《自然通讯》，10(1):5402，2019年11月doi:10.1038/s41467-019-13341-9。

[11] Massimo Equi。带标记图中字符串匹配的上下界。博士论文，赫尔辛基大学，2022年6月[12] Massimo Equi、Roberto Grossi、Veli M¨akinen 和 Alexandru I. Tomescu。图的字符串匹配复杂度。载于 Christel Baier、Ioannis Chatzigiannakis、Paola Flocchini 和 Stefano Leonardi 编，第*46*届自动机、语言与编程国际研讨会（*ICALP 2019*），*2019*年*7*月*9*日至*12*日，希腊帕特雷，第132卷，*LIPIcs*，第55:1–55:15页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2019年 doi:10.4230/LIPIcs.ICALP.2019.55。[13] Massimo Equi、Veli M¨aki nen 和 Alexandru I. Tomescu。除非强指数时间假设不成立，否则无法在多项式时间内为图建立索引以支持亚二次时间的字符串匹配。载于*SOFSEM 2021*：计算机科学理论与实践，第608–622页，Cham，2021年。施普林格国际出版社。[14]Massimo Equi、Tuukka Norri、Jarno Alanko、Bastien Cazaux、Alexan- dru I. Tomescu 和 Veli M¨akinen。创始图索引的算法与复杂性。*Algorithmica*，2022年7月。doi:10.1007/s00453-022-01007-w。[15] Martin Farach。大字母表下的最优后缀树构建。载于第*38*届计算机科学基础年度研讨会论文集，第137–143页。IEEE，1997年。[16] Michael L. Fredman 和 Dan E. Willard。借助融合树突破信息论屏障。载于 Harriet Ortiz 编，第*22*届 *ACM* 计算理论年度研讨会论文集，*1990*年*5*月*13*日至*17*日，美国马里兰州巴尔的摩，第1–7页。ACM，1990年。 doi:10.1145/100216.100217。[17] Erik Garrison、Jouni Sir´en、Adam Novak、Glenn Hickey、Jordan Eizenga、Eric Dawson、William Jones、Shilpa Garg、Charles Markello、Michael Lin 和 Benedict Paten。变异图工具包通过在参考序列中表示遗传变异，改进了读段比对。《自然·生物技术》，36，2018年8月。 doi:10.1038/nbt.4227。[18] Dan 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。 doi:10.1017/cbo9780511574931。[19] G. Jacobson。节省空间的静态树和图。载于*FOCS* 论文集，第549–554页，1989年。[20] Daehwan Kim、Joseph Paggi、Chanhee Park、Christopher Bennett 和 Steven Salzberg。基于图的基因组比对和基因分型：使用 HISAT2 和 HISAT-genotype。《自然·生物技术》，37:1，2019年8月。 doi:10.1038/s41587-019-0201-4。[21] Moritz G. Maaß。词缀树的线性双向在线构建。*Algorithmica*，37(1):43–74，2003年。 doi:10.1007/s00453-003-1029-2。

[22] Veli M¨akinen、Bastien Cazaux、Massimo Equi、Tuukka Norri 和 Alexan- dru I. Tomescu。可索引创始区块图的线性时间构建。载于 Carl Kingsford 和 Nadia Pisanti 编，生物信息学算法第20届国际研讨会 WABI 2020，2020年9月7日至9日，意大利比萨（线上会议），LIPIcs 第172卷，页码 7:1–7:18。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2020年。doi:10.4230/LIPIcs.WABI.2020.7.[23] Veli M¨akinen、Gonzalo Navarro、Jouni Sir´en 和 Niko V¨alim¨aki。高度重复序列集合的存储与检索。《计算生物学杂志》，17(3):281–308，2010年。[24] Udi Manber 和 Eugene W. Myers。后缀数组：一种在线字符串搜索的新方法。《SIAM计算杂志》，22(5):935–948，1993年。 doi:10.1137/0222058。[25] S. Muthukrishnan。文档检索问题的高效算法。载于《第十三届ACM-SIAM离散算法年度研讨会论文集》，SODA '02，页码 657–666，美国，2002年。工业与应用数学学会。[26] Tuukka Norri。《对齐文本的创始分段》。博士论文，赫尔辛基大学，2022年6月。[27]Tuukka Norri、Bastien Cazaux、Saska D¨onges、Daniel Valenzuela 和 Veli M¨akinen。创始重构实现可扩展且无缝的泛基因组分析。《生物信息学》，37(24):4611–4619，2021年7月。 arXiv:https://academic.oup.com/bioinformatics/article-pdf/37/24/4611/41726965/btab516.pdf， doi:10.1093/bioinformatics/btab516。[28] Tuukka Norri、Bastien Cazaux、Dmitry Kosolobov 和 Veli M¨akinen。线性时间最小分段实现可扩展的创始重构。《分子生物学算法》，14(1):12:1–12:15，2019年。[29] Nicola Rizzo 和 Veli M¨akinen。最小高度的可索引弹性创始图。载于 Hideo Bannai 和 Jan Holub 编，第33届组合模式匹配年度研讨会 CPM 2022，2022年6月27日至29日，捷克共和国布拉格，LIPIcs 第223卷，页码 19:1–19:19。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2022年。doi:10.4230/LIPIcs.CPM.2022.19。[30] Nicola Rizzo 和 Veli M¨akinen。可索引弹性创始图的线性时间构建。载于 Cristina Bazgan 和 Henning Fernau 编，组合算法——第33届国际研讨会 IWOCA 2022，德国特里尔，2022年6月7日至9日，论文集，《计算机科学讲义》第13270卷，页码 480–493。施普林格，2022年。 doi:10.1007/978-3-031-06678-8_35。[31]Korbinian Schneeberger、J¨org Hagmann、Stephan Ossowski、Norman Warthmann、Sandra Gesing、Oliver Kohlbacher 和 Detlef Weigel。针对多个基因组同时比对短读段。《基因组生物学》，10:R98，2009年。

[32] Harold Herbert Seward。电子数字计算机在商业运营中的应用：信息排序。技术报告 R-232，麻省理工学院数字计算机实验室，1954年。论题。[33] Jouni Sir´en、Niko V¨alim¨aki 和 Veli M¨akinen。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》，11(2):375–388，2014年。[34] Dirk Strothmann。词缀数组数据结构及其在 RNA 二级结构分析中的应用。《理论计算机科学》，389(1-2):278–294，2007年。 doi:10.1016/j.tcs.2007.09.029。[35]计算泛基因组学联盟。计算泛基因组学：现状、前景与挑战。*Briefings Bioinform.*，19(1):118–135，2018年。 doi:10.1093/bib/bbw089。[36]Esko Ukkonen。后缀树的在线构建。*Algorithmica*，14(3):249–260，1995年。 doi:10.1007/BF01206331。