arXiv:2302.04229v1 [cs.DS] 8 Feb 2023

加权　编辑距离计算：字符串、树与 Dyck

Debarati Das[1], Jacob Gilbert[2], MohammadTaghi Hajiaghayi[2], Tomasz Kociumaka[3], 和 Barna Saha[4]

[1]宾夕法尼亚州立大学，美国
debaratix710@gmail.com[2]马里兰大学，美国
jgilber8@umd.edu hajiaghayi@gmail.com[3]马克斯·普朗克信息学研究所，德国
tomasz.kociumaka@mpi-inf.mpg.de[4]加州大学圣迭戈分校，美国 barnas@ucsd.edu

摘要

给定两个长度为 n、字母表为 Σ 的字符串，以及它们编辑距离的上界 k，约四十年前 Myers（Algorithmica'86）以及 Landau 和 Vishkin（JCSS'88）提出的算法，可在 $\mathcal{O}(n+k^2)$ 时间内计算无权字符串编辑距离。迄今为止，它仍是精确计算编辑距离的最快算法，并且在强指数时间假设下是最优的（STOC'15）。多年来，这一结果启发了许多进展，˜包括字符串编辑距离的快速近似算法，以及用于推广到树编辑距离和 Dyck 编辑距离的类似 $\mathcal{O}(n+\text{poly}(k))$ 时间算法。令人惊讶的是，所有这些结果都仅适用于无权实例。

虽然无权编辑距离在理论上具有基础性，但几乎所有现实应用都需要加权编辑距离，即为不同的编辑操作（插入、删除和替换）赋予不同权重，并且权重可能随被编辑的字符而变化。给定一个权重函数 $w\colon \Sigma\cup\{\varepsilon\}\times\Sigma\cup\{\varepsilon\}\to\mathbb{R}_{\geq 0}$（满足 $w(a,a)=0$且 $w(a,b)\geq 1$，对所有满足 $a\neq b$ 的 $a,b\in\Sigma\cup\{\varepsilon\}$ 均成立），目标是找到一个使编辑总权重最小的对齐。除了普通的 $\mathcal{O}(n^2)$ 时间动态规划算法及其几乎显而易见的 $\mathcal{O}(nk)$ 时间实现（其中 k 是所求总权重的上界）之外，前述关于无权编辑距离的进展都不适用于加权情形。本文提出了首个能够精确计算加权字符串编辑距离的 $\mathcal{O}(n+\text{poly}(k))$ 时间算法，从而弥合了数十年来我们对无权与加权编辑距离的认识之间一个根本性的差距。随后，我们将这一结果推广到加权树编辑距离和 Dyck 编辑距离，并引入若干新技术，由此得到一种确定性算法，即便对于无权树编辑距离，其性能也优于以往工作。鉴于加权编辑距离具有根本重要性，我们相信，我们的加权编辑距离 $\mathcal{O}(n+\text{poly}(k))$ 算法将有助于推动该领域进一步取得重大进展。

1 引言

字符串编辑距离及其若干变体自20世纪60年代以来已被研究了数十年 [Lev65、NW70、WF74]。从历史上看，为了简化问题并使理论结果更加简洁，大多数关于这些问题的研究都假设编辑操作具有单位权重。直到今天，

无权编辑距离的最快精确算法归功于 Myers [Mye86] 以及 Landau 和 Vishkin [LV88], , 他们针对两个长度为 n 的字符串，在其编辑距离上界为 k 的情况下，得到了一个 $\mathcal{O}(n + k^2)$ 时间的解法。根据强指数时间假设 [BI18]，现已知这一界限是最优的（除了次多项式因子之外）。多年来， [Mye86, LV88] 的圣杯级结果启发了许多关于字符串编辑距离（无权）的快速近似算法的研究 [CDG+20、GKS19、GRS20、KS20a] ，以及针对（无权）Dyck 和树编辑距离等推广问题的类似 $\tilde{\mathcal{O}}(n + \text{poly}(k))$ 时间[1] 算法 [BO16、FGK+22a、DGH+22]。然而，几乎所有现实世界的应用都需要加权编辑距离，即为不同的编辑操作（插入、删除和替换）分配不同的权重，并且权重可能随所编辑的字符而变化 [WF74、ZS89、Kur96、Gus97、PM02、JM09、FFF+16、Ski20、KXI20、GWK21、CIG22]。因此，既有研究的理论结果与这些结果在现实世界中的实用性之间存在巨大差距。在本文中，我们弥合了对无权和加权编辑距离理解之间的这一根本性鸿沟：我们提出了首个计算加权编辑距离及其推广——加权树编辑距离和 Dyck 树编辑距离的非平凡算法。

更具体地说，在本文中，我们提出了首个 $\mathcal{O}(n + \text{poly}(k))$时间算法，用于精确计算加权编辑距离。给定权重函数 $w\colon \Sigma \cup \{\varepsilon\} \times \Sigma \cup \{\varepsilon\} \to \mathbb{R}_{\geq 0}$(规范化后满足 $w(a, b) \geq 1$ for $a \neq b$)，目标是在总权重不超过给定阈值 k的前提下，找到一个使编辑操作（插入、删除和替换）的总权重最小的对齐。值得注意的是，除了常规的 $\mathcal{O}(n^2)$时间动态规划算法及其几乎显而易见的 $\mathcal{O}(nk)$时间实现之外，前述关于无权编辑距离的所有进展都不适用于这一加权变体。随后，我们将结果推广到加权树编辑距离和 Dyck 编辑距离，并引入了若干新技术，甚至改进了无权树编辑距离问题的求解：作为我们结果的副产品，我们给出一个确定性的$\mathcal{O}(n+k^7 \log\ k)$时间解法，其速度远快于 Das、Gilbert、Hajiaghayi、Kociumaka、Saha 和 Saleh [DGH+22]提出的随机化 $\mathcal{O}(n \log\ n+k^{15} \log^2 k)$时间算法。

加权编辑距离计算看似停滞不前，这能解释吗？正如我们稍后将看到的，即使是单调性这类基本性质——它对高效计算无权编辑距离至关重要 [Mye86, LV88], ——在考虑加权操作时也会失效。这使得局部匹配方法无法适用，而这种方法似乎是实现有界（无权）编辑距离线性时间算法所必需的 [Mye86, LV88, BO16, FGK+22a, DGH+22]；相反，我们需要从全局角度考察序列，以找到匹配子串，同时保持线性运行时间。面对这些障碍，我们最大的贡献是提出了一种加权编辑距离核化方法，不仅适用于字符串，也适用于树和 Dyck 编辑距离实例。有趣的是，我们的核与权重无关，也就是说，核化算法无需知道权重函数 w。鉴于加权编辑距离的基础性，我们相信，我们用于加权编辑距离的 $\mathcal{O}(n + \text{多项式}(k))$ 算法将有助于推动该领域进一步取得重大进展。

1.1 相关工作

字符串编辑距离： 编辑距离是计算机科学中最基础的问题之一，自20世纪60年代以来便一直受到研究 [Lev65、NW70、WF74]。在无权编辑距离问题中，给定两个长度至多为 n的字符串，目标是找出将一个字符串转换为另一个字符串所需的最少编辑操作次数（插入、删除和替换）。给定一个参数 k 作为编辑距离的上界，一种由

[1] $\tilde{\mathcal{O}}(\cdot)$ 记号省略了输入规模 n 的多对数因子。

Myers [Mye86] 和 Landau 和 Vishkin [LV88] 通过将后缀树与一种巧妙的贪心方法相结合，以 $\mathcal{O}(n+k^2)$ 的时间完成这一任务。只要 $k=\mathcal{O}(\sqrt{n})$，上述算法的运行时间就与 n 成线性关系。对于更大的 k，人们已广泛研究编辑距离的近似算法 [LMS98, Ind01, BYJKK04, BES06, AO09, AKO10], ，尤其是近来 [HRS19, CDG+20, GRS20, KS20b, BR20, BEG+21]。这一研究最终取得了目前由 Andoni 和 Nosatzki 给出的最佳界 [AN20], ：他们针对任意常数 $\epsilon>0$，得到了运行时间为 $\mathcal{O}(n^{1+\epsilon})$ 的常数因子近似算法。这些工作都要求满足单调性，并假设如果在两个字符串中都添加匹配后缀，就能轻易扩展一个最优解；而在加权编辑距离实例中，这些条件未必成立。因此，Kuszmaul [Kus19], 提出的加权编辑距离当前最佳近似算法，给出的权衡要差得多：对于任意 $0\le\tau\le 1$，该算法在 $\tilde{\mathcal{O}}(n^{2-\tau})$ 时间内提供 $\mathcal{O}(n^{\tau})$ 倍近似。

树编辑距离：树编辑距离问题最早由 Selkow [Sel77], 提出，是编辑距离的一种推广，其任务是计算两棵带有节点标签的有根有序树之间的差异度量。在无权树编辑距离版本中，每次节点插入、删除或重新标记操作的代价均为单位代价。该问题有许多应用，包括编译器优化 [DMRW10], 、结构化数据分析 [Cha99, BGK03, FLMM09],、图像分析 [BS98], 以及计算生物学 [HT84, SZ90, Gus97, LMS98, Bil05]。目前，求解精确树编辑距离算法的最佳运行时间界由 D¨urr [D¨ur22] 给出；经过一系列改进，从 $\mathcal{O}(n^6)$ [Tai79]，到 $\mathcal{O}(n^4)$ [ZS89], ，再到 $\mathcal{O}(n^3\log n)$ [Kle98], 、 $\mathcal{O}(n^3)$ [DMRW10], ，以及 $\mathcal{O}(n^{2.9546})$ [Mao21]，他为该问题提出了一个 $\mathcal{O}(n^{2.9149})$ 时间算法。此外，Boroujeni、Ghodsi、Hajiaghayi 和 Seddighin [BGHS19]给出了一种树编辑距离的 $(1+\epsilon)$ 近似算法，运行时间为 $\tilde{\mathcal{O}}(n^2)$。最近，Seddighin 和 Seddighin [SS22] 提出了一种用于树编辑距离的 $\mathcal{O}(n^{1.99})$ 时间 $(3+\epsilon)$ 近似算法 ˜（该算法建立在 [BGHS19] 此前提出的、运行时间为 $\mathcal{O}(n)$ 的 $\mathcal{O}(\sqrt{n})$ 因子近似算法之上）。此外，Das、Gilbert、Hajiaghayi、Kociumaka、Saha 和 Saleh [DGH+22]提出了一种 $\tilde{\mathcal{O}}(n+k^{15})$ 时间算法，用于求解距离有上界时的精确树编辑距离（另请参阅 Akmal 和 Jin [AJ21], 提出的 $\tilde{\mathcal{O}}(nk^2)$ 时间算法；该算法改进了此前用于有界树编辑距离问题的、运行时间为 $\mathcal{O}(nk^3)$ 的算法 [Tou05]）。

就加权树编辑距离而言，最快的算法由 Demaine、Mozes、Rossman 和 Weimann 提出，[DMRW10], 其时间复杂度为 $\mathcal{O}(n^3)$ 时间，这与 Bringmann、Gawrychowski、Mozes 和 Weimann 提出的条件下界相匹配，[BGMW20]（该下界此前由 Abboud [Abb14]猜想）。具体而言，除非全点对最短路径问题存在真正次立方时间的解法，否则不存在真正次立方时间的加权树编辑距离算法。除非加权 k-团问题存在一个 $\mathcal{O}(n^{k-\epsilon})$ 时间算法，否则对于常数大小字母表上的树，该下界仍然成立。

Dyck 编辑距离：Dyck 编辑距离问题是编辑距离的另一种变体，属于通用语言编辑距离的范畴 [AP72, Mye95, Sah17, BGSW19]，并有许多实际应用，例如修复层次化数据文件，尤其是 XML 和 JSON 文件 [Har78, Koz97]。在该问题的无权版本中，给定一个由 n 括号组成的字符串，目标是找出使该字符串达到平衡所需的最少编辑次数（字符插入、删除和替换）。针对该问题的精确算法 [BGSW19, CDX22, D¨ur22] 和近似算法 [Sah14, DKS22] 均已有研究成果。求解精确 Dyck 编辑距离至少与布尔矩阵乘法一样困难 [ABW18]。有界 Dyck 编辑问题近年来也受到多项研究：Backurs 和 Onak [BO16] 提出了首个运行时间为 $\mathcal{O}(n+k^{16})$ 的算法，随后该算法被改进至 $\mathcal{O}(n+k^5)$ [FGK+22a],，最终借助快速矩阵乘法将运行时间降至 $\mathcal{O}(n+k^{4.5442})$ [FGK+22b, D¨ur22]。除了

$\mathcal{O}(n^3)$ 时间的语言编辑距离精确算法 [Mye95], 这些结果不适用于加权情形。

1.2 我们的贡献

本文的主要贡献是提出了用于加权字符串、树和 Dyck 编辑距离的新算法。我们将权重函数定义为函数 $w: \Sigma \cup \{\varepsilon\} \times \Sigma \cup \{\varepsilon\} \to \mathbb{R}_{\geq 0}$ such that$w(a,a) = 0$ and $w(a,b) \geq 1$ for $a \neq b$。如果 $a, b \in \Sigma$，那么 $w(a,\varepsilon)$ 是删除 a 的代价，$w(\varepsilon,b)$ 是插入 b 的代价，而 $w(a,b)$ 是将 a 替换为 b 的代价。假设 $w(a,a) = 0$ 表示匹配符号可以零代价对齐，而假设 $w(a,b) \geq 1$for $a \neq b$ 则表示权重已规范化，使得每次编辑的代价至少为一。如果权重函数还满足三角不等式（我们对树和 Dyck 编辑距离作此假设），则称其为拟度量。在权重计算方面，我们考虑任何均匀模型：在该模型中，实数仅进行比较和加法 [PR05], ，例如 RAM 模型。

我们将 $\mathsf{ed}^w(X,Y)$ 定义为字符串 X 和 Y在权重函数 w下对齐的最小代价。此外，我们将 $\mathsf{ed}^w_{\leq k}(X,Y)$ 定义为 $\mathsf{ed}^w(X,Y)$（如果其值至多为 k），否则为 ∞。我们给出了首个加权有界编辑距离算法，其运行时间为 $\mathcal{O}(n+\mathsf{poly}(k))$。

定理 1.1. 给定长度至多为 n 的字符串 X,Y、一个整数 $k \in \mathbb{Z}_+$ 和一个权重函数 w，可以在 $\mathcal{O}(n+k^5)$ 时间内计算 $\mathsf{ed}^w_{\leq k}(X,Y)$。

与字符串编辑距离类似，我们定义 $\mathsf{ted}^w(F,G)$ 为森林 F和 G 在权重函数 w下树对齐的最小代价。我们类似地定义 $\mathsf{ted}^w_{\leq k}(F,G)$，并给出首个加权树编辑距离算法，其运行时间为 $\mathcal{O}(n+\mathrm{poly}(k))$。在无权情形下，我们的确定性算法显著快于文献 [DGH+22] 中最先进的随机化算法。

定理 1.2. 给定长度至多为 n 的森林 F,G、一个整数 $k \in \mathbb{Z}_+$ 和一个拟度量 w，可以在 $\mathcal{O}(n+k^{15})$ 时间内计算 $\mathsf{ted}^w_{\leq k}(F,G)$。此外，可以在 $\mathcal{O}(n+k^7 \log k)$ 时间内计算 $\mathsf{ted}^{\leq k}(F,G)$。

最后，我们定义 $\mathsf{Dyck}^w_{\leq k}(X)$ 为 X 与一个 编辑$^w_{\leq k}(X,Y)$ 之间的最小距离，其中该字符串Y 属于 Dyck 语言。我们给出了首个加权 Dyck 编辑距离算法，其运行时间为$\mathcal{O}(n+\mathrm{poly}(k))$。在这种设定下，字母表由左括号和右括号组成。除了满足三角不等式外，我们还需要假设权重函数对同类型的左括号和右括号赋予相近的权重。这一性质可用偏斜度量来刻画，相关定义见第 节，正式定义于第 4.2节。

定理 1.3. 给定长度为 n 的字符串 X、一个整数 $k \in \mathbb{Z}_+$ 和一个偏斜度量 w，可以在 $\mathcal{O}(n+k^{12})$ 时间内计算 $\mathsf{dyck}^w_{\leq k}(X)$。

我们注意到，尽管我们的算法假设 k 已知，但也可以得到与定理 1.1 至 1.3 类似的运行时间，只需用目标距离替代阈值 k。为此，只需从使运行时间达到 k 的最大值 $\mathcal{O}(n)$ 开始，例如，$k = \Theta(n^{1/5})$（对于字符串），并持续将阈值 k 加倍，直到算法输出 ∞。首次得到的有限结果必定就是目标距离；由于后续迭代的运行时间构成几何级数，总运行时间由最后一次迭代决定，此时 k 至多是目标距离的两倍。

1.3 概述

用于计算无权和加权实例编辑距离的传统算法采用动态规划，运行时间为 $\mathcal{O}(n^2)$。给定两个字符串 X 和 Y，动态规划表 D 中的表项 $D[i,j]$ 存储字符串 X 和 Y 的前缀分别截至索引 i 和 j 时的加权（无权）编辑距离。也就是说 $D[i,j] := \mathsf{ed}(X[0\,..\,i)\ Y[0\,..\,j))$. 然后

$$D[i+1,j+1] = \min\{D[i,j+1]+1, D[i+1,j]+1, D[i,j]+\delta(X[i],Y[j])\}\text{：无权编辑距离}$$

$$D[i+1,j+1] = \min\{D[i,j+1]+w(X[i],\varepsilon), D[i+1,j]+w(\varepsilon,Y[j]), D[i,j]+w(X[i],Y[j])\}.$$
：加权编辑距离

递归定义中的第一项对应于删除 $X[i]$，第二项对应于插入 $Y[j]$，第三项则对应于匹配或替换（如果 $X[i] = Y[j]$，则 $\delta(X[i],Y[j])$ =0；否则 $\delta(X[i],Y[j]) = 1$）。显然，$D[|X|,|Y|]$ 等于 X 和 Y 之间的加权（无权）编辑距离总和，并且可以在 $\mathcal{O}(n^2)$ 时间内计算得出。

如果加权（无权）编辑距离有上界 $k < n$，则可以将运行时间改进为 $\mathcal{O}(nk)$。在这种情况下，只需计算 D 主对角线周围仅有 $2k+1$ 条对角线所对应的项。然而，无权和加权编辑距离计算的发展过程之间的相似之处到此为止。

无权编辑距离计算的首个重大突破出现在 20 世纪 80 年代末 [Mye86, LV88]。当编辑距离不超过 k 时，人们提出了一种运行时间为 $\mathcal{O}(n+k^2)$ 的无权编辑距离算法，由此得到了一种用于 $k \le \sqrt{n}$ 的线性时间算法。该算法利用了无权编辑距离的两个简单而强大的性质，即 (i) 单调性：$D[i+1,j+1] \ge D[i,j]$；以及 (ii) 贪心扩展：如果 $X[i] = Y[j]$，那么 $D[i+1,j+1] = D[i,j]$。这两个性质共同意味着，如果我们能通过预处理步骤找出 X 和 Y 中的最长相等子串，那么只需计算 D 中的 $\mathcal{O}(k^2)$ 个条目。更准确地说，对于 $2k+1$ 条对角线中的每一条，只需计算其中至多 $k+1$ 个满足 $k+1 \ge D[i+1,j+1] > D[i,j]$ 的条目。预处理步骤利用线性时间构造后缀树，从而能够在常数时间内回答任意最长相等子串查询，使总体运行时间达到 $\mathcal{O}(n+k^2)$。此后所有针对无权字符串编辑距离的快速近似算法，都无一例外地依赖上述两个性质。

遗憾的是，上述两个性质均不适用于加权编辑距离计算。以下简单示例将清楚地说明这一点。

1. 不具有单调性：设 $X = \mathsf{ab}$、$Y = \mathsf{c}$ 和 $w(\mathsf{a},\mathsf{c}) + w(\mathsf{b}\ \varepsilon) < w(\mathsf{a}\ \varepsilon)$。则 $D[1,0] = w(\mathsf{a}\ \varepsilon)$ 和 $D[2,1] = w(\mathsf{a},\mathsf{c}) + w(\mathsf{b}\ \varepsilon) < w(\mathsf{a}\ \varepsilon)$。2. 不存在贪心扩展：设 $X = \mathsf{ab}$、$Y = \mathsf{b}$ 和 $w(\mathsf{a},\mathsf{b}) + w(\mathsf{b}\ \varepsilon) < w(\mathsf{a}\ \varepsilon)$。将 a 替换为 b，并从 X 中删除 b，比删除 a 并匹配后续的 b 更便宜。

从某种意义上说，这解释了该领域在加权实例方面进展缓慢的原因。我们需要截然不同的方法和新的思路。

当 k，即两个输入字符串之间的最小加权编辑距离较小时，显然，输入字符串中的大多数字符都能完美匹配，因此在编辑距离计算中不会产生任何代价。我们算法的主要思想是为输入字符串寻找较小的代表性实例，然后在这些代表性实例上运行 $\mathcal{O}(nk)$ 时间的加权编辑距离算法，以求得原始的加权编辑距离。事实上，我们证明，有界加权编辑距离的任意实例都可以通过大小为 $\mathcal{O}(k^4)$ 的字符串求解。我们的算法从大小为 $\mathcal{O}(n)$ 的字符串中构造出大小为 $\mathcal{O}(k^4)$ 的核，耗时 $\mathcal{O}(n)$；随后，利用动态规划的 $\mathcal{O}(nk)$ 时间加权扩展算法，可以在 $\mathcal{O}(k^5)$ 时间内求解得到的较小实例。

我们还将核化的思想扩展到树编辑距离和 Dyck 编辑距离的加权实例，首次给出了 $\mathcal{O}(n + \mathrm{poly}(k))$算法。值得注意的是，我们的算法是确定性的，并且相较于近期关于无权树编辑距离的随机化算法 [DGH+22] 有显著改进。我们证明，可以在线性时间内，从每个问题的原始实例中计算出较小的 $\mathcal{O}(\mathrm{poly}(k))$大小的核，然后运行基于动态规划的算法来计算最终的编辑距离值。

为了找到这样的核，我们利用在两个输入字符串 X 和 Y中同步出现的子串，也就是说，它们分别在 X 和 Y 中的位置 x 和 y处出现，并满足$|x - y| = \mathcal{O}(k)$。我们的核化算法首先尝试用 $\mathcal{O}(k)$ 对同步出现来覆盖输入字符串（几乎全部）。如果无法做到这一点，我们便得出结论：编辑距离必定很大，即 $\mathsf{ed}^{w}_{\leq k}(X, Y) = \infty$。否则，我们会采用一种新颖的编辑距离等价关系概念，使得子串 P 的同步出现可以替换为等价子串 P' 的同步出现，而不影响编辑距离 $\mathsf{ed}^{w}_{\leq k}(X, Y)$。为此，我们提供一种线性时间算法，它能针对任意字符串 P，计算出一个与其编辑距离等价、大小为 $\mathsf{poly}(k)$ 的字符串 P' 。

等价片段的类似概念也是我们针对加权树编辑距离和 Dyck 编辑距离的算法的核心。我们的三个算法都采用以下高级步骤：

1. 将输入对象划分为 $\mathcal{O}(k)$ 个片段，其中大多数可以两两配对，形成同步出现。2. 如果算法未能找到足够长的同步出现，则报告编辑距离超过 k。3. 否则，对于每一对同步出现，用一个较短的等价替代片替换原片段。4. 使用已知的动态规划算法求解所得的小规模实例。

1.3.1 加权字符串编辑距离

我们现在介绍如何获得定理 通过实现上述高层方案来证明定理 1.1 .

编辑距离等价字符串 我们的加权编辑距离算法背后最大的技术贡献，是一个线性时间过程（推论 2.14）：给定字符串 P，计算出一个长度为 $\mathcal{O}(k^3)$ 的等价字符串。在第一阶段，该过程消除 k-周期性：只要处理中的字符串包含形如 Q^{4k+1} 且满足 $|Q| \in [1\,..\,2k]$ 的片段，就将该片段替换为 Q^{4k}。如引理 2.9 所示，字符串 Q^{4k+1} 和 Q^{4k} 等价，因此这一步保持了它与输入字符串 P 的等价性。最终，第一阶段得到一个避免 k-周期性且与 P 等价的字符串（示例见图 1）。该过程在引理 2.13 中实现，其中底层算法从左到右处理输入字符串 P，并针对每个遇到的形如 Q^{4k+1} 且满足 $|Q| \in [1\,..\,2k]$ 的片段，删除 Q 的第一个副本。

在引理 2.11 中，我们证明：若 P 避免 k 周期性且满足 $|P| \geq 42k^3$，则它等价于 $P[0\,..\,21k^3) \cdot P[|P| - 21k^3\,..\,|P|)$，也就是说，等价于将其长度为 $21k^3$ 的前缀与长度为 $21k^3$ 的后缀串接起来（删除中间的字符）。为此，我们考虑字符串 X 和 Y 中 P 的同步出现，以及一个代价为 $\mathsf{ed}^{w}(X, Y) \leq k$、将 X 映射到 Y 的最优对齐 $\mathcal{A}$ 。我们注意到，$\mathcal{A}$ 必须与 P 在 X 中出现位置的长度为 $21k^3$ 的前缀内一个长度为 $10k^2$ 的片段完全匹配。此外，由于 P 避免 k 周期性，$\mathcal{A}$ 只能将该片段与 P 在 Y 中出现位置的对应片段匹配。对称地，$\mathcal{A}$ 必须与 P 的长度为 $21k^3$ 的后缀内两个长度为 $10k^2$ 的片段副本匹配。由此可知，对于某个值，$\mathcal{A}$ 将 $P[d\,..\,|P| - e)$ 的两个副本对齐

d， $e \in [0\,..\,21k^3]$。由于 $\mathcal{A}$是最优的，因此它必须与 $P[d\,..\,|P|-e$的两个副本完全匹配。因此，$P[21k^3\,..\,|P|-21k^3)$ 可以从 P在 X和 Y中的同步出现中移除，而不影响代价 $\mathsf{ed}^w_{\le k}(X,\ Y)$。因此，如果第一阶段返回的字符串长度至少为 $42k^3$，那么推论 2.14 中的算法会移除该字符串中除开头的 $21k^3$ 个字符和末尾的 $21k^3$ 个字符之外的所有字符（例如，见图 2）。

线性时间核 为了应用编辑距离等价性的概念，我们需要找出 X和 Y中的同步出现。为此，我们检查 $\mathsf{ed}(X,Y) \le k$是否成立。如果不成立，那么对于每个归一化权重函数 w，$\mathsf{ed}^w(X,Y) \ge \mathsf{ed}(X,Y) > k$均成立，因此我们已经知道 $\mathsf{ed}^w_{\le k}(X,Y) = \infty$。另一方面，如果 $\mathsf{ed}(X,Y) \le k$，那么我们构造一个最优无权 $\mathcal{A}$ 对齐，将 X 映射到 Y。正如事实 2.7 中的严格证明所示，X中未编辑的字符至多形成 $k+1$ 个片段，这些片段与 $\mathcal{A}$完美匹配。X 中的每个片段与其在 Y中经由 $\mathcal{A}$ 映射得到的像共同构成一个同步出现。因此，我们可以用一个等价字符串的出现来替换这些同步出现，该等价字符串的长度为 $\mathcal{O}(k^3)$。由于我们已将 X 和 Y划分为 $\mathcal{O}(k)$个已编辑字符以及 $\mathcal{O}(k)$个同步出现，由此得到长度为 $\mathcal{O}(k^4)$的字符串 X'和 Y'，满足 $\mathsf{ed}^w_{\le k}(X',Y') = \mathsf{ed}^w_{\le k}(X,Y)$。为了高效构造 $\mathcal{A}$，我们使用 [Mye86, LV88] 的 $\mathcal{O}(n+k^2)$ 无权编辑距离算法。不过，如果 $n \le k^4$，那么一开始就无需担心缩减 X 和 Y 的规模，因此不必构造最优无权对齐；否则，$\mathcal{O}(n+k^2) = \mathcal{O}(n)$，构造规模为 $\mathcal{O}(k^4)$ 的核需要线性时间；有关我们针对加权字符串编辑距离所构造的核的详细信息，请参见定理 2.15。

如前所述，一旦我们得到一个大小为 $\mathcal{O}(k^4)$ 的核 (X',Y')，就可以运行 $\mathcal{O}(nk)$ 时间的加权编辑距离算法，在 $\mathcal{O}(k^5)$ 时间内计算 $\mathsf{ed}^w_{\le k}(X',Y') = \mathsf{ed}^w_{\le k}(X,Y)$。

1.3.2 加权树编辑距离

我们的加权树编辑距离算法采用了相同的总体思路。然而，与字符串编辑距离相比，它面临两项主要挑战。首先，周期性的结构丰富得多，需要用到两个概念：森林的水平周期性和上下文的垂直周期性。因此，我们需要分别定义森林和上下文的树编辑距离等价关系。尽管如此，假设权重函数 w 满足三角不等式，我们仍然可以构造大小分别为 $\mathcal{O}(k^3)$ 和 $\mathcal{O}(k^4)$ 的等价森林和上下文。第二项挑战是，用于计算无权树编辑距离的最先进算法具有随机性，其运行时间为 $\mathcal{O}(n\log\ n + \mathsf{poly}(k))$，而非 $\mathcal{O}(n+\mathsf{poly}(k))$。因此，为了实现确定性线性时间核，我们需要另寻方法来识别大型同步出现。我们的替代方案是通过多次迭代（基本上每次都将规模减半）来缩减输入，而不是一次性完成。这样，我们仍然可以得到大小为 $\mathcal{O}(k^5)$ 的核，其渐近规模与通过最优无权对齐所能得到的核一样小。

树中的周期性直观地说，树中的两种周期性对应于将字符串解释为树的两种方式。对于字符串 X，水平嵌入会构造一棵树，其中$|X|$ 个叶节点附着在根节点上，并依次以 X中的字符作为标签；而垂直嵌入会构造一条路径，其中 $|X|$ 个节点依次以 X中的字符作为标签。类似地，形式语言理论中的森林代数（综述见 [BW08] ）涉及两个自然的幺半群：森林的水平幺半群（运算为串接，记作 $\cdot$）和上下文的垂直幺半群（运算为复合，记作 $\star$）。上下文可以定义为一棵在某个叶节点处有一个孔的树；将一个上下文放入另一个上下文的孔中，即可将二者复合。此外，将森林放入上下文的孔中会得到一个森林。为了形式化这些概念，并轻松移植

为字符串设计的组合学和算法工具，我们将森林解释为平衡括号字符串；参见第 3.1 节。

根据 [DGH+22]，水平幂是同一森林的多个副本的串接，而垂直幂是同一上下文的多个副本的组合；示例见图 3。更具体地说，如果一个森林包含形式为 Q^{4k+1} 的子森林，其中森林 Q 的大小为 $|Q| \leq 4k$，那么我们称该森林具有水平 k-周期性；如果一个上下文可以表示为若干个上下文的组合，其中包括 Q^{6k+1}，且上下文 Q 的大小为 $|Q| \leq 8k$，那么我们称该上下文具有垂直k-周期性。

树编辑距离等价森林 我们的加权树编辑距离算法的第一个组成部分，是一个线性时间过程：给定森林 P，构造一个大小为 $\mathcal{O}(k^3)$ 的等价森林。该子程序的第一阶段会消除水平 k-周期性：只要处理中的森林包含形式为 Q^{4k+1} 且 $|Q| \in [1\,..\,4k]$ 的子森林，就用 Q^{4k} 替换该子森林。如引理 3.5 所示，森林 Q^{4k+1} 与 Q^{4k} 等价，因此这一步保留了它与输入森林 P的等价关系。该阶段的高效实现依赖于这样一个事实：若将 P 解释为字符串，那么水平 k-周期性可以解释为形如 Q^{4k+1} 的子串，其中 平衡字符串 Q足够短。因此，我们可以重用引理 2.13 来获得一个与 P 等价且不含水平 k-周期性的森林。

在引理3.7中，我们证明了任意两个大小至少为 $74k^3$ 且不含水平 k周期性的森林之间的等价性。[2] 基于这一结果，如果水平周期性约简得到的森林大小至少为 $74k^3$，我们就返回一个大小恰为 $74k^3$的规范森林；详情见引理 3.17 。

树编辑距离等价上下文 我们的下一个关键环节是一个线性时间算法：给定一个上下文 P，构造一个大小为 $\mathcal{O}(k^4$的等价上下文。首先，我们对 P 中的每个极大森林（不包含孔）应用前述过程。然后，消除垂直 k周期性：只要 P中包含一个形如 Q^{6k+1} 且满足 $|Q| \in [1\,..\,8k]$的上下文，就将其替换为 Q^{6k}。如引理 3.10所示，上下文 Q^{6k+1} 与 Q^{6k} 等价，因此这一步保持了与输入上下文 P的等价性。为了高效实现，将主干路径（即从 P 的根节点到孔的路径）解释为一个字符串，其中每个字符编码对应节点的标签，以及连接在该处、位于主干路径左侧和右侧的子树。这样，垂直k 周期性就可以解释为所构造字符串中的周期性，因此可以再次使用引理2.13 。

在引理 3.12中，我们证明了大小至少为 $578k^4$ 且避免垂直 k-周期性的任意两个上下文与大小超过 $74k^3$的子森林之间的等价关系。因此，如果垂直周期性归约得到的上下文大小至少为 $578k^4$，我们会将其替换为大小恰好为 $578k^4$的规范上下文；详见引理3.18 。

线性时间核 与字符串的情形类似，为了应用树编辑距离等价的概念，我们需要在输入森林 F 和 G 中识别森林与上下文的同步出现。 如上所述，为了获得确定性线性时间核，我们不能使用 [DGH+22] 的算法来获得至多包含 k 次编辑的 F 到 G 的树对齐映射。 相反，我们开发了一种迭代式替代方案。 在每一步中，我们将 F 分解为 $\mathcal{O}$ 个大小至多为 $\frac{n}{2k}$ 的上下文和森林（统称为片段）；详情见引理 3.15。 接下来，我们最大化分解中可在 G 中拥有互不相交同步出现的片段数量；引理 3.16 使用动态规划，在 $\mathcal{O}(n + k^4)$ 时间内实现这一步。 如果 $\mathsf{ted}(F, G) \leq k$,

[2]这一表述比字符串对应的引理 2.11更强，因为现在我们假设权重函数 w 满足三角不等式。

那么未匹配的片段不超过 k 个（最优对齐至多编辑 k 个片段）。我们用大小为 $\mathcal{O}(k^5)$ 的等价片段替换匹配的片段，从而得到大小至多为 $\frac{n}{2}+\mathcal{O}(k^5)$ 的森林，其中第一项对应未匹配片段；见定理 3.19。只要 $n=\omega(k^5)$，这一过程实质上就会将输入大小减半。因此，如推论 3.20 所示，这仍然能得到一个线性时间算法，生成大小为 $\mathcal{O}(k^5)$ 的森林 F' 和 G'，使得 $\mathsf{ted}^w_{\le k}(F',G') = \mathsf{ted}^w_{\le k}(F,G)$。

得到这样一个大小为 $\mathcal{O}(k^5)$ 的核 (F',G') 后，我们可以运行三次时间复杂度的加权编辑距离算法 [DMRW10]，在 $\mathcal{O}(k^{15})$ 时间内计算 $\mathsf{ted}^w_{\le k}(X',Y') = \mathsf{ted}^w_{\le k}(X,Y)$，总运行时间为 $\mathcal{O}(n+k^{15})$。此外，我们通过采用 [AJ21], 的 $\mathcal{O}(nk^2\log n)$ 时间算法，大幅改进了无权树编辑距离问题的当前最佳结果；该算法使无权树编辑距离的总运行时间达到 $\mathcal{O}(n+k^7\log k)$。

1.3.3 加权 Dyck 编辑距离

在本文的最后一节中，加权 Dyck 编辑距离算法采用了与字符串和树编辑距离算法类似的方法。不过，许多证明和细节都专门针对 Dyck 编辑距离问题，并有其自身的一系列复杂之处和难点，我们将在下文中加以概述。

给定字母表 $\Sigma = T\cup\overline{T}$ 上的字符串 X（其中 T 和 $\overline{T}$ 分别是左括号和右括号的集合）、整数 $k\in\mathbb{Z}_+$，以及表示每种编辑操作（括号插入、删除和替换）代价的偏斜度量权重函数 w，我们的目标是计算将 X 转换为 Σ 上括号匹配表达式所需的编辑序列的最小权重，前提是所有编辑的总权重不超过 k。本文设计了一种确定性算法，可在 $\mathcal{O}(n+k^{12})$ 时间内实现这一目标。对于这个问题的无权版本，近期的解决方案 [FGK+22b, Dür22] 能够在 $\mathcal{O}(n+k^{4.5442})$ 时间内计算 Dyck 编辑距离。与此前的算法 [BO16, FGK+22a], 一样，该算法首先执行贪心预处理步骤，穷尽删除任意两个相邻字符 $X[i]X[i+1]$，使得 $X[i]$ 是左括号、$X[i+1]$ 是同一类型的右括号。通过一个简单的论证可以证明，预处理后的字符串的 Dyck 编辑距离与输入字符串 X 的 Dyck 编辑距离完全相同。

预处理 我们尝试对加权版本采用类似的方法，但事实证明，这种简单的分析不足以构造约简字符串。例如，设输入字符串为 ({()。对于一般的权重函数 w，并不能明显看出最优匹配应当始终匹配最后两个括号。事实上，若考虑这样一种权重函数：将 { 替换为) 的代价为 10，而将 (替换为 } 的代价为 5，那么任何最优匹配都应匹配第一个和最后一个括号，而不是最后两个。因此，在本文中，我们假设权重函数 w 是偏斜度量。形式上，我们假设 w 满足三角不等式和反对称性，即对于所有 p_1、$p_2\in\Sigma\cup\{\varepsilon\}$，均有 $w(p_1,p_2) = w(\overline{p_2},\overline{p_1})$，其中 $\overline{p}$ 是与 p（以及 $\overline{\varepsilon}=\varepsilon$）互补的括号。利用 w的这一性质，我们证明可以采用类似的贪心预处理（如无权版本中所述），将X 约简为字符串 X'，同时保持加权 Dyck 编辑距离不变。不过，我们的论证复杂得多，需要根据其他替代对齐的结构逐情况分析（命题 4.7）。尽管如此，显然可以在线性时间内完成贪心预处理。

戴克编辑距离等价字符串 接下来，按照字符串编辑距离中所述的类似策略，我们进一步缩减 X'，生成一个长度为 $\mathcal{O}(k^4)$ 的字符串 X''，同时保持

加权 Dyck 编辑距离。首先，我们引入 k-同步性的概念。仅包含左括号的子串 P 与仅包含右括号的子串 $\overline{P}$ 是 k-同步的，当且仅当 $\overline{P}$ 出现在 P 之后、二者长度相同，且它们的高度差至多为 $2k$。基于这一点以及 Dyck 匹配的非交叉性质，我们首先论证：如果 P、$\overline{P}$ 的长度足够大且距离被 k 所界定，那么在最优对齐中存在一个与子串 $\ell' \in \overline{P}$ 匹配的子串 $\ell \in P$（为便于分析，我们固定其中一个最优对齐）。现在，如果我们用 $P \setminus \ell$ 替换 P，并用 $P' \setminus \ell'$ 替换 P'，那么在所得字符串中，距离保持不变（事实 4.12）。据此，对于任意两个 k-同步的子串 P、$\overline{P}$，我们可以按如下方式降低它们的周期性：如果 $P = Q^e$ 且 $\overline{P} = \overline{Q^e}$（其中 Q 是一个指数 e 较大的本原串），那么 Q 至少有一次出现会在 $\overline{P}$ 中与其反向互补对应物 $\overline{Q}$ 匹配。因此，我们可以移除匹配部分而不改变距离，并将指数降低 1。重复此过程，直到 e 变小（引理 4.13）。

接下来，假设 P、$\overline{P}$ 避免周期性，可以证明存在一对索引 i、$j \in [0\,..\,78k^3]$，使得在最优对齐中，$P[i]$ 与 $\overline{P}[|P|-1-i]$ 匹配，且 $P[|P|-1-j]$ 与 $\overline{P}[j]$ 匹配。因此，根据 $|P| = |\overline{P}|$ 这一事实以及 Dyck 最优对齐的非交叉性质，i 和 $|P|-1-j$ 之间的所有索引也都匹配；于是，从 P、$\overline{P}$ 中移除这些匹配的字符，不会影响 Dyck 编辑距离。由此，我们将每对 k-同步配对替换为长度仅为 $156k^3$ 的子串（用 P、$\overline{P}$ 的首尾各 $78k^3$ 个字符替换），生成字符串 X''，使得 X 和 X'' 的加权 Dyck 编辑距离保持不变（引理 4.15，推论 4.17）。

线性时间核 最后，我们证明，若距离不超过 k，则可在时间 $\mathcal{O}(n + k^5)$ 内将 X 划分为 $\mathcal{O}(k)$ 对互不相交的 k-同步子串（另加 $\mathcal{O}(k)$ 个单独字符），因此 X'' 的总长度不超过 $\mathcal{O}(k^4)$。首先，对输入字符串 X 进行预处理，生成 X'。接下来，我们检查 ded(X') 是否满足 $\leq k$；若满足，则在时间 $\mathcal{O}(n + k^5)$ 内计算 X' 的无权最优 Dyck 对齐 $\mathcal{M}$，[FGK+22a]。然后，我们论证，由 $\mathcal{M}$ 匹配的 X' 的任意一对子串都是 k-同步的。因此，利用 $\mathcal{M}$，我们找出由 $\mathcal{M}$匹配的 T^* 和 $\overline{T}^*$ 中的极大子串集合。子串之所以称为极大，是因为若要向左或向右扩展该子串本身或其匹配项，就必须付出编辑代价。由于无权 Dyck 编辑距离不大于加权版本，且假设 $\mathcal{M}$ 的代价不超过 k，我们可以证明字符串 X' 可划分为$\mathcal{O}(k)$ 对不同的 k-同步子串。此外，通过对 X'进行从左到右扫描，可以在线性时间内找到这些极大片段。随后，我们按如下方式从 X' 构造字符串 X''：(i) 对于每一对 k-同步子串，按照上文所述算法对其进行约简，并添加两个对应的字符串，每个长度为 $O(k^3)$；(ii) 将由 $\mathcal{M}$ 编辑的所有字符以完全相同的方式添加到 X''（定理 4.19）。

最后，我们使用 X'' 的动态规划算法计算 [Mye95]的加权 Dyck 编辑距离，时间复杂度为 $\mathcal{O}(k^{12})$。

2 字符串编辑距离

2.1 预备知识

字符串 $Y \in \Sigma^n$ 是一个由字母表 Σ中的 $|Y| := n$ 个字符组成的序列。对于 $i \in [0\,..\,n)$，我们用 $Y[i]$表示 Y 的第 i个字符。若存在某些整数 $0 \leq i \leq j \leq |Y|$使得 $X = Y[i] \cdots Y[j-1]$ 成立，我们就说字符串 X 在字符串Y中作为子串出现。我们将 X 的底层出现记作 $Y[i\,..\,j)$。形式上，$Y[i\,..\,j)$是 Y的一个片段，可以通过引用 Y及其端点 i、j来表示。片段 $Y[i\,..\,j)$也可以记作

$Y[i..j-1]$、$Y(i-1..j-1]$或 $Y(i-1..j)$。形式为 $Y[0..j)$的片段是 Y的前缀，而形式为 $Y[i..n)$的片段是 Y的后缀。

定理 2.1（最长公共扩展查询 [LV88、FFM00]）。字符串 X、Y 可以在线性时间内进行预处理，从而可以在 最长公共扩展（LCE）查询中回答以下问题：给定位置 $\mathcal{O}$（1）时间内，给定位置 $x \in [0..|X|]$ 和 $y \in [0..|Y|]$，计算最大的 ℓ 使得 $X[x..x+\ell) = Y[y..y+\ell)$。

如第 1.3 节所述，高次幂周期性在我们的算法中发挥着关键作用，现在我们可以在此正式定义字符串的高次幂周期性。若对于所有 $i \in [0..n-p)$ 都成立，则整数 $p \in [1..n]$ 是字符串 $Y \in \Sigma^n$ 的周期。在这种情况下，前缀 $Y[0..p)$ 被称为 Y 的字符串周期。我们用 $\mathsf{per}(Y)$ 表示 Y 的最小周期。字符串 Y 的指数定义为 $\mathsf{exp}(Y) := \frac{|Y|}{\mathsf{per}(Y)}$，若字符串 Y 满足 $\mathsf{exp}(Y) \geq 2$，则称其具有周期性。

定理 2.2（2-周期查询 [KRRW15，BII+17]）。字符串 X 可以在线性时间内进行预处理，以便在常数时间内判断任意给定片段 $X[i..j)$ 是否具有周期性；若是，则计算其最短周期 $\mathsf{per}(X[i..j))$。

对于字符串 Y和整数 $m \geq 0$，我们将 Y的 m次幂记作 Y^m，定义为 m 个 Y串接而成。非空字符串 $Y \in \Sigma^n$ 若不能表示为某个字符串 X 的 $m > 1$次幂 $Y = X^m$，则称其为本原字符串。对于字符串 $Y \in \Sigma^n$，我们定义正向旋转 $\mathsf{rot}(Y) = Y[1]\cdots Y[n-1]Y[0]$。一般而言，移位量为 $s \in \mathbb{Z}$的循环旋转 $\mathsf{rot}^s(Y)$ 可通过重复执行 rot 或其逆操作 rot^{-1}得到。字符串 Y是本原的，当且仅当它不同于其所有非平凡旋转；也就是说，只有当 s 是 n的倍数时，$Y = \mathsf{rot}^s(Y)$ 才成立。

2.2 编辑距离对齐与加权编辑距离

在本小节中，我们讨论对齐及其加权代价，这为描述将字符串 X 转换为 Y所需的编辑序列提供了一种形式化方法。

定义 2.3。序列 $\mathcal{A} = (x_t, y_t)_{t=0}^m$是将片段 $X[x..x')$对齐到片段 $Y[y..y')$上的一个对齐，如果 $(x_0, y_0) = (x, y)$、$(x_m, y_m) = (x', y')$，且 $(x_{t+1}, y_{t+1}) \in \{(x_t+1, y_t+1)、(x_t+1, y_t)、(x_t, y_t+1)\}$ 对于 $t \in [0..m)$成立。将 $X[x..x')$对齐到 $Y[y..y')$上的所有对齐的集合记作 $\mathsf{A}(X[x..x')\ Y[y..y'))$。

给定一个对齐 ment $\mathcal{A} = (x_t, y_t)_{t=0}^m \in \mathsf{A}(X[x..x')\ Y[y..y'))$, 对于每个 $t \in [0..m)$，我们称 t

- $\mathcal{A}$ 若 $(x_{t+1}, y_{t+1}) = (x_t+1, y_t)$，则删除 $X[x_t]$。
- $\mathcal{A}$ 若 $(x_{t+1}, y_{t+1}) = (x_t, y_t+1)$，则插入 $Y[y_t]$。
- $\mathcal{A}$ 若 $(x_{t+1}, y_{t+1}) = (x_t+1, y_t+1)$，则将 $X[x_t]$ 对齐到 $Y[y_t]$，记作 $X[x_t] \sim_{\mathcal{A}} Y[y_t]$。
- $\mathcal{A}$ 若 $X[x_t] \sim_{\mathcal{A}} Y[y_t]$ 且 $X[x_t] = Y[y_t]$，则 $X[x_t]$ 与 $Y[y_t]$匹配，记作 $X[x_t] \simeq_{\mathcal{A}} Y[y_t]$。
- $\mathcal{A}$ 将 $X[x_t]$ 替换为 $Y[y_t]$，如果 $X[x_t] \sim_{\mathcal{A}} Y[y_t]$，但 $X[x_t] \neq Y[y_t]$。

插入、删除和替换统称为（字符）编辑。

例 2.4。举一个对齐的例子，考虑字符串 X =abc 和 Y =bd。一种最优对齐 $\mathcal{A}$可能是 {(0, 0), (1, 0), (2, 1), (3, 2)}。配对 (0, 0)、(1, 0) 表示删除了 $\mathcal{A}$ 的 $X[0]$ =a。配对 (1, 0)、(2, 1)、(3, 2) 表示 $\mathcal{A}$ 与 $X[1..2] \sim_{\mathcal{A}} Y[0..1]$对齐，即 bc $\sim_{\mathcal{A}}$ bd。此外，$X[1]$ 与 $Y[0]$ 匹配，因为 $X[1] = Y[0] =$ b，而 $X[2]$ 被替换为 $Y[1]$，因为 $X[2] =$ c $\neq$ d $= Y[1]$。

对于字母表 Σ，我们定义 $\Sigma := \Sigma \cup \{\varepsilon\}$，其中 ε 是 Σ上的空串。我们称函数 $w\colon \Sigma \times \Sigma \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ 为权重函数，若对所有 $a \in \Sigma$，均满足 $w(a, a) = 0$。相对于权重函数 w，对齐 $\mathcal{A} \in \mathsf{A}(X[x..x')\ Y[y..y'))$ 的代价（记为 $\mathsf{ed}_{\mathcal{A}}^w(X[x..x')\ Y[y..y'))$）定义为 $\mathcal{A}$执行的编辑的总代价，其中：

- 删除 $X[x]$ 的代价为 $w(X[x], \varepsilon)$,
- 插入 $Y[y]$ 的代价为 $w(\varepsilon, Y[y])$,
- 将 $X[x]$ 替换为 $Y[y]$ 的代价为 $w(X[x], Y[y])$.

.....的宽度　对齐 $(x_t, y_t)_{t=0}^{m} \in \mathsf{A}(X[x\,..\,x')\ Y[y\,..\,y'))$ 的定义为 $\max_{t=0}^{m} |x_t - y_t|$。

我们通常考虑将整个字符串 $X[0\,..\,|X|)$ 对齐到整个字符串 $Y[0\,..\,|Y|)$，并将所有此类对齐构成的集合记为 $\mathsf{A}(X,Y) = \mathsf{A}(X[0\,..\,|X|)\ Y[0\,..\,|Y|))$。相对于权重函数 w，字符串 $X, Y \in \Sigma^*$ 的加权编辑距离定义为 $\mathsf{ed}^w(X,Y) = \min_{\mathcal{A} \in \mathsf{A}(X,Y)} \mathsf{ed}^w_{\mathcal{A}}(X,Y)$。对于 $k \in \mathbb{R}_{\geq 0}$，我们也记为

$$\mathsf{ed}^w_{\leq k}(X,Y) = \begin{cases} \mathsf{ed}^w(X,Y) & \text{if } \mathsf{ed}^w(X,Y) \leq k, \\ \infty & \text{otherwise.} \end{cases}$$

在文献中，X 和 Y 的（加权）编辑距离有时定义为将 X 变换为 Y的一系列编辑的最低代价。如下述事实所示（其技术性证明推迟至附录 A），只要 w 是拟度量，也就是说，对任意 a、b、$c \in \Sigma$，它满足三角不等式 $w(a,b) + w(b,c) \geq w(a,c)$，这种基于序列的观点就等价于我们基于对齐的观点。在基于序列的观点中，假设 w 是拟度量并不失一般性（单个字符可以被编辑多次，因此可以用距离闭包替换w，而不影响编辑距离）。另一方面，我们基于对齐的观点更为一般，能够涵盖违反三角不等式的加权编辑距离。

事实 2.5. 如果 w 是 Σ上的拟度量，那么 ed^w 是 Σ^*上的拟度量。在这种情况下，$\mathsf{ed}^w(X,Y)$ 也可以等价地定义为将 X 变换为 Y的一系列编辑的最低代价。

虽然我们针对字符串的算法适用于任意权重函数，但其树和 Dyck 对应版本假设 w 是拟度量。具体而言，它们依赖于附录中证明的以下事实A。

事实 2.6. 考虑字符串 X及其片段 $X[i\,..\,j)$。那么，对于每个拟度量 w，我们都有 $\mathsf{ed}^w(X, X[i\,..\,j)) = \mathsf{ed}^w(X[0\,..\,i) \cdot X[j\,..\,|X|)\ \varepsilon)$.

虽然我们的主要结果针对加权编辑距离，但我们的算法也依赖于无权编辑距离过程。如果 w 是 Σ 上的离散度量（也就是说，对于每个$a, b \in \Sigma$，都有 $w(a,b) = 0$ 当且仅当 $a = b$，而 $w(a,b) = 1$ 则在其他情况下成立），那么我们会在 ed^w 和 $\mathsf{ed}^w_{\mathcal{A}}$ 中省略上标 w。这样便得到单位代价编辑距离（也称为无权编辑距离或莱文斯坦距离）。我们称权重函数 w 为规范化的，即对于所有满足 $a \neq b$ 的 $a, b \in \Sigma$，$w(a,b) \geq 1$ 都成立。在这种情况下，对于所有字符串 $X, Y \in \Sigma^*$ 以及对齐 $\mathcal{A} \in \mathsf{A}(X,Y)$，$\mathsf{ed}^w_{\mathcal{A}}(X,Y) \geq \mathsf{ed}_{\mathcal{A}}(X,Y)$ 均成立。

给定一个对齐 $\mathcal{A} = (x_t, y_t)_{t=0}^{m} \in \mathsf{A}(X,Y)$，对于每个 $\ell, r \in [0\,..\,m]$，满足 $\ell \leq r$，我们称$\mathcal{A}$ 将 $X[x_\ell\,..\,x_r)$ 对齐到 $Y[y_\ell\,..\,y_r)$，记作 $X[x_\ell\,..\,x_r) \sim_{\mathcal{A}} Y[y_\ell\,..\,y_r)$。此时，对于任意权重函数 w，我们用 $\mathsf{ed}^w_{\mathcal{A}}(X[x_\ell\,..\,x_r)\ Y[y_\ell\,..\,y_r))$ 表示将 $X[x_\ell\,..\,x_r)$ 对齐到 $Y[y_\ell\,..\,y_r)$ 所诱导的对齐的代价。如果 $\mathsf{ed}^w_{\mathcal{A}}(X[x_\ell\,..\,x_r)\ Y[y_\ell\,..\,y_r)) = 0$，我们称 $\mathcal{A}$ 以 $Y[y_\ell\,..\,y_r)$ 匹配 $X[x_\ell\,..\,x_r)$，记作 $X[x_\ell\,..\,x_r) \simeq_{\mathcal{A}} Y[y_\ell\,..\,y_r)$。

事实 2.7. 考虑 $k \in \mathbb{Z}_{\geq 0}$、字符串 X, Y，以及一个代价为 $\mathsf{ed}_{\mathcal{A}}(X,Y) \leq k$ 的对齐 $\mathcal{A} \in \mathsf{A}(X,Y)$。那么，字符串 X 可以划分为至多 k 个单个字符（$\mathcal{A}$ 会删除或替换这些字符），以及至多 $k+1$ 个片段；这些片段与 Y 的片段完美匹配。

证明。令 $\mathcal{A} = (x_t, y_t)_{t=0}^{m}$，并令 $t_1 < \cdots < t_e$ 为 $[0\,..\,m)$ 中与 $\mathcal{A}$ 中的编辑相对应的索引。那么，$\mathcal{A}$ 完美匹配的极大片段为 $X[0\,..\,x_{t_1})$、$X[x_{t_i+1}\,..\,x_{t_{i+1}})$（对于 $i \in [1\,..\,e)$），以及 $X[x_{t_e+1}\,..\,|X|)$。此外，对于每个满足 $x_{t_i+1} > x_{t_i}$ 的 $i \in [1\,..\,e]$，$\mathcal{A}$ 会删除或替换 $X[x_{t_i}]$。每次编辑都会为 $\mathcal{A}$ 的代价贡献一个单位，因此该分解至多包含 $e \leq k$ 个经过编辑的字符和 $e+1 \leq k+1$ 个完美匹配的片段。 □

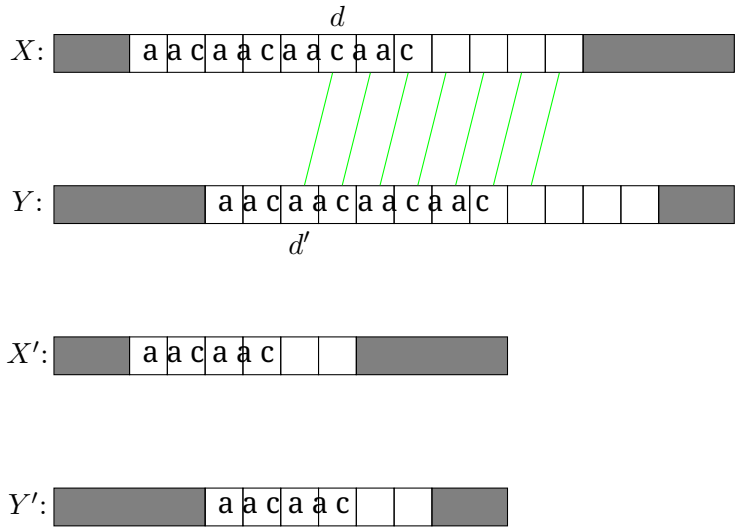

图1：对 X 和 Y 进行周期性缩减，其中最优对齐由连接两个字符串中字符的线表示。在 X 和 Y 中索引为 d、d' 的位置处，周期子串完全对齐（由绿色线条表示），因此，我们可以缩减这些周期子串的幂，构造出 X' 和 Y'，使得 $\mathsf{ed}(X,Y) = \mathsf{ed}(X',Y')$。

2.3 组合基础

在介绍加权字符串编辑距离的算法之前，我们先讨论编辑距离等价子串，这是我们的主要技术贡献之一。

定义 2.8. 对于 $k \in \mathbb{Z}_{\geq 0}$ 和权重函数 w，如果字符串 P、P' 满足以下条件，则称它们是编辑距离 $\overset{w}{\leq k}$-等价的

$$\mathsf{ed}^{w}_{\leq k}(X,Y) = \mathsf{ed}^{w}_{\leq k}(X[0\,..\,p_X)\cdot P'\cdot X[p_X+|P|\,..\,|X|), Y[0\,..\,p_Y)\cdot P'\cdot Y[p_Y+|P|\,..\,|Y|))$$

对于所有字符串 X 和 Y，若 P 分别在位置 p_X 和 p_Y 处出现，且满足 $|p_X - p_Y| \leq k$，则上述条件成立。我们称 P 在 X 和 Y 中的这类出现为 k-同步出现。

首先，我们证明，改变周期子串的幂次不会改变该子串的任何同步出现的编辑距离代价。其次，我们证明，计算编辑距离时只需考虑子串较短的前缀和后缀。在这两种情况下，我们都能证明，在编辑距离实例中，任何最优对齐都必须对齐具有同步出现的较长子串（先是周期性的，再是非周期性的），因此计算编辑距离时无需考虑其中大多数较长子串。相关示例见图 1和图 2，展示了这两种编辑距离等价变换步骤。

引理 2.9. 设 $k \in \mathbb{Z}_+$，Q 是一个字符串，且 e、$e' \in \mathbb{Z}_{\geq 4k}$。那么，对于任意权重函数 w，Q^e 和 $Q^{e'}$ 都是 $\mathsf{ed}^{w}_{\leq k}$- 等价的。

证明. 不失一般性，我们假设 Q 是本原的。（如果 $Q = R^m$ 对于 $m \in \mathbb{Z}_{\geq 2}$，那么 $Q^e = R^{me}$ 和 $Q^{e'} = R^{me'}$ 可以解释为 R 的幂，而不是 Q 的幂。）假设 Q^e 分别在字符串 X 和 Y 中的位置 p_X 和 p_Y 处出现，且满足 $|p_X - p_Y| \leq k$。记 $X' = X[0\,..\,p_X)\cdot Q^{e'}\cdot X[p_X+|Q^e|\,..\,|X|)$ 和 $Y' = Y[0\,..\,p_Y)\cdot Q^{e'}\cdot Y[p_Y+|Q^e|\,..\,|Y|)$。此外，令 $q = |Q|$，并令 $\mathcal{A} \in \mathsf{A}(X,Y)$ 为一个对齐，使得 $\mathsf{ed}^{w}(X,Y) = \mathsf{ed}^{w}_{\mathcal{A}}(X,Y) \leq k$。

断言 2.10。存在 $i_X, i_Y \in [0\,..\,3k]$ 使得

$$X[p_X + i_X\cdot q\,..\,p_X + (i_X+1)\cdot q) \simeq_{\mathcal{A}} Y[p_Y + i_Y\cdot q\,..\,p_Y + (i_Y+1)\cdot q).$$

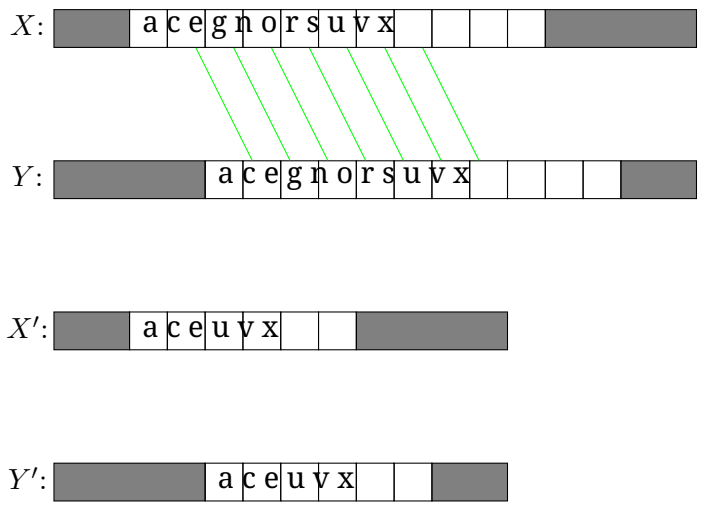

图 2：对于子串 P 的任意避免 k-周期性的同步出现，任何最优对齐（用连接两个字符串中字符的线表示）都必须匹配 P 的大部分内部字符（见绿色线条）。我们可以通过移除这些匹配的字符来构造字符串 X'、Y'，使得 $\mathsf{ed}(X,Y) = \mathsf{ed}(X',Y')$。

证明。令 $(t_X, t_Y) \in \mathcal{A}$ 为满足 $t_X \geq p_X$ 和 $t_Y \geq p_Y$ 的 $\mathcal{A}$ 中最左的元素。根据 X 和 Y 之间的对称性，不失一般性地假设 $t_X = p_X$。考虑从位置 $p_X + i \cdot q$ 开始的 X 中 Q 的 $k+1$ 次出现，其中 $i \in [0\,..\,k]$。对齐 $\mathcal{A}$ 至少与其中一个完全匹配；因此，我们可以定义 $i_X \in [0\,..\,k]$，使得 $\mathcal{A}$ 与某个片段 $Y[s_Y\,..\,s_Y + q$ 完全匹配，即 $X[p_X + i_X \cdot q\,..\,p_X + (i_X + 1) \cdot q)$。由于 $(t_X, t_Y) \in \mathcal{A}$，$\mathcal{A}$ 的非交叉性质意味着 $s_Y \geq t_Y \geq p_Y$。此外，由于 $\mathsf{ed}_{\mathcal{A}}(X,Y) \leq k$ 且 $X[p_X + i_X \cdot q] \simeq_{\mathcal{A}} Y[s_Y]$，我们有 $s_Y \leq (p_X + i_X \cdot q) + k \leq p_X + kq + k \leq p_Y + kq + 2k \leq p_Y + 3kq$。此外，由于 Q 是本原的（即不同于其所有非平凡循环旋转），我们得出，对于某个 $i_Y \in [0\,..\,3k]$，有 $s_Y = p_Y + i_Y \cdot q$。□

现在，如果将 $Q^e = X[p_X\,..\,p_X + e \cdot q) = Y[p_Y\,..\,p_Y + e \cdot q)$ 替换为 $Q^{e'}$（对于 $e' \geq e - 1$），我们可以将其解释为：用 $Q^{1+e'-e}$ 替换 $Q = X[p_X + i_X \cdot q\,..\,p_X + (i_X + 1) \cdot q) = Y[p_Y + i_Y \cdot q\,..\,p_Y + (i_Y + 1) \cdot q)$。根据断言 2.10，可以在不修改其代价的情况下对 $\mathcal{A}$ 进行简单调整，因此 $\mathsf{ed}^w(X',Y') \leq \mathsf{ed}^w_{\mathcal{A}}(X,Y) = \mathsf{ed}^w(X,Y)$。如果 $e' < e - 1$，我们就重复上述论证，一次将指数 e 减少一步，同时仍可得出 $\mathsf{ed}^w(X',Y') \leq \mathsf{ed}^w(X,Y)$。无论哪种情况，通过 (X,Y,e) 与 (X',Y',e') 之间的对称性，都可得到反向不等式。□

如果字符串不包含任何形如 $Q^{4k+1}$with $|Q| \in [1\,..\,2k]$ 的子串，我们就称该字符串避免 k-周期性。

引理 2.11. 设 $k \in \mathbb{Z}_+$，并设 P、P' 是长度至少为 $42k^3$ 的字符串，满足 $P[0\,..\,21k^3) = P'[0\,..\,21k^3)$ 和 $P[|P| - 21k^3\,..\,|P|) = P'[|P'| - 21k^3\,..\,|P'|)$ 均避免 k-周期性。那么，对于任意权重函数 w，P 和 P' 都是 $\mathsf{ed}^w_{\leq k}$- 等价的。

证明。假设 P 分别在字符串 X 和 Y 中出现在位置 p_X 和 p_Y，且满足 $|p_X - p_Y| \leq k$。记 $X' = X[0\,..\,p_X) \cdot P' \cdot X[p_X + |P|\,..\,|X|)$ 和 $Y' = Y[0\,..\,p_Y) \cdot P' \cdot Y[p_Y + |P|\,..\,|Y|)$。此外，设 $\mathcal{A} \in \mathsf{A}(X,Y)$ 是一个对齐，使得 $\mathsf{ed}^w(X,Y) = \mathsf{ed}^w_{\mathcal{A}}(X,Y) \leq k$。

断言 2.12. 存在 $d, e \in [0\,..\,21k^3]$ 使得

$$X[p_X + d\,..\,p_X + |P| - e) \sim_{\mathcal{A}} Y[p_Y + d\,..\,p_Y + |P| - e).$$

证明。将 $X[p_X..p_X+21k^3)$ 划分为表示删除或替换 $\mathcal{A}$ 的单个字符，以及与 Y 的片段完美匹配的 $\mathcal{A}$ 极大片段。根据事实 2.7，这种极大片段的数量至多为 $k+1$，且其总长度至少为 $21k^3-k\geq 20k^3$。因此，其中一个片段的长度至少为 $\frac{20k^3}{k+1}\geq 10k^2$。于是，设 $R:=X[r_X..r_X+|R|)$ 是包含在 $X[p_X..p_X+21k^3)$ 中、长度至少为 $10k^2$，且与 $Y[r_Y..r_Y+|R|)$ 完美匹配的片段 $\mathcal{A}$。此外，设 $r'_Y:=r_X+p_Y-p_X$。若 $r_Y=r'_Y$，则令 $d:=r_X-p_X=r_Y-p_Y$，使得 $(p_X+d,p_Y+d)\in\mathcal{A}$。否则，$Y[r_Y..r_Y+|R|)$ 和 $Y[r'_Y..r'_Y+|R|)$ 都是 R 在 Y 中的出现。此外，$0<|r_Y-r'_Y|\leq|r_Y-r_X|+|r'_Y-r_X|\leq \mathsf{ed}^w_{\mathcal{A}}(X,Y)+|p_Y-p_X|\leq 2k$。因此，$\mathsf{per}(R)\leq|r_Y-r'_Y|\leq 2k$，且 $\mathsf{exp}(R)\geq\frac{|R|}{2k}\geq 4k+1$。由于 $Y[r'_Y..r'_Y+|R|)$ 包含在 $Y[p_Y..p_Y+21k^3)=P[0..21k^3)$ 中，这与关于 $P[0..21k^3)$ 避免 k-周期性的假设相矛盾。

通过对称的论证可知，对于某个 $e\in[0..21k^3]$，$(p_X+|P|-e,p_Y+|P|-e)$ 成立，因此我们可以得出 $X[p_X+d..p_X+|P|-e)\sim_{\mathcal{A}}Y[p_Y+d..p_Y+|P|-e)$。□

根据断言 2.12，我们有 $X[p_X+d..p_X+|P|-e)\sim_{\mathcal{A}}Y[p_Y+d..p_Y+|P|-e)$。两个片段都匹配 $P[d..|P|-e)$，因此 $\mathcal{A}$ 的最优性保证 $X[p_X+d..p_X+|P|-e)\simeq_{\mathcal{A}}Y[p_Y+d..p_Y+|P|-e)$。因此，若将 $P=X[p_X..p_X+|P|)=Y[p_Y..p_Y+|P|)$ 替换为 P'，我们可以将其解释为 $P[d..|P|-e)=X[p_X+d..p_X+|P|-e)=Y[p_Y+d..p_Y+|P|-e)$，其中包含 $P'[d..|P'|-e)$。由于 $X[p_X+d..p_X+|P|-e)\simeq_{\mathcal{A}}Y[p_Y+d..p_Y+|P|-e)$，对齐 $\mathcal{A}$ 可以直接调整而不改变其代价，因此 $\mathsf{ed}^w(X',Y')\leq\mathsf{ed}^w_{\mathcal{A}}(X,Y)=\mathsf{ed}^w(X,Y)$。反向不等式可由 (X,Y,P) 与 (X',Y',P') 的对称性得出。□

2.4 算法

以下引理使我们能够将任意字符串 P 转换为一个避免 k-周期性、并且对于每个权重函数 w都与 P 具有编辑等价性的字符串 P'。该引理以一般形式陈述，以便在后续章节中重复使用。

引理 2.13。设 $e\in\mathbb{Z}_+$，并令 $\mathcal{Q}$ 为一个长度至多为 e的本原字符串族。存在一种算法，通过反复将输入字符串 P 中的 Q^{e+1}的一次出现（对于某个 $Q\in\mathcal{Q}$）替换为 Q^e的一次出现，最终得到一个不包含任何 Q^{e+1} 出现（对于任意 $Q\in\mathcal{Q}$）的字符串 P'。此外，借助一个常数时间预言机（用于测试给定的 P 的本原片段是否属于 $\mathcal{Q}$），该算法可以在线性时间内实现。

证明。 在预处理阶段，我们为 LCE 和 2-周期查询构建数据结构，所需时间为 P；参见定理 2.1 和 2.2。在主阶段，我们的算法从左到右扫描字符串 P，同时维护一个字符串 R和一个索引 $r\in[0..|P|]$，使得 $R\cdot P[r..|P|)$：

- 由 P通过反复将 Q^{e+1}（对于某个 $Q\in\mathcal{Q}$）的一次出现替换为 Q^e的一次出现而得到，
- 不包含任何从小于 $|R|$的位置开始的 Q^{e+1} 出现（对于 $Q\in\mathcal{Q}$）。

我们用 $R:=\varepsilon$和 $r:=0$初始化该过程。每一步中，我们测试 $P[r..r+2q)$是否具有周期性；如果是，则取其最短周期 q；否则，设置 $q:=1$。然后，我们进一步检查 $P[r..r+q)\in\mathcal{Q}$以及 $P[r..r+eq)=P[r+q..r+q+eq)$。如果两项测试都成功，我们将索引 r移至位置 $r+q$。否则，我们将 $P[r]$追加到 R，并将 r递增。

让我们分析该算法的正确性。首先，假设 $P[r..|P|)$没有形如 Q^{e+1} 的前缀（对任何 $Q\in\mathcal{Q}$都成立）。特别地，$P[r..r+q)\notin\mathcal{Q}$ 或 $m<eq$。因此，我们的算法将 $P[r]$ 追加到 R，并将 r 加 1。由于 $R\cdot P[r..|P|)$未发生变化，且 $P[r..|P|)$没有形如 Q^{e+1} 的前缀（对任何 $Q\in\mathcal{Q}$都成立），不变式仍然得到满足。

算法 1：将 P 中每个出现的 $Q \in \mathcal{Q}$ 的幂的指数限制为至多 e。

```
周期性约简(P, e, 𝒬):
    R ← ε;
    r ← 0;
    当 r < |P| 时
        如果 r + 2e ≤ |P| 和 P[r..r + 2e) 是周期性的，则 q ← per(P[r..r + 2e));
        否则 q ← 1;
        m ← max{ℓ : P[r..r + ℓ) = P[r + q..r + q + ℓ)};
        如果 m ≥ eq 且 P[r..r + q) ∈ 𝒬 则
            r ← r + q;
        else
            R ← R · P[r];
            r ← r + 1;
    返回 R;
```

接下来，假设 $P[r..|P|)$ 对某个 $Q \in \mathcal{Q}$ 具有形如 Q^{e+1} 的前缀。如果 $|Q| \neq 1$，那么 $P[r..r+2e)$ 的最短周期是 $|Q|$，因为 Q 是本原的，且 $|Q| \leq e$。另一方面，如果 $|Q| = 1$，那么 $P[r..r+2e)$ 要么具有周期 1，要么具有至少 $e+2$ 的周期。在所有情况下，该算法都能正确识别 $q = |Q|$。此外，随后进行的测试 $P[r..r+q)$ 是否属于 $\mathcal{Q}$ 以及 $m \geq eq$ 是否成立，结果均为成功。因此，该算法将 $R \cdot P[r..|P|)$ 转换为 $R \cdot P[r+q..|P|)$；这是一个有效操作，因为 $P[r..|P|)$ 的前缀 Q^{e+1} 被替换为 $P[r+q..|P|)$ 的前缀 Q^e。因此，剩下需要证明的是，$R \cdot P[r+q..|P|)$ 不包含任何 $\hat{} Q^{e+1}$ 的出现（对任意 $\hat{} Q \in \mathcal{Q}$ 都成立），且该出现的起始位置小于 $|R|$。由于 $R \cdot P[r..|P|)$ 不包含这样的出现，$\hat{} Q^{e+1}$ 的出现必须在位置 $|R|+m$ 或更大处结束。因此，$\hat{}$ 片段 $P[r+q..r+q+m)$ 具有周期 q 和 $\hat{}q := |Q|$。此外，根据 Q 和 $\hat{}Q$ 的本原性，周期性引理 [FW65] 蕴含 $q = \hat{}q$，这是由于 $m \geq eq \geq e+q-1 \geq \hat{}q+q-1$。然而，这意味着 q 是 $P[r..r+q+m]$ 的一个周期，与 m 的定义矛盾。

总运行时间为线性时间，包括定理 2.1 和 2.2 中数据结构的预处理和查询时间，因为 **while** 循环的每次迭代都只需常数时间。 □

推论 2.14. 存在线性时间算法，对于给定的字符串 P 和整数 $k \in \mathbb{Z}_+$，构造一个长度至多为 $42k^3$ 的字符串，使其对于每个权重函数 w 都与 P $\mathsf{ed}^w_{\leq k}$- 等价。

算法 2：构造一个长度至多为 $42k^3$ 且与 P $\mathsf{ed}^w_{\leq k}$-等价的字符串。

```
字符串规约(P, k):
    P′ ← 周期性约简(P, 4k, {Q ∈ Σ+ : |Q| ≤ 2k 和 Q 为本原});
    如果 |P′| ≥ 42k^3，则返回 P′[0..21k^3) · P′[|P′| − 21k^3..|P′|);
    否则返回 P′;
```

证明。我们设置 $P' := \mathsf{PeriodicityReduction}(P, 4k, \mathcal{Q})$，其中 $\mathcal{Q}$ 由长度属于 $[1..2k]$的所有本原字符串组成。我们返回 $P'' := P'[0..21k^3) \cdot P'[|P'| - 21k^3..|P'|)$，或者 P'，具体取决于 $|P'| \geq 42k^3$ 是否成立。根据引理 2.9 和 2.13，字符串 P' 与 $\overset{w}{\leq k}$在编辑距离意义下等价，并且避免了 k-周期性。因此，若 $|P'| \leq 42k^3$，则该算法正确。否则，引理 2.11 意味着 P'' 与 P' 在编辑距离意义下等价（并且根据传递性，也与 P等价），因为 $P'[0..21k^3)$ 并且

$P'[|P'|-21k^3\,..\,|P'|)$ 避免 k-周期性。由于引理 2.13，运行时间为线性（当且仅当本原片段的长度不超过 $\mathcal{Q}$ 时，该片段才属于其中；对此进行测试需要 $2k$，耗时 $\mathcal{O}(1)$ 时间。 □

定理 2.15. 存在一个线性时间算法，给定字符串 X、Y 和整数 $k \in \mathbb{Z}_+$，构造长度至多为 $85k^4$ 的字符串 X'、Y'，使得对于任意权重函数 w，$\mathsf{ed}^w_{\le k}(X,Y) = \mathsf{ed}^w_{\le k}(X',Y')$ 成立。

算法 3：构造长度至多为 $85k^4$ 的字符串 X'、Y'，使得 $\mathsf{ed}^w_{\le k}(X,Y) = \mathsf{ed}^w_{\le k}(X',Y')$

```
1  StringKernel(X, Y, k):
2      如果 |X| ≤ 85k^4 和 |Y| ≤ 85k^4，则返回 (X, Y);
3      如果对于某个 a ∈ Σ，ed(X, Y) > k，则返回 (a^{k+1}, ε);
4      令 (x_t, y_t)_{t=0}^m ∈ A(X, Y) 是满足 ed_A(X, Y) ≤ k 的一个对齐;
5      X', Y', P ← ε;
6      对于 t ← 0 到 m，执行
7          如果 t < m 且 x_{t+1} > x_t 且 y_{t+1} > y_t 且 X[x_t] = Y[y_t]，则
8              P ← P · X[x_t]
9          else
10             P ← 字符串规约(P, k);
11             X' ← X' · P;
12             Y' ← Y' · P;
13             P ← ε;
14             如果 t < m 且 x_{t+1} > x_t，则 X' ← X' · X[x_t];
15             如果 t < m 且 y_{t+1} > y_t，则 Y' ← Y' · Y[y_t];
16     返回（X', Y'）
```

证明。我们的过程由算法 3 实现。首先，如果 X 和 Y 的长度都不超过 $85k^4$，则我们原样返回 X 和 Y。如果 $\mathsf{ed}(X,Y) > k$，则返回字符串 a^{k+1} 和ε，其中 $a \in \Sigma$ 是任意字符。如果 $\mathsf{ed}(X,Y) \le k$，我们构造一个对齐 $\mathcal{A} := (x_t, y_t)_{t=0}^m \in \mathsf{A}(X,Y)$，其（无权）代价至多为 k。随后，我们在从左到右扫描对齐 $\mathcal{A}$的过程中构造输出字符串 X' 和 Y'：将 X 和Y 中所有 $\mathcal{A}$ 编辑的字符分别追加到 X' 和 Y'。此外，对于 X 和 Y 中每一对完美匹配的极大片段，我们应用推论 2.14 中的规约，并将所得字符串同时追加到 X' 和 Y'。

现在证明所得实例 (X',Y') 满足 $\mathsf{ed}^w_{\le k}(X,Y) = \mathsf{ed}^w_{\le k}(X',Y')$。当算法在第 2 行返回 (X,Y) 时，这显然成立。如果 $\mathsf{ed}(X,Y) > k$，则 $\mathsf{ed}^{\le k}(X,Y) = \infty = \mathsf{ed}^{\le k}(a^{k+1},\varepsilon)$，因此 $\mathsf{ed}^w_{\le k}(X,Y) = \infty = \mathsf{ed}^w_{\le k}(a^{k+1},\varepsilon)$ 也成立，因为采用归一化权重函数的加权编辑距离至少与无权编辑距离一样大。在 $\mathsf{ed}(X,Y) \le k$这一剩余情形中，我们保持如下不变式：在 for 循环每次迭代开始时，以及每次执行第 10 行和第 13 行之后，$|X'| - |Y'| = x_t - y_t$ 和 $\mathsf{ed}^w_{\le k}(X,Y) = \mathsf{ed}^w_{\le k}(X' \cdot P \cdot X[x_t\,..\,x_m)\ Y' \cdot P \cdot Y[y_t\,..\,y_m))$ 均成立。容易看出，字符串 $X' \cdot P \cdot X[x_t\,..\,x_m)$ 和 $Y' \cdot P \cdot Y[y_t\,..\,y_m)$ 仅在第 10 行发生变化，此时将 P替换为 $\mathsf{StringReduction}(P,k)$。根据 $\mathsf{ed}^w_{\le k}$-等价关系的定义（定义 2.8），这一步的正确性显然成立，因为 $\mathsf{StringReduction}(P,k)$ 与 P是 $\mathsf{ed}^w_{\le k}$-等价的。

接下来，我们证明返回的字符串长度至多为 $85k^4$。当算法在第 2 行或第 3行终止时，这一点显然成立。否则，我们应用事实 2.7，注意到 X 可分解为

至多 k 个字符，其中 $\mathcal{A}$ 会删除或替换这些字符（这些字符会被复制到 X'），以及至多$k+1$ 个极大片段，其中 $\mathcal{A}$ 与 Y 中的片段完美匹配（这些片段会在应用 X'后被复制到 字符串规约）。根据推论 2.14的保证，我们可以得出 $|X'| \leq k + (k+1) \cdot 42k^3 \leq 85k^4$。对称地，我们有 $|Y'| \leq 85k^4$。

接下来需要分析我们这一过程的时间复杂度。我们使用 Landau–Vishkin 算法 [LV88] 来检查 $\mathsf{ed}(X,Y) \leq k$ 是否成立；若成立，则构造对齐 $\mathcal{A}$。这一操作的代价为$\mathcal{O}(n+k^2)$ 时间，即 $\mathcal{O}(n)$，因为仅当 $n \geq k^4 \geq k^2$时我们才执行此步骤。扫描对齐 $\mathcal{A}$ 需要 $\mathcal{O}(m) = \mathcal{O}(n)$ 时间，其中包括推论 2.14的应用；这些应用处理的字符串总长度至多为 n。 □

将字符串长度缩减至 $\mathcal{O}(k^4)$ 后，我们可以使用经典动态规划 [WF74]，在 $\mathsf{ed}^w_{\leq k}(X,Y)$ $\mathcal{O}(k^8)$ 时间内完成计算。然而，由于 w 是规范化的， [WF74]的运行时间可以降至 $\mathcal{O}(nk)$。为完整起见，我们将在下文描述这一改进。

命题 2.16. 给定长度至多为 n 的字符串 X、Y，一个整数 $k \in \mathbb{Z}_+$，以及一个权重函数 w，可在 $\mathcal{O}(nk)$ 时间内计算 $\mathsf{ed}^w_{\leq k}(X,Y)$ 的值。

证明。 回想一下， [WF74] 算法维护一个表 $D[0\,..\,|X|, 0\,..\,|Y|]$，使得 $D[i,j] = \mathsf{ed}^w(X[0\,..\,i)\ Y[0\,..\,j))$ 对每个 $i \in [0\,..\,|X|]$ 和 $j \in [0\,..\,|Y|]$ 都成立。我们有 $D[0,0] = 0$，而其余条目则通过以下公式分别在 $\mathcal{O}(1)$ 时间内构造：

$$D[i,j] = \min \begin{cases} D[i-1,j] + w(X[i-1], \varepsilon) & \text{if } i > 0, \\ D[i,j-1] + w(\varepsilon, Y[j-1]) & \text{if } j > 0, \\ D[i-1,j-1] + w(X[i-1], Y[j-1]) & \text{if } i,j > 0. \end{cases} \tag{1}$$

为了计算 $\mathsf{ed}^w_{\leq k}(X,Y)$，我们使用一个经过修改的表 $D'[0\,..\,|X|, 0\,..\,|Y|]$，使得 $D'[0,0] = 0, D'[i,j] = \infty$（如果 $|i-j| > k$），而其余条目则使用 (1) 计算（将 D 替换为D'）。通过直接的归纳论证可知，$D'[i,j] \geq D[i,j]$ 对所有 $i \in [0\,..\,|X|]$ 和$j \in [0\,..\,|Y|]$ 都成立，而且 $D[i,j] \leq k$蕴含 $D'[i,j] = D[i,j]$。对于 $|i-j| > k$，这是正确的，因为w是规范化的，因此 $D[i,j] = \mathsf{ed}^w(X[0\,..\,i)\ Y[0\,..\,j)) \geq \mathsf{ed}(X[0\,..\,i)\ Y[0\,..\,j)) \geq |i-j| > k$。另一方面，对于 $|i-j| \leq k$，论证基于归纳假设以及权重函数 w 取非负值这一事实。可以隐式设置 $|i-j| > k$ 对应的条目 $D'[i,j] = \infty$，从而将运行时间降低到 $\mathcal{O}(nk)$。

□

定理 1.1. 给定长度至多为 n 的字符串 X, Y、整数 $k \in \mathbb{Z}_+$ 和权重函数 w，值 $\mathsf{ed}^w_{\leq k}(X,Y)$ 可以在 $\mathcal{O}(n+k^5)$ 时间内计算。

证明。我们首先应用定理 2.15，构造长度为 $\mathcal{O}(k^4)$ 的字符串 X', Y'，使得 $\mathsf{ed}^w_{\leq k}(X',Y') = \mathsf{ed}^w_{\leq k}(X,Y)$。然后，我们使用命题 2.16 计算 $\mathsf{ed}^w_{\leq k}(X',Y')$。这两个步骤的运行时间分别为 $\mathcal{O}(n)$ 和 $\mathcal{O}(k^4 \cdot k) = \mathcal{O}(k^5)$，总计为 $\mathcal{O}(n+k^5)$。 □

3 树编辑距离

3.1 预备知识

对于字母表 Σ，我们定义一个括号集合 $\mathsf{P}_\Sigma := \bigcup_{a \in \Sigma} \{(_a,)_a\}$，其括号的标签取自 Σ。节点标签取自 Σ 的森林是由 Σ 上的括号构成的平衡字符串。形式化地，标签取自 Σ 的森林集合定义为满足以下条件的最小子集 $\mathcal{F}_\Sigma \subseteq \mathsf{P}^*_\Sigma$：

- $\varepsilon \in \mathcal{F}_\Sigma$,

- $F \cdot G \in \mathcal{F}_\Sigma$ 对于每个 F、$G \in \mathcal{F}_\Sigma$,
- $(_a \cdot F \cdot)_a \in \mathcal{F}_\Sigma$ 对每个 $F \in \mathcal{F}_\Sigma$ 和 $a \in \Sigma$。

对于森林 F，我们将节点集合 V_F 定义为满足以下条件的配对集合 $(i,\ j) \in [0\,..\,|F|) : F[i]$ 是左括号，$F[j]$ 是右括号，且 $F[i\,..\,j]$ 是平衡的。对于节点 $u = (i,\ j) \in V_F$，我们用 $o(u) := i$ 和 $c(u) := j$ 表示左括号和右括号的位置。森林 F 是一棵树，当且仅当 $(0,\ |F| - 1) \in V_F$。

事实 3.1。森林 F 可以在线性时间内进行预处理，从而能够在常数时间内判断任意给定的片段 $F[i\,..\,j)$ 是否平衡。

证明。定义高度函数 $H : [0\,..\,|F|] \to \mathbb{Z}$ 使得 $H(i)$ 等于 $F[0\,..\,i)$ 中左括号的数量减去 $F[0\,..\,i)$。由于 F 是平衡的，片段 $F[i\,..\,j)$ 是平衡的，当且仅当 $H(i) = H(j) = \text{最小值}_{m\in[i\,..\,j]} H(m)$。使用区间最小值查询（RMQ）进行线性时间预处理后，可以在 $\mathcal{O}(1)$ 时间内检验这一条件 [HT 84]。 □

在 Σ 上带有节点标签的上下文是一个二元组 $C = \langle C_L; C_R\rangle \in \mathrm{P}_\Sigma \times \mathrm{P}_\Sigma$，满足 $C_L \cdot C_R$ 是一棵树。上下文 C 的节点集合 V_C 与其底层树 $C_L \cdot C_R$ 的节点集合相同。上下文 C 的深度是这样的节点 $u \in V_C$ 的数量：其左括号属于 C_L，且右括号属于 C_R，即 $o(u) < |C_L| < c(u)$。

上下文 C、D 的（垂直）组合得到一个上下文 $C \star D := \langle C_L \cdot D_L; D_R \cdot C_R\rangle$。此外，上下文 C 与森林 F 的垂直组合得到一棵树 $C \star F := C_L \cdot F \cdot C_R$。如果上下文 C 不能表示为 $e \geq 2$ 个相同上下文的垂直组合，则称其为本原上下文。

如果 $C_L = F[o(u)\,..\,o(u) + |C_L|)$ 且 $C_R = F(c(u) - |C_R|\,..\,c(u)],$，则称上下文 C 在节点 $u \in V_F$ 处出现在森林 F 中；等价地，对于某个森林 G，有 $F[o(u)\,..\,c(u)] = C \star G$

3.2 森林对齐与加权森林编辑距离

我们首先通过形式化定义森林对齐来讨论加权树编辑距离。森林对齐与字符串对齐类似，只是增加了少量限制，以确保这些对齐能够对森林进行有效编辑。

定义 3.2. 如果对每个 $u \in V_F$ 都满足以下一致性条件，我们就称对齐 $\mathcal{A} \in \mathsf{A}(F, G)$ 是森林 F 和 G 的森林对齐：

- 要么同时删除 $F[o(u)]$ 和 $F[c(u)],$，要么
- 存在 $v \in V_G$，使得 $F[o(u)] \sim_{\mathcal{A}} G[o(v)]$ 且 $F[c(u)] \sim_{\mathcal{A}} G[c(v)]$。

Th从 F 到 G 的所有森林对齐的集合记作 $\mathsf{TA}(F, G) \subseteq \mathsf{A}(F, G)$.

¯定义 $\overline{\mathrm{P}_\Sigma} = \mathrm{P}_\Sigma \cup \varepsilon$，以及一个映射 $\lambda : \mathrm{P}_\Sigma \to \overline{\Sigma}$，使得 $\lambda((a) = \lambda()a) = a$ 对每个 $a \in \Sigma$都成立，并且 $\lambda(\varepsilon) = \varepsilon$。对于权重函数 $w : \Sigma \times \Sigma \to \mathbb{R}_{\geq 0}$，我们定义相应的权重函数 $\tilde{w} : \mathrm{P}_\Sigma \times \mathrm{P}_\Sigma \to \mathbb{R}_{\geq 0}$，使得对所有 $p, q \in \mathrm{P}_\Sigma$，$\tilde{w}(p, q) = w(\lambda(p)\ \lambda(q))$。森林对齐 $\mathcal{A} \in \mathsf{TA}(F, G)$ 相对于权重函数 w 的代价定义为 $\mathsf{ted}^w_{\mathcal{A}}(F, G) := \frac{1}{2}\mathsf{ed}^{\tilde{w}}_{\mathcal{A}}(F, G)$。此外，对于任意两个森林 F, G，我们定义加权树编辑距离 $\mathsf{ted}^w(F, G) = \min^{\mathcal{A}\in\mathsf{TA}(F,G)} \mathsf{ted}^w_{\mathcal{A}}(F, G)$；对于阈值 $k \in \mathbb{R}_{\geq}0$，我们设

$$\mathsf{ted}^w_{\leq k}(F, G) = \begin{cases} \mathsf{ted}^w(F, G) & \text{if } \mathsf{ted}^w(F, G) \leq k, \\ \infty & \text{otherwise.} \end{cases}$$

¯如果 w 是 Σ 上的离散度量，则省略上标。

¯事实 3.3. 如果 w 是 Σ 上的拟度量，那么 ted^w 是 $\mathcal{F}_\Sigma$ 上的拟度量。在这种情况下，$\mathsf{ted}^w(F,G)$ 也可以定义为将 F 转换为 G 的编辑序列的最小代价，其中，插入一个标签为 b 的节点的代价为 $w(\varepsilon, b)$，删除一个标签为 a 的节点的代价为 $w(a,\varepsilon)$，将节点标签从 a 改为 b 的代价为 $w(a,b)$。

证明。考虑任意森林 F、G、$H \in \mathcal{F}_\Sigma$，以及对齐 $A = (x_t, y_t)_{t=0}^m \in \mathsf{TA}(F,G)$ 和 $B = (\hat{}y_t, \hat{}z_t)_{t=0}^{\hat{m}} \in \mathsf{TA}(G,H)$。我们可以像事实 2.5 的证明中那样构造乘积对齐 $\mathcal{A} \otimes \mathcal{B}$，它满足 $\mathsf{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(F,H) \le \mathsf{ed}^w_{\mathcal{A}}(F,G) + \mathsf{ed}^w_{\mathcal{B}}(G,H)$。因此，剩下要证明的是 $\mathcal{A}\otimes\mathcal{B}$ 是树对齐。

考虑任意节点 $u_F \in V_F$。如果 $\mathcal{A}$ 删除 u_F（也就是说，它删除了两个字符 $F[o(u_F)]$ 和 $F[c(u_F)]$），那么 $\mathcal{A}\otimes\mathcal{B}$ 也会删除 u_F；参见 $\mathcal{A}\otimes\mathcal{B}$ 的递归定义中的情形 2。另一种可能是，$\mathcal{A}$ 将 u_F 与某个节点 $u_G \in V_G$ 对齐（也就是说，它将 $F[o(u_F)]$ 与 $G[o(u_G)]$ 对齐，并将 $F[c(u_F)]$ 与 $G[c(u_G)]$ 对齐）。如果 $\mathcal{B}$ 删除 u_G，那么 $\mathcal{A}\otimes\mathcal{B}$ 会删除 u_F；参见 $\mathcal{A}\otimes\mathcal{B}$ 的递归定义中的情形 6。最后，如果 $\mathcal{B}$ 将 u_G 与某个节点 $u_H \in V_H$ 对齐，那么 $\mathcal{A}\otimes\mathcal{B}$ 会将 u_F 与 u_H 对齐；参见 $\mathcal{A}\otimes\mathcal{B}$ 的递归定义中的情形 7。□

3.3 组合基础

3.3.1 森林

与加权字符串编辑距离的讨论类似，在给出树编辑距离算法之前，我们先证明：对于输入实例森林中大型子森林的同步出现，存在编辑距离较小的等价森林。

定义 3.4。对于 $k \in \mathbb{Z}_{\ge 0}$ 和权重函数 w，若森林 P, P' 满足

$$\mathsf{ted}^w_{\le k}(F, P\cdot F\cdot P') = \mathsf{ted}^w_{\le k}(F[0\,..\,p_F)\cdot P'\cdot F[p_F+|P|\,..\,|F|), G[0\,..\,p_G)\cdot P'\cdot G[p_G+|P|\,..\,|G|)$$

则称其为 ted F -等价；该条件对所有满足 $|p_F - p_G| \le 2k$、且 P 分别在森林 F 和 G 中出现于位置 p_F 和 p_G 的情况均成立。

引理 3.5。设 $k \in \mathbb{Z}_+$，设 Q 是一个森林，并设 e、$e' \in \mathbb{Z}_{\ge 4k}$。那么，对于任意归一化权重函数 w，Q^e 和 $Q^{e'}$ 都是 $\mathsf{ted}^w_{\le k}$-等价的。

证明。我们不妨假设 Q 是本原的。（如果对于 $m \in \mathbb{Z}_{\ge 2}$ 有 $Q = R^m$，那么 $Q^e = R^{me}$ 和 $Q^{e'} = R^{me'}$ 可以解释为 R 的幂，而不是 Q 的幂。）假设 Q^e 出现在森林 F 和 G 中，位置分别为 p_F 和 p_G，并满足 $|p_F - p_G| \le 2k$。记 $F' = F[0\,..\,p_F)\cdot Q^{e'}\cdot F[p_F+|Q^e|\,..\,|F|)$ 和 $G' = G[0\,..\,p_G)\cdot Q^{e'}\cdot G[p_G+|Q^e|\,..\,|G|)$。此外，令 $q = |Q|$，并令 $\mathcal{A}$ 为一个森林对齐，使得 $\mathsf{ted}^w(F,G) = \mathsf{ted}^w_{\mathcal{A}}(F,G) \le k$。

命题 3.6。存在 $i_F, i_G \in [0\,..\,3k]$，使得

$$F[p_F + i_F\cdot q\,..\,p_F + (i_F+1)\cdot q) \simeq_{\mathcal{A}} G[p_G + i_G\cdot q\,..\,p_G + (i_G+1)\cdot q).$$

证明。令 $(f_b, g_b) \in \mathcal{A}$ 为 $\mathcal{A}$ 中满足 $f_b \ge p_F$ 和 $g_b \ge p_G$ 的最左元素。根据 F 和 G 之间的对称性，不失一般性地假设 $f_b = p_F$。考虑 F 中从位置 $p_F + i\cdot q$ 开始的 Q 的 $k+1$ 次出现，其中 $i \in [0\,..\,k]$。由于 Q 是平衡的，对齐 $\mathcal{A}$（无权代价至多为 k）会与其中至少一次出现完全匹配；因此我们可以定义 $i_F \in [0\,..\,k]$，使得 $\mathcal{A}$ 恰好与某个片段 $G[g_a\,..\,g_a+q]$ 匹配（$F[p_F+i_F\cdot q\,..\,p_F+(i_F+1)\cdot q)$）。根据 b 的定义，我们有 $a \ge b$，从而 $g_a \ge g_b \ge p_G$。此外，由于 $\mathsf{ted}_{\mathcal{A}}(F,G) \le k$ 且 $F[p_F+i_F\cdot q] \sim_{\mathcal{A}} G[g_a]$，我们有 $g_a \le (p_F + i_F\cdot q) + 2k \le p_F + kq + 2k \le p_G + kq + 4k \le p_G + 3kq$，其中最后一个不等式由 $q \ge 2$ 得出（请回想，Q 是平衡的，因此其长度为偶数）。此外，由于 Q 是本原的（即与其所有非平凡循环旋转均不同），我们得出对于某个 $i_G \in [0\,..\,3k]$，有 $g_a = p_G + i_G\cdot q$。

□

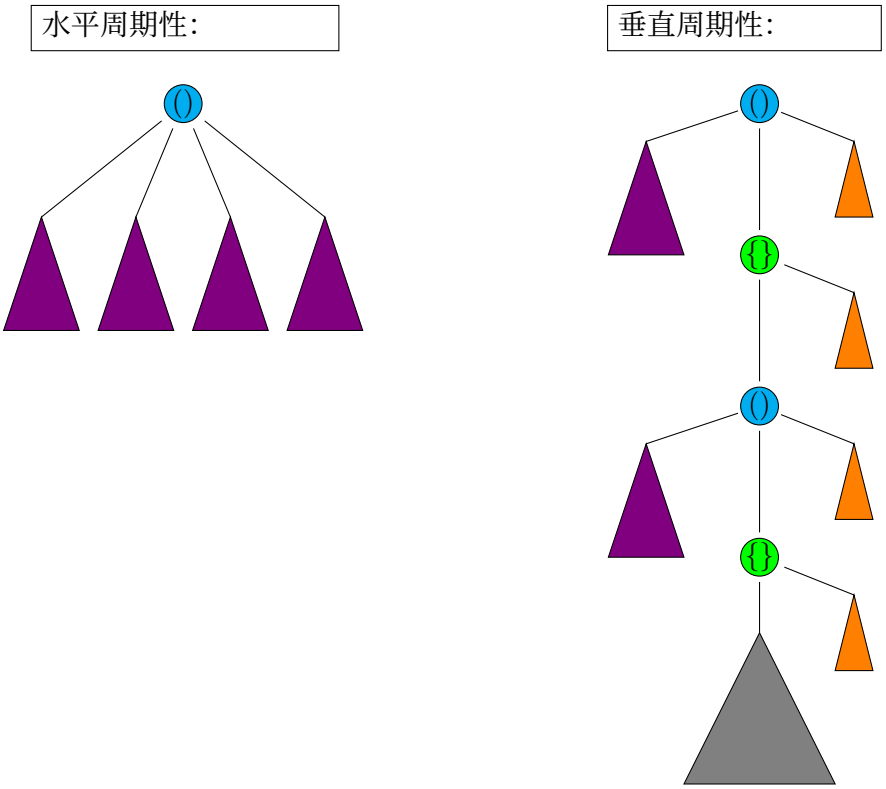

图3：左图：水平周期性，其字符串表示为“([. . .][. . .][. . .][. . .])”。右图：垂直周期性，其字符串表示为“([. . .]{([. . .]{[. . .][. . .]}[. . .])[. . .]}[. . .])”。

现在，若将 $Q^e = F[p_F \,..\, p_F + e \cdot q) = G[p_G \,..\, p_G + e \cdot q)$ 替换为 $Q^{e'}$（其中 $e' \geq e - 1$），我们可以将其解释为把 $Q = F[p_F + i_F \cdot q \,..\, p_F + (i_F + 1) \cdot q) = G[p_G + i_G \cdot q \,..\, p_G + (i_G + 1) \cdot q)$ 替换为 $Q^{1+e'-e}$。根据命题 3.6，$\mathcal{A}$ 可以直接调整而不改变其代价，因此 $\mathsf{ted}^w(F', G') \leq \mathsf{ted}^w_{\mathcal{A}}(F, G) = \mathsf{ted}^w(F, G)$。如果 $e' < e - 1$，我们就重复上述论证，每次将指数递减一步，仍可得出 $\mathsf{ted}^w(F', G') \leq \mathsf{ted}^w(F, G)$。无论哪种情况，根据 (F, G, e) 与 (F', G', e') 之间的对称性，反向不等式也成立。 □

若不存在长度为 $|Q| \in [1 \,..\, 4k]$ 的森林 Q 使得 Q^{4k+1} 出现在 F中，则称森林 F 避免水平 k-周期性。

引理3.7。设 $k \in \mathbb{Z}_+$，且 P、P'是长度分别为 $|P|$、$|P'| \geq 74k^3$且避免水平k-周期性的森林。那么，对于任意归一化拟度量 w，P 和 P' 都是 $\mathsf{ted}^w_{\leq k}$-等价的。

证明。假设 P 分别在森林 F 和 G 中出现在位置 p_F 和 p_G，并满足$|p_F - p_G| \leq 2k$。记 $F' = F[0 \,..\, p_F) \cdot P' \cdot F[p_F + |P| \,..\, |F|)$和 $G' = G[0 \,..\, p_G) \cdot P' \cdot G[p_G + |P| \,..\, |G|)$。

令 $\mathcal{A} = (f_t, g_t)_{t=0}^m$为一种对齐，使得 $\mathsf{ted}^w(F, G) = \mathsf{ted}^w_{\mathcal{A}}(F, G) \leq k$。此外，令 $(f_a, g_a) \in \mathcal{A}$ 为 $\mathcal{A}$ 中满足 $f_a \geq p_F$或 $g_a \geq p_G$ 的最左元素，并令 $(f_b, g_b) \in \mathcal{A}$为 $\mathcal{A}$ 中满足 $f_b \geq p_F + |P|$ 且 $g_b \geq p_G + |P|$的最左元素。我们构造一种对齐 $\mathcal{A}'$，使其：

- 以与 $G[0 \,..\, g_a)$ 相同的方式将 $F[0 \,..\, f_a)$ 与 $\mathcal{A}$ 对齐；
- 删除 $F[f_a \,..\, p_F)$ 并插入 $G[g_a \,..\, p_G)$（这些片段中至少有一个为空）；
- 匹配 $F[p_F \,..\, p_F + |P|) = P$与 $G[p_G \,..\, p_G + |P|) = P$；

- 删除 $F[p_F+|P|\,..\,f_b)$并插入 $G[p_G+|P|\,..\,g_b)$（这些片段中至少有一个为空）；
- 与 $F[f_b\,..\,|F|)$ 对齐的方式和 $\mathcal{A}$ 与 $G[g_b\,..\,|G|)$ 对齐的方式相同。

为了证明 $\mathcal{A}'$ 是森林对齐，我们来考察 F中的一个节点 u 的几种可能情况。

- 如果 u位于 $P = F[p_F\,..\,p_F+|P|)$ 内部，那么 $\mathcal{A}'$会将 u与 $P = G[p_G\,..\,p_G+|P|)$ 内部对应的节点匹配。
- 如果 u位于 $P = F[p_F\,..\,p_F+|P|)$ 外部，且 $\mathcal{A}$将 u与 $P = G[p_G\,..\,p_G+|P|)$ 内部的一个节点 v对齐，那么由于 $\mathcal{A}\ni(f_a,g_a),(f_b,g_b)$ 的非交叉性质，$o(u)\ c(u)\ \in [f_a\,..\,p_F)\cup[p_F+|P|\,..\,f_b)$。因此，$\mathcal{A}'$删除 u。
- 如果 u位于 $P = F[p_F\,..\,p_F+|P|)$ 外部，且 $\mathcal{A}$将 u与 $P = G[p_G\,..\,p_G+|P|)$ 外部的一个节点 v对齐，那么由于 $\mathcal{A}\ni(f_a,g_a),(f_b,g_b)$ 的非交叉性质，$o(u)\ c(u)\ \in [0\,..\,f_a)\cup[f_b\,..\,|F|)$。因此，$\mathcal{A}'$也将 u与 v对齐。
- 如果 u位于 $P = F[p_F\,..\,p_F+|P|)$ 之外，且 $\mathcal{A}$删除了 u，那么 $\mathcal{A}'$也会删除 u。

我们的下一个目标是证明 $\mathsf{ted}^w_{\mathcal{A}'}(F,G)\le\mathsf{ted}^w_{\mathcal{A}}(F,G)$。这依赖于以下断言。

命题 3.8. 存在 $t\in[a\,..\,b]$，使得 $f_t-g_t=p_F-p_G$。

证明。我们将 $P = F[p_F\,..\,p_F+|P|)$ 划分为表示对 $\mathcal{A}$ 的删除或替换的单个字符，以及与 G 的片段完美匹配的极大片段 $\mathcal{A}$。根据事实 2.7，这类片段的数量至多为 $2k+1$，且其总长度至少为 $|P|-2k$。因此，这些片段之一（记作 $R = F[r_F\,..\,r_F+|R|)$）的长度至少为 $\frac{|P|-2k}{2k+1}\ge 24k^2$。假设与 R 完美匹配的 G 的片段为 $G[r_G\,..\,r_G+|R|)$。如果 $r_F-p_F=r_G-p_G$，那么对于满足 $(f_t,g_t)=(r_F,r_G)$ 的 t，断言成立。否则，我们注意到 R 的周期为 $q:=|(r_F-p_F)-(r_G-p_G)|\in[1\,..\,4k]$。令 $q=R[0\,..\,q)$，并注意到 $\frac{|R|}{q}\ge\frac{24k^2}{4k}\ge 4k+2$。因此，$Q^{4k+2}$ 是 P 的子串；由于 P 不具有水平 k-周期性，我们得出结论：Q 的任何循环旋转都不是平衡的。

由于 Q 是平衡字符串 P的子串，这意味着 Q 中的左括号数量与 Q中的右括号数量不匹配。根据对称性（至多相差一个反转），不失一般性地假设 Q 中的左括号多于右括号。因此，存在一个节点 u，位于 F中，使得 $o(u)\in[r_F\,..\,r_F+q)$，但 $c(u)\ge r_F+|R|$。特别地，$c(u)-o(u)\ge|R|-q\ge 24k^2-4k>8k$。令 v、v' 为 G 中分别与 u 通过 $\mathcal{A}$ 和$\mathcal{A}'$匹配的节点。注意，$|o(v)-o(v')|\le 4k$，且 $|c(v)-c(v')|\le 4k$。由于 $c(v')-o(v')=c(u)-o(u)>8k$，我们得出结论：v 是 v' 的祖先节点，或反之亦然。无论哪种情况，我们都有$0\ge(o(v')-o(v))\cdot(c(v')-c(v))=((o(u)-o(v))-(p_F-p_G))\cdot((c(u)-c(v))-(p_F-p_G))$。值 $(f_t-g_t)-(p_F-p_G)$ 在后续索引 t处至多变化一。该值在 $(f_t,g_t)=(o(u)\ o(v))$ 和 $(f_t,g_t)=(c(u)\ c(v))$ 处符号不同，因此它必定在某个中间索引 t处等于 0。 □

对齐 $\mathcal{A}$和 $\mathcal{A}'$仅在如何将 $F[f_a\,..\,f_t)$ 与 $G[g_a\,..\,g_t)$、以及 $F[f_t\,..\,f_b)$ 与 $G[g_t\,..\,g_b)$ 对齐方面有所不同；根据事实 2.6，$\mathcal{A}'$给出了这些片段的一个最优对齐。现在，如果将 $P = F[p_F\,..\,p_F+|P|)$ $= G[p_G\,..\,p_G+|P|)$ 修改为 P'，那么无需改变代价即可轻易调整 $\mathcal{A}'$，因此 $\mathsf{ted}^w(F',G')\le\mathsf{ted}^w_{\mathcal{A}'}(F,G)=\mathsf{ted}^w(F,G)$。反向不等式可由 (F,G,P) 与 (F',G',P') 之间的对称性得出。 □

3.3.2 上下文

定义 3.9。对于 $k \in \mathbb{Z}_{\geq 0}$ 和权重函数 w，上下文 $P = \langle P_L;\ P_R \rangle$ 和 $P' = \langle P'_L;\ P'_R \rangle$ 称为 $\mathsf{ted}^w_{\leq k}$-等价，如果

$$\mathsf{ted}^w_{\leq k}(F, G) = \mathsf{ted}^w_{\leq k}(F[0\,..\,o(u)) \cdot P'_L \cdot F[o(u) + |P_L|\,..\,c(u) - |P_R|] \cdot P'_R \cdot F(c(u)\,..\,|F|),\\ G[0\,..\,o(v)) \cdot P'_L \cdot G[o(v) + |P_L|\,..\,c(u) - |P_R|] \cdot P'_R \cdot G(c(v)\,..\,|G|))$$

对于所有森林 F 和 G，若 P 分别出现在节点 u 和 v 处，并满足 $|o(u) - o(v)| \leq 2k$ 以及 $|c(u) - c(v)| \leq 2k$，则上述结论成立。

引理 3.10. 设 $k \in \mathbb{Z}_+$，令 Q 为一个上下文，并令 e、$e' \in \mathbb{Z}_{\geq 6k}$。那么，对于每个归一化权重函数 w，Q^e 和 $Q^{e'}$ 是 $\mathsf{ted}^w_{\leq k}$-等价的。

证明. 不失一般性，我们假设 Q 是本原的。（若 $Q = R^m$ 对于 $m \in \mathbb{Z}_{\geq 2}$，则 $Q^e = R^{me}$ 和 $Q^{e'} = R^{me'}$ 可以解释为 R 的幂，而不是 Q 的幂。）令 $Q = \langle Q_L;\ Q_R \rangle$ 满足 $q_L = |Q_L|$ 和 $q_R = |Q_R|$。假设 Q^e 分别出现在森林 F 和 G 的节点 u 和 v 处，并满足 $|o(u) - o(v)| \leq 2k$ 以及 $|c(u) - c(v)| \leq 2k$。记

$$F' = F[0\,..\,o(u)) \cdot Q_L^{e'} \cdot F[o(u) + |Q_L^e|\,..\,c(u) - |Q_R^e|] \cdot Q_R^{e'} \cdot F(c(u)\,..\,|F|),\\ G' = G[0\,..\,o(v)) \cdot Q_L^{e'} \cdot G[o(v) + |Q_L^e|\,..\,c(u) - |Q_R^e|] \cdot Q_R^{e'} \cdot G(c(v)\,..\,|G|).$$

对于 $i \in [0\,..\,e)$，令 u_i 为 F 中满足 $o(u_i) = o(u) + i \cdot q_L$（以及 $c(u_i) = c(u) - i \cdot q_R$）的节点，并令 v_i 为 G 中满足 $o(v_i) = o(v) + i \cdot q_L$（以及 $c(v_i) = c(v) - i \cdot q_R$）的节点。此外，令 $\mathcal{A}$ 为一个最优森林对齐，使得 $\mathsf{ted}(F, G) = \mathsf{ted}_{\mathcal{A}}(F, G) \leq k$。

断言 3.11。存在 $i_F, i_G \in [0\,..\,5k]$ 使得

$$F[o(u) + i_F \cdot q_L\,..\,o(u) + (i_F + 1) \cdot q_L) \simeq_{\mathcal{A}} G[o(v) + i_G \cdot q_L\,..\,o(v) + (i_G + 1) \cdot q_L),\\ F(c(u) - (i_F + 1) \cdot q_R\,..\,c(u) - i_F \cdot q_R] \simeq_{\mathcal{A}} G(c(v) - (i_F + 1) \cdot q_R\,..\,c(v) - i_G \cdot q_R].$$

证明。令 $(f_b, g_b) \in \mathcal{A}$ 为 $\mathcal{A}$ 中满足 $f_b \geq o(u)$ 且 $g_b \geq o(v)$ 的最左元素。根据 F 与 G 的对称性，不失一般性地假设 $f_b = o(u)$。考虑 F 中位于位置 $(o(u) + i \cdot q_L, c(u) + i \cdot q_R)$（$i \in [0\,..\,k]$）的 $k+1$ 个互不相交的 C 出现。对齐 $\mathcal{A}$（无权代价至多为 k）必须将这些出现之一与 G 中的某个上下文完美匹配。我们选取一个如此完美匹配的出现的索引 $i_F \in [0\,..\,k]$，并假设它出现在 G 的节点 v' 处。

特别是，

$$F[o(u_{i_F})\,..\,o(u_{i_F}) + q_L) \simeq_{\mathcal{A}} G[o(v')\,..\,o(v') + q_L),\\ F(c(u_{i_F}) - q_R\,..\,c(u_{i_F})] \simeq_{\mathcal{A}} G(c(v') - q_R\,..\,c(v')].$$

由于 $(f_b, g_b) \in \mathcal{A}$，根据 $\mathcal{A}$ 的非交叉性质，必有 $o(v') \geq g_b \geq o(v)$。同时，由于 $\mathcal{A}$ 的无权代价不超过 k，因此有 $o(v') \leq o(u_{i_F}) + 2k \leq o(u) + kq_L + 2k \leq o(v) + kq_L + 4k \leq o(v) + 5kq_L$。类似地，$c(v') \geq c(v) - 5kq_R$，这也意味着 $c(v') \leq c(v)$。

我们的下一个目标是证明，对于某个 $i_G \in [0\,..\,5k]$，$v' = v_{i_G}$。为反证，假设对于某个 $i \in [0\,..\,5k$，$o(v_i) < o(v') < o(v_{i+1})$。由于 $c(v') > c(v) - 5kq_R$，这也意味着 $c(v_i) > c(v') > c(v_{i+1})$，即 v' 是连接 v_i 和 v_{i+1} 的路径上的一个节点。假设这条路径的长度为 ℓ，且节点 v' 与 v_i 之间的距离为 ℓ'。因此，$G[o(v_i)\,..\,o(v'))$ 有

ℓ' 个未匹配的左括号，占 Q_L中 ℓ 个未匹配的左括号。此外，$G[o(v_i)\,..\,o(v'))\cdot G[o(v')\,..\,o(v_{i+1}))$ $= Q_L = G[o(v')\,..\,o(v_{i+1}))\cdot G[o(v_i)\,..\,o(v'))$，因此存在一个本原串 Q_L，使得 $G[o(v')\,..\,o(v_{i+1}))$ 和 $G[o(v_i)\,..\,o(v'))$ 都是 Q_L 的幂。未匹配左括号的数量为 Q_L，它必须是 ℓ 和 ℓ' 的公因数，即 Q_L可以表示为指数为 $\ell/\gcd(\ell,\ell')$ 的字符串幂。对称地论证可知，Q_R可以表示为指数为 $\ell/\gcd(\ell,\ell')$ 的字符串幂。综上，我们得出结论：C可以表示为指数为 $\ell/\gcd(\ell,\ell')$ 的上下文幂，这与 C的本原性矛盾。因此，如所述，对于某个 $i_G\in[0\,..\,5k]$，$v'=v_{i_G}$成立；特别地，$o(v')=o(v)+i_Gq_L$且$c(v')=c(v)-i_Gq_R$。

□

现在，如果将节点 u,v处的 Q^e出现替换为 $e'\geq e-1$个 $Q^{e'}$，我们可以将其解释为把节点 u_{i_F},v_{i_G}处的 Q出现替换为 $Q^{1+e'-e}$。根据断言 3.11，$\mathcal{A}$可以轻易调整而不改变其代价，因此 $\mathsf{ted}^w(F',G')\leq\mathsf{ted}^w_{\mathcal{A}}(F,G)=\mathsf{ted}^w(F,G)$。如果 $e'<e-1$，我们重复上述论证，每次将指数递减一步，仍可得出 $\mathsf{ted}^w(F',G')\leq\mathsf{ted}^w(F,G)$。无论哪种情况，通过 (F,G,e) 与 (F',G',e') 之间的对称性，都可得到反向不等式。

□

我们称上下文 $P=\langle P_L;P_R\rangle$ 避免垂直 k-周期性，若它无法表示为 $P=C\star Q^{6k+1}\star D$，其中上下文 C,Q,D 满足 $|Q|\in[1\,..\,8k]$。

引理 3.12. 设 $k\in\mathbb{Z}_+$，设 $P=\langle P_L;P_R\rangle,P'=\langle P'_L;P'_R\rangle$是长度为 $|P_L|+|P_R|,|P'_L|+|P'_R|\geq 578k^4$ 的上下文，它们避开垂直 k-周期性，且其两半不包含长度超过 $74k^3$的平衡子字符串。那么，对于每个归一化权重函数 w，P和 P'是 $\mathsf{ted}^w_{\leq k}$-等价的。

证明. 假设 P分别出现在森林 F和 G中的节点 u和 v处，且满足 $|o(u)-o(v)|\leq 2k$和 $|c(u)-c(v)|\leq 2k$。记

$$F'=F[0\,..\,o(u))\cdot P'_L\cdot F[o(u)+|P_L|\,..\,c(u)-|P_R|]\cdot P'_R\cdot F(c(u)\,..\,|F|),$$
$$G'=G[0\,..\,o(v))\cdot P'_R\cdot G[o(v)+|P_L|\,..\,c(v)-|P_R|]\cdot P'_R\cdot G(c(v)\,..\,|G|).$$

设 $\mathcal{A}=(f_t,g_t)_{t=0}^m$是一个最优森林对齐，使得 $\mathsf{ted}(F,G)=\mathsf{ted}_{\mathcal{A}}(F,G)\leq k$。此外，设 $(f_a,g_a)\in\mathcal{A}$是 $\mathcal{A}$中满足 $f_a\geq o(u)$或 $g_a\geq o(v)$的最左元素；设 $(f_b,g_b)\in\mathcal{A}$是 $\mathcal{A}$中满足 $f_b\geq o(u)+|P_L|$且 $g_b\geq o(v)+|P_L|$的最左元素；设 $(f_c,g_c)\in\mathcal{A}$是 $\mathcal{A}$中满足 $f_c>c(u)-|P_R|$或 $g_c>c(v)-|P_R|$的最左元素；并设 (f_d,g_d)是 $\mathcal{A}$中满足 $f_d>c(u)$且 $g_d>c(v)$的最左元素。我们构造一个对齐 $\mathcal{A}'$，使其满足：

- 以与 $\mathcal{A}$ 相同的方式将 $F[0\,..\,f_a)$ 与 $G[0\,..\,g_a)$ 对齐；
- 删除 $F[f_a\,..\,o(u))$ 并插入 $G[g_a\,..\,o(v))$（这些片段中至少有一个为空）；
- 将 $F[o(u)\,..\,o(u)+|P_L|)=P_L$与 $G[o(v)\,..\,o(v)+|P_L|)$ 匹配 $=P_L$；
- 如果 $b>c$，则删除 $F[o(u)+|P_L|\,..\,c(u)-|P_R|]$并插入 $G[o(v)+|P_L|\,..\,c(v)-|P_R|]$；
- 如果 $b\leq c$，则删除 $F[o(u)+|P_L|\,..\,f_b)$ 并插入 $G[o(v)+|P_L|\,..\,g_b)$（这些片段中至少有一个为空）；
- 如果 $b\leq c$，则以与 $\mathcal{A}$ 相同的方式将 $F[f_b\,..\,f_c)$ 与 $G[g_b\,..\,g_c)$ 对齐；
- 如果 $b\leq c$，则删除 $F[f_c\,..\,c(u)-|P_R|]$并插入 $G[g_c\,..\,c(v)-|P_R|]$（这些片段中至少有一个为空）；

- 匹配 $F(c(u)-|P_R|\,..\,c(u)]=P_R$与 $G(c(v)-|P_R|\,..\,c(v)]=P_R$；
- 删除 $F(c(u)\,..\,f_d)$并插入 $G(c(v)\,..\,g_d)$（这些片段中至少有一个为空）；
- 以与 $\mathcal{A}$相同的方式将 $F[f_d\,..\,|F|)$ 与 $G[g_d\,..\,|G|)$ 对齐。

为了证明 $\mathcal{A}'$ 是一种森林对齐，让我们考虑 F中的一个节点 u' 的几种可能性 .

- 如果 u' 属于 $P = \langle F[o(u) \,..\, o(u) + |P_L|);\ F(c(u) - |P_R| \,..\, c(u)]\rangle$，那么 $\mathcal{A}'$ 会与属于 $P = \langle G[o(v) \,..\, o(v) + |P_L|);\ G(c(v) - |P_R| \,..\, c(v)]\rangle$ 的对应节点匹配。
- 如果 u' 在 $F[o(u) \,..\, c(u)]$ 之外，且 $\mathcal{A}$ 与 G 内部的一个节点 v' 对齐 u'，$],$ 那么 $o(u')\ c(u') \in [f_a \,..\, o(u)) \cup (c(u) \,..\, f_d)$，这是因为 $\mathcal{A} \ni (f_a, g_a)$、$(f_d, g_d)$ 的非交叉性质 .

  因此，$\mathcal{A}'$ 删除 u'。
- 如果 u' 在 $F[o(u) \,..\, c(u)]$ 之外，且 $\mathcal{A}$ 与 G 外部的一个节点 v' 对齐 u'，$],$ 那么 $o(u')\ c(u') \in [0 \,..\, f_a) \cup [f_d \,..\, |F|)$，这是因为 $\mathcal{A} \ni (f_a, g_a)$、$(f_d, g_d)$ 的非交叉性质。因此，$\mathcal{A}'$ 也与 v' 对齐 u'。

- 如果 u' 在 $F[o(u) \,..\, c(u)]$ 之外，且 $\mathcal{A}$ 删除 u'，那么 $\mathcal{A}'$ 也删除 u'。
- 如果 $b > c$ 和 u' 位于 $F[o(u) + |P_L| \,..\, c(u) - |P_R|]$ 内部，那么 $\mathcal{A}'$ 会删除 u'。
- 如果 $b \le c$, u' 位于 $F[o(u) + |P_L| \,..\, c(u) - |P_R|]$ 内部，且 $\mathcal{A}$ 与 G 外部的一个节点 v' 对齐，即 u' 与该节点对齐，那么由于 $\mathcal{A} \ni (f_b, g_b), (f_c, g_c)$ 的非交叉性质，$o(u')\ c(u') \in [o(u) + |P_L| \,..\, f_b) \cup [f_c \,..\, c(u)$。因此，$\mathcal{A}'$ 会删除 u'。
- 如果 $b \le c$, u' 位于 $F[o(u) + |P_L| \,..\, c(u) - |P_R|]$ 内部，且 $\mathcal{A}$ 与 G 内部的一个节点 v' 对齐，即 u' 与该节点对齐，那么由于 $\mathcal{A} \ni (f_b, g_b), (f_c, g_c)$ 的非交叉性质，$o(u')\ c(u') \in [f_b \,..\, f_c)$。因此，$\mathcal{A}'$ 也与 v' 对齐。

- 如果 $b \le c$, u' 位于 $F[o(u) + |P_L| \,..\, c(u) - |P_R|]$ 内部，并且 $\mathcal{A}$ 删除 u'，那么 $\mathcal{A}'$ 也会删除 u'。

现在我们来证明 $\mathsf{ted}^w_{\mathcal{A}'}(F, G) \le \mathsf{ted}^w_{\mathcal{A}}(F, G)$。这依赖于以下断言。

断言 3.13. 存在 $t_L \in [a \,..\, b]$，使得 $f_{t_L} - g_{t_L} = o(u) - o(u)$；并且存在 $t_R \in [c \,..\, d]$，使得 $f_{t_L} - g_{t_L} = c(u) - c(u)$。

证明。利用对称性（至多相差一个反转），不失一般性地，我们可以只考虑第一个断言。此外，利用 F 与 G 之间的对称性，不失一般性地，我们可以假设 $f_a = o(u)$；特别地，这意味着 $f_a - g_a \ge o(u) - o(v)$。如果存在 $t \in [a \,..\, b]$，使得 $f_t - g_t \le o(u) - o(v)$，那么，由于 $f_t - g_t$ 在后续位置上的变化至多为 1，因此也存在 $t_L \in [a \,..\, b]$，使得 $f_{t_L} - g_{t_L} = o(u) - o(v)$。因此，剩下的情况是，对于所有 $t \in [a \,..\, b]$，$f_t - g_t > o(u) - o(v)$ 均成立。

令我们将 P 表示为 e 个上下文 $P = P_0 \star \cdots \star P_{e-1}$ 的垂直组合，其中 e 是 P 的深度。注意，对于每个 $i \in [0 \,..\, e)$，F 中节点 u 处的 P 的出现以及 G 中 v 处的出现，会诱导出 F 中某些节点 u_i 处以及 G 中 v_i 处的 P_i 的出现。由于 $F(o(u_i) \,..\, o(u_i) + |P_{i,L}|)$ 和 $F(c(u_i) - |P_{i,R}| \,..\, c(u_i))$ 是平衡的，我们得出 $|P_i| \le 2 \cdot (74k^3 + 1) \le 150k^3$。我们可以将 $[0 \,..\, e)$ 分解为至多 k 个单个索引 i，使得 $\mathcal{A}$ 与 v_i 处 P_i 的出现不完全匹配；还可分解为至多 $k + 1$ 个区间 $[i \,..\, i')$，使得 $\mathcal{A}$ 与 v_i 处的出现 $P_i \star \cdots \star P_{i'-1}$ 完全匹配到 F 中的一个上下文。我们选择这样一个使 $|P_i \star \cdots \star P_{i'-1}|$ 最大化的区间 $[i \,..\, i')$；其长度至少为 $\frac{578k^4 - k \cdot 150k^3}{k+1} \ge 214k^3$。令 $i'' \in [i \,..\, i')$ 为满足 $|P_{i''} \star \cdots \star P_{i'-1}| > 8k$ 的最大索引；注意 $|P_i \star \cdots \star P_{i''-1}| \ge 214k^3 - (150k^3 + 8k) \ge 56k^2$。

对于每个 $j \in [i \,..\, i'']$，将 u'_j 记为与 v_j 由 $\mathcal{A}$ 匹配的节点。注意，$|o(u'_j) - o(u_j)| \le 4k$，且 $|c(u'_j) - c(u_j)| \le 4k$。此外，$o(u'_j) - o(v_j) > o(u) - o(v) = o(u_j) - o(v_j)$ 蕴含 $o(u'_j) > o(u_j)$。由于 $|P_j \star \cdots \star P_{i'-1}| > 8k$，我们得出结论，存在某个 $j' \in [j \,..\, i')$，使得 $u'_j = u_{j'}$。此外，若 $j > i$，则 u'_j 必须是 u'_{j-1} 的子节点。因此，存在 $\delta > 0$，使得对于所有 $j \in [i \,..\, i'']$，$u'_j = u_{j+\delta}$ 均成立。对于 $j \in [i \,..\, i'')$，这意味着 $P_j = P_{j+\delta}$，并且 $P_j \star \cdots \star P_{j+\delta-1}$ 的两半长度均至多为 $4k$。特别地，若定义 $Q = P_i \star \cdots \star P_{i+\delta-1}$，那么由于 $\frac{|P_i \star \cdots \star P_{i''-1}|}{8k} \ge \frac{56k^2}{8k} \ge 6k + 1$，我们得出结论：$Q^{6k+1}$ 分别在 F 和 G 中的位置 u_i 和 v_i 处出现。这与 P 不具有垂直周期性的假设矛盾。 □

对齐 $\mathcal{A}$和 $\mathcal{A}'$ 唯一的区别在于它们如何对齐以下片段 s:

- $F[f_a \,..\, f_{t_L}]$ 与 $G[g_a \,..\, g_{t_L}]$：此处，$\mathcal{A}'$ 将一个片段与另一个片段的后缀完美匹配；根据事实 2.6，这是最优的。
- $F[f_{t_L} \,..\, f_{t_R}]$ 与 $G[g_{t_L} \,..\, g_{t_R}]$ 的情况是 $b > c$：此处，$\mathcal{A}'$ 的代价等于删除 $F[o(u) + |P_L| \,..\, c(u) - |P_R|]$ 并插入 $G[o(v) + |P_L| \,..\, c(v) - |P_R|]$ 的代价。根据事实 2.6，这两项代价都不超过 $\mathcal{A}$ 对齐 $F[f_{t_L} \,..\, f_c]$ 与 $G[g_{t_L} \,..\, g_c]$，以及对齐 $F[f_b \,..\, f_{t_R}]$ 与 $G[g_b \,..\, g_{t_R}]$ 的代价。

- $F[f_{t_L} \,..\, f_b]$ 与 $G[g_{t_L} \,..\, g_b]$ 的情况是 $b \le c$：此处，$\mathcal{A}'$ 匹配使一个片段与另一个片段的前缀完全吻合；根据事实 2.6，这是最优的。
- $F[f_c \,..\, f_{t_R}]$ 与 $G[g_c \,..\, f_{t_R}]$ 的情况是 $b \le c$：此处，$\mathcal{A}'$ 匹配使一个片段与另一个片段的后缀完全吻合；根据事实 2.6，这是最优的。
- $F[f_{t_R} \,..\, f_d)$ 与 $G[g_{t_R} \,..\, g_d)$：在此，$\mathcal{A}'$ 与另一方的一个前缀完美匹配；根据事实2.6，这是最优的。

若将 P 处于 F 中的节点 u 以及 G 中的节点 v 的出现修改为 P' 的出现，则可以直接调整 $\mathcal{A}'$，而不改变其代价，因此 $\mathsf{ted}^w(F', G') \le \mathsf{ted}^w_{\mathcal{A}'}(F, G) = \mathsf{ted}^w(F, G)$。由 (F, G, P) 与 (F', G', P') 之间的对称性可得反向不等式。 □

3.4 算法

我们称森林 F 的一片为一个平衡片段 $F[i \,..\, j)$，或为一对片段 $\langle F[i \,..\, i'),\ F[j' \,..\, j)\rangle$，它们构成一个上下文，也就是说，$F[i \,..\, j)$ 是一棵树，而 $F[i' \,..\, j')$ 是平衡的。对于一个片段 $F[i \,..\, j)$，我们用 $\mathcal{P}(F[i \,..\, j))$ 表示包含在 $F[i \,..\, j)$ 中的片的集合。此外，令 $\mathcal{P}(F) = \mathcal{P}(F[0 \,..\, |F|))$。

定义 3.14. 若集合 $\mathsf{D} \subseteq \mathcal{P}(F[i \,..\, j))$ 是森林 F 中平衡片段 $F[i \,..\, j)$ 的片段分解，并满足以下条件之一：

- $\mathsf{D} = \emptyset$ 和 $i = j$;
- $\mathsf{D} = \{F[i \,..\, j)\}$ 和 $i < j$;
- 对于某些片段分解 D_L（对应 $F[i \,..\, m)$）和 D_R（对应 $F[m \,..\, j)$），其中 $m \in (i \,..\, j)$，有 $\bullet\mathsf{D} = \mathsf{D}_L \cup \mathsf{D}_R$。
- $\mathsf{D} = \{\langle F[i \,..\, i'),$ 对于上下文 $\langle F[i \,..\, i'),\ F[j' \,..\, j)\rangle \in \mathcal{P}(F)$ 以及片段分解 D'（对应 $F[i' \,..\, j')$，有 $F[j' \,..\, j)\rangle\} \cup \mathsf{D}'$。

引理 3.15. 存在线性时间算法，对于给定的森林 F 和整数 $t \ge 2$，可构造出 F 的片段分解 D，该分解至多包含 $\max(1, \frac{6|F|}{t} - 1)$ 个长度至多为 t 的片段。

算法 4： $\mathcal{D}(i, j)$：构造平衡片段 $F[i \,..\, j)$ 的分解。

```
1  如果 j = i 则返回 ∅;
2  如果 j ≤ i + t 则返回 {F[i..j)};
3  i' ← i; j' ← j;
4  当 true 时执行                                   // F[i'..j') 是平衡的，并且 (i' − i) + (j − j') ≤ t
5  |  令 m ∈ [i'..j'] 满足条件：F[i'..m) 是一棵树;
6  |  如果 (m − i) + (j − j') ≤ t 则 i' ← m;
7  |  否则如果 m < j' 并且 (i' − i) + (j − m) ≤ t 则 j' ← m;
8  |  否则如果 F[i..j) 不是一棵树，则返回 D(i, i') ∪ D(i', m) ∪ D(m, j') ∪ D(j', j);
9  |  否则如果 m = j' 并且 (i' + 1 − i) ≤ (j − j' + 1) ≤ t 则 i' ← i' + 1; j' ← j' − 1;
10 |  否则返回 {⟨F[i..i'), F[j'..j)⟩} ∪ D(i', m) ∪ D(m, j');
```

证明。算法 4 提供了一个递归过程，对于 F的每个平衡片段 $F[i\,..\,j)$，构造由大小至多为 t的片段组成的 $F[i\,..\,j)$ 的片段分解 $\mathcal{D}(i,j)$。在 $j=i$和 $j\in(i\,..\,i+t]$ 的边界情况下，我们分别返回 $\mathcal{D}(i,j)=\emptyset$和 $\mathcal{D}(i,j)=\{F[i\,..\,j)\}$。否则，我们迭代扩展片段 $F[i\,..\,i')$ 和 $F[j'\,..\,j)$（初始为空），同时维护以下不变量：

(a) $F[i'\,..\,j')$ 是平衡的；(b) $|F[i\,..\,i')|+|F[j'\,..\,j)|\le t$；(c) 如果 $F[i\,..\,j)$ 是一棵树，那么 $F[i\,..\,i')=F[j'\,..\,j)=\varepsilon$或 $\langle F[i\,..\,i'); F[j'\,..\,j)\rangle$是一个上下文；(d) 如果 $F[i\,..\,j)$ 不是一棵树，那么 $F[i\,..\,i')$ 和 $F[j'\,..\,j)$ 是平衡的。

在每次迭代中，我们找出一个位置 $m\in(i'\,..\,j']$使得 $F[i'\,..\,m)$是一棵树（由于 $j-i>t$且根据不变式 (a)、(b)，这样的位置总是存在）。

(1) 只要不违反不变式 (b)，我们就设置 $i'\leftarrow m$。(2) 如果 $m\neq j'$，只要不违反不变式 (b)，我们就设置 $j\leftarrow m$。(3) 如果 $m=j'$且 $F[i\,..\,j)$是一棵树，只要不违反不变式 (b)，我们就设置 $(i',j')\leftarrow(i'+1,j'-1)$。(4) 否则，如果 $F[i\,..\,j)$是一棵树，我们返回 $\mathcal{D}(i,j):=\{\langle F[i\,..\,i'); F[j'\,..\,j)\rangle\}\cup\mathcal{D}(i',m)\cup\mathcal{D}(m,j')$；如果 $F[i\,..\,j)$不是一棵树，我们返回 $\mathcal{D}(i,j):=\mathcal{D}(i,i')\cup\mathcal{D}(i',m)\cup\mathcal{D}(m,j')\cup\mathcal{D}(j',j)$。

容易看出，情况 (1)–(3) 保持了不变量，因此情况 (4) 得到一个有效的片段分解，其中每个片段的大小至多为 t。

接下来，我们证明片段数至多为 $\max(1,6(j-i)/t-3)$（如果 $F[i\,..\,j)$是一棵树），否则至多为 $\max(1,6(j-i)/t-1)$。当 $j\le i+t$时，这显然成立，此时算法 4 在第 1 行或第 2行终止。如果 $F[i\,..\,j)$是一棵大小为 $j-i>t$的树，那么算法 4 在第 10行终止。我们考虑几种子情况：

1. 如果 $m-i'\le\frac{2t}{3}$且 $j'-m\le\frac{t}{3}$，那么 $|\mathcal{D}(i,j)|\le3<6(j-i)/t-3$，因为 $j-i>t$。2. 如果 $m-i'\le\frac{2t}{3}$且 $j'-m>\frac{t}{3}$，那么 $(m-i)+(j-j')>t$，因为第 6 行的测试未通过。因此，$j'-m<j-i-t$且 $|\mathcal{D}(i,j)|\le2+|\mathcal{D}(m,j')|\le2+6(j'-m)/t-1<6(j-i-t)/t+1<6(j-i)/t-3$。3. 如果 $m-i'>\frac{2t}{3}$且 $j'-m=0$，那么 $(i'+1-i)+(j-j'+1)>t$，因为第 9 行的测试未通过。因此，$j'-i'\le j-i-t+1$且 $|\mathcal{D}(i,j)|\le1+|\mathcal{D}(i',j')|\le1+6(j'-i')/t-3\le6(j-i-t+1)/t-2<6(j-i)/t-3$。4. 如果 $m-i'>\frac{2t}{3}$且 $0<j'-m\le\frac{t}{3}$，那么 $(i'-i)+(j-m)>t$，因为第 7 行的测试未通过。因此，$m-i'<j-i-t$且 $|\mathcal{D}(i,j)|\le2+|\mathcal{D}(i',m)|\le2+6(m-i')/t-3<6(j-i-t)/t-1<6(j-i)/t-3$。5. 如果 $m-i'>\frac{2t}{3}$且 $j'-m>\frac{t}{3}$，那么 $|\mathcal{D}(i,j)|\le1+|\mathcal{D}(i',m)|+|\mathcal{D}(m,j')|\le1+6(m-i')/t-3+6(j'-m)/t-1\le6(j-i)/t-3$。

如果 $F[i\,..\,j)$不是一棵树，则算法 4 在第 8 行终止。我们考虑以下几种子情况：

1. 若 $m-i'\le\frac{2t}{3}$和 $j'-m\le\frac{t}{3}$，则 $|\mathcal{D}(i,j)|\le4<6(j-i)/t-1$，因为 $j-i>t$。2. 若 $m-i'\le\frac{2t}{3}$和 $j'-m>\frac{t}{3}$，则 $(m-i)+(j-j')>t$，因为第 6 行的测试未通过。因此，$j'-m<j-i-t$和 $|\mathcal{D}(i,j)|\le3+|\mathcal{D}(m,j')|\le3+6(j'-m)/t-1<6(j-i-t)/t+2<6(j-i)/t-1$。3. 若 $m-i'>\frac{2t}{3}$和 $j'-m=0$，则 $|\mathcal{D}(i,j)|\le2+|\mathcal{D}(i',j')|\le2+6(j'-i')/t-3\le6(j-i)/t-1$。4. 若 $m-i'>\frac{2t}{3}$和 $0<j'-m\le\frac{t}{3}$，则 $(i'-i)+(j-m)>t$，因为第 7 行的测试未通过。因此，$m-i'<j-i-t$和 $|\mathcal{D}(i,j)|\le3+|\mathcal{D}(i',m)|\le3+6(m-i')/t-3<6(j-i-t)/t<6(j-i)/t-1$。5. 若 $m-i'>\frac{2t}{3}$和 $j'-m>\frac{t}{3}$，则 $|\mathcal{D}(i,j)|\le2+|\mathcal{D}(i',m)|+|\mathcal{D}(m,j')|\le2+6(m-i')/t-3+6(j'-m)/t-1\le6(j-i)/t-2<6(j-i)/t-1$。

接下来，我们需要为算法提供一个线性时间实现。我们假设，表示同一节点的左括号和右括号之间存在双向指针。这类指针可以通过对输入森林进行线性时间的基于栈的预处理来构造 F。每次 **while** 循环迭代都会增加 $j - j' + i' - i$ (最后一次除外)，因此单次调用 $\mathcal{D}(i, j)$ 函数的代价为 $\mathcal{O}(t)$，这是由不变式 (b)所保证的。调用总次数为$\mathcal{O}(|\mathcal{D}(0, |F|)|) = \mathcal{O}(\frac{1}{t} \cdot |F|)$，因此包括预处理在内的总运行时间为 $\mathcal{O}(|F|)$。 □

引理 3.16。给定总规模为 n 的森林 F 和 G、F 的片段分解 D，以及一个整数 $s \in \mathbb{Z}_+$，可以在 $\mathcal{O}(n + |\mathsf{D}|s^3)$ 时间内找到一个最大规模集合 $S \subseteq \mathsf{D} \times \mathcal{P}(G)$；对于某个宽度至多为 s 的对齐 $\mathcal{A} \in \mathsf{TA}(F, G)$，该集合只包含完美匹配的片段配对。

算法 5：计算 $\mathcal{S}(D_{i,\ j}, G[i' \mathinner{..} j'))$ 的一个最大规模元素。

```
配对(D_{i, j}, G[i' .. j')):
    S ← ∅;
    如果 i' < min(j', i + s)，则 S ←(最大值) 配对(D_{i, j}, G[i' + 1 .. j'));
    如果 j' > max(i', j − s)，则 S ←(最大值) 配对(D_{i, j}, G[i' .. j' − 1));
    如果 D_{i, j} = {F[i .. j)} 且 F[i .. j) = G[i' .. j') 则
        S ←(max) {(F[i .. j) G[i' .. j'))};
    如果 D_{i, j} = D_{i, m} ∪ D_{m, j} 对于某个 m ∈ (i .. j) 则
        对每个 m' ∈ [m − s .. m + s] ∩ [i' .. j'] 执行
            S ←(最大) 配对(D_{i, m}, G[i' .. m')) ∪ 配对(D_{m, j}, G[m' .. j'));
    如果 D_{i, j} = {⟨F[i .. i + ℓ); F[j − r .. j)⟩} ∪ D_{i+ℓ, j−r} 则
        如果 F[i .. i + ℓ) = G[i' .. i' + ℓ) 和 F[j − r .. j) = G[j' − r .. j') 以及 G[i' + ℓ .. j' − r)
          是平衡的，那么
            S ←(max) {(⟨F[i .. i + ℓ); F[j − r .. j)⟩, ⟨G[i' .. i' + ℓ); G[j' − r .. j')⟩)} ∪
              配对(D_{i+ℓ,j−r}, G[i' + ℓ .. j' − r))
        S ←(最大值) 配对(D_{i+ℓ,j−r}, G[最大值(i + ℓ − s, i') .. 最小值(j − r + s, j')));
    返回 S;
```

证明。对于平衡片段 $F[i \mathinner{..} j)$ 的片段分解 $\mathsf{D}_{i,j}$ 和片段 $G[i' \mathinner{..} j')$，令 $\mathcal{S}(D_{i,j}, G[i' \mathinner{..} j'))$ 表示 $\mathsf{D}_{i,\ j} \times \mathcal{P}(G[i' \mathinner{..} j'))$ 的所有子集组成的族：对于某个宽度至多为 s 的对齐$\mathcal{A} \in \mathsf{A}(F[i \mathinner{..} j)\ G[i' \mathinner{..} j'))$，这些子集仅包含彼此完美匹配的片段配对。算法 5 实现了递归过程 $\mathsf{Pairs}(\mathsf{D}_{i,j}, G[i' \mathinner{..} j'))$，在假设 $i' \in [i - s \mathinner{..} i + s]$ 和 $j' \in [j - s \mathinner{..} j + s]$ 的情况下，计算 $\mathcal{S}(D_{i,j}, G[i' \mathinner{..} j'))$ 中大小最大的元素。它使用一个 $S \xleftarrow{\max} S'$ 运算符：当 $|S'| > |S|$ 时将 $S \leftarrow S'$ 赋值为相应值。该算法返回以下候选项中最大的一个：

1. $\emptyset$。由于每个宽度至多为 s 的对齐 $\mathcal{A} \in \mathsf{A}(F[i \mathinner{..} j)\ G[i' \mathinner{..} j'))$ 都见证 $\emptyset \in \mathcal{S}(D_{i,j}, G[i' \mathinner{..} j'))$，因此这显然有效。2. 如果 $i' < \min(j', i+s)$，则 $\mathsf{Pairs}(D_{i,j}, G[i'+1 \mathinner{..} j'))$。设 $S = \mathsf{Pairs}(D_{i,j}, G[i'+1 \mathinner{..} j'))$，并具有一个见证对齐 $\mathcal{A}' \in \mathsf{A}(F[i \mathinner{..} j)\ G[i'+1 \mathinner{..} j'))$。通过在 $\mathcal{A}'$ 前置 (i, i') 得到的对齐对应于插入 $Y[i']$，并见证 $S \in \mathcal{S}(D_{i,j}, G[i' \mathinner{..} j'))$。3. 如果 $j' > \max(i', j - s)$，则 $\mathsf{Pairs}(D_{i,j}, G[i' \mathinner{..} j' - 1))$。设 $S = \mathsf{Pairs}(D_{i,j}, G[i' \mathinner{..} j' - 1))$，并具有一个见证对齐 $\mathcal{A}' \in \mathsf{A}(F[i \mathinner{..} j)\ G[i' \mathinner{..} j' - 1))$。通过在 $\mathcal{A}'$ 后置 (j, j') 得到的对齐对应于插入 $Y[j' - 1]$，并见证 $S \in \mathcal{S}(D_{i,j}, G[i' \mathinner{..} j'))$。

4. $\{(F[i\,..\,j)\ G[i'\,..\,j'))\}$ 如果 $\mathsf{D}_{i,\ j} = \{F[i\,..\,j)\}$ 且 $F[i\,..\,j) = G[i'\,..\,j')$。对齐 $(i+t, j+t)_{t=0}^{i'-i} \in \mathsf{A}(F[i\,..\,j)\ G[i'\,..\,j'))$ 见证 $\{(F[i\,..\,j)\ G[i'\,..\,j'))\} \in \mathcal{S}(D_{i,\ j}, G[i'\,..\,j'))$。

5. 配对$(D_{i,m}, G[i'\,..\,m')) \cup$配对$(D_{m,j}, G[m'\,..\,j'))$，如果 $\mathsf{D}_{i,j} = \mathsf{D}_{i,m} \cup \mathsf{D}_{m,j}$，其中某个 $m \in (i\,..\,j)$ 和 $m' \in [m-s\,..\,m+s] \cap [i'\,..\,j']$。将 $S_L =$配对$(D_{i,m}, G[i'\,..\,m'))$ 和 $S_R =$配对$(D_{m,j}, G[m'\,..\,j'))$ 分别记作，其见证对齐为 $\mathcal{A}_L \in \mathsf{A}(F[i\,..\,m)\ G[i'\,..\,m'))$ 和 $\mathcal{A}_R \in \mathsf{A}(F[m\,..\,j)\ G[m'\,..\,j'))$。在公共端点 (m, m') 处拼接 $\mathcal{A}_L$ 和 $\mathcal{A}_R$，即可得到一个见证对齐 $S_L \cup S_R \in \mathcal{S}(D_{i,j}, G[i'\,..\,j'))$。6. $\{(\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle, \langle G[i'\,..\,i'+\ell);\ G[j'-r\,..\,j')\rangle)\} \cup$配对$(D_{i+\ell,j-r}, G[i'+\ell\,..\,j'-r))$，如果 $\mathsf{D}_{i,\ j} = \{\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle\} \cup \mathsf{D}_{i+\ell,j-r}$ 且 $\langle G[i'\,..\,i'+\ell);\ G[j'-r\,..\,j')\rangle$ 是 G 中的上下文，匹配 $\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle$。考虑集合 $S' =$配对$(D_{i+\ell,j-r}, G[i'+\ell\,..\,j'-r))$ 和见证对齐 $\mathcal{A}' \in \mathsf{A}(F[i+\ell\,..\,j-r)\ G[i'+\ell\,..\,j'-r))$。在公共端点处拼接 $(i+t, i'+t)_{t=0}^{\ell}$, $\mathcal{A}'$ 和 $(j+t, j'+t)_{t=-r}^{0}$，即可得到一个见证对齐 $S' \cup \{(\langle F[i\,..\,i+\ell);F[j-r\,..\,j)\rangle, \langle G[i'\,..\,i'+\ell);\ G[j'-r\,..\,j')\rangle)\} \in \mathcal{S}(D_{i,j}, G[i'\,..\,j'))$。

7. 配对$(D_{i+\ell,j-r}, G[\max(i+\ell-s, i')\,..\,\min(j-r+s, j')))$ 若 $\mathsf{D}_{i,j} = \{\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle\} \cup \mathsf{D}_{i+\ell,\ j-r}$. 令 $S =$配对$(D_{i+\ell,j-r}, G[i'+\ell'\,..\,j'-r'))$, 其中 $\ell' = \max(0, i+\ell-s-i')$ 且$r' = \max(0, j'-j+r-s)$，并有见证对齐 $\mathcal{A}' \in \mathsf{A}(F[i+\ell, j-r)\ G[i'+\ell', j'-r'))$. 拼接 $(i+t, i'+t)_{t=0}^{\ell'}$, $(i+t, i'+\ell')_{t=\ell'}^{\ell}$, $\mathcal{A}'$, $(j+t, j'-r')_{t=-r'}^{-r}$ 和 $(j+t, j'+t)_{t=-r'}^{0}$，得到一个见证 $S \in \mathcal{S}(D_{i,j}, G[i'\,..\,j'))$ 的对齐。

接下来，考虑一个最大规模元素 $S \in \mathcal{S}(D_{i,\ j}, G[i'\,..\,j'))$，以及一个宽度至多为 s 的见证对齐 $\mathcal{A} \in \mathsf{A}(F[i\,..\,j)\ G[i'\,..\,j'))$。

(a) 如果 $\mathsf{D}_{i,j} = \emptyset$，则必有 $S = \emptyset$，这由候选项 1 覆盖。(b) 假设 $\mathsf{D}_{i,j} = \{F[i\,..\,j)\}$。$S = \emptyset$ 的情形由候选项 1 覆盖。否则，$S = \{(F[i\,..\,j)\ G[i''\,..\,j'')\}$，其中 $i'' \in [i'\,..\,i+s]$ 和 $j'' \in [j-s\,..\,j']$。这由若干次应用候选项 2、若干次应用候选项 3，最后再应用一次候选项 4 覆盖。(c) 假设对于某个 $m \in (i\,..\,j)$，有 $\mathsf{D}_{i,j} = \mathsf{D}_{i,m} \cup \mathsf{D}_{m,j}$。由于 $\mathcal{A}$ 的宽度不超过 s，必有 $(m, m') \in \mathcal{A}$，其中 $m' \in [m-s\,..\,m+s] \cap [i'\,..\,j']$。因此，$S$ 可以表示为 $\mathcal{S}(\mathsf{D}_{i,m}, G[i'\,..\,m'))$ 中的一个元素与 $\mathcal{S}(D_{m,j}, G[m'\,..\,j'))$ 中的一个元素之并。故此情形由候选项 5 覆盖。(d) 假设 $\mathsf{D}_{i,j} = \{\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle\} \cup \mathsf{D}_{i+\ell,j-r}$。如果 S 不含任何形如 $(\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle, \langle G[i''\,..\,i''+\ell);\ G[j''-r\,..\,j'')\rangle)$ 的数对，则 $S \in \mathcal{S}(D_{i+\ell,j-r}, G[\max(i+\ell-s, i')\,..\,\min(j-r+s, j')))$，此情形由候选项 7 覆盖。否则，必有 $i'' \in [\max(i', i-s)\,..\,i+s]$ 且 $j'' \in [j-s\,..\,\min(j', j+s)]$。这是通过若干次应用候选项 2、若干次应用候选项 3，最后再应用一次候选项 4 覆盖的，因为 $S \setminus \{(\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle, \langle G[i'\,..\,i'+\ell);\ G[j'-r\,..\,j')\rangle)\} \in \mathcal{S}(D_{i+\ell,j-r}, G[i'+\ell\,..\,j'-r))$。

至此，我们完成了对算法 5 正确性的证明。所求集合 S 通过调用配对$(\mathsf{D}\ G[0\,..\,|G|))$ 得到，只要 $s \geq \big||F|-|G|\big|$即有效。否则，不存在宽度至多为 $\mathcal{A} \in \mathsf{TA}(F,\ G)$ 的对齐，因此我们返回 $S = \emptyset$。

至于高效实现，我们使用记忆化，确保每次调用配对最多执行一次。调用次数为 $\mathcal{O}(|\mathsf{D}| \cdot s^2)$，每次调用执行 $\mathcal{O}(s$条指令。为了在 $\mathcal{O}(1)$时间内实现每条指令，我们将集合实现为附带大小信息的持久化链表（这是可行的，因为每次并集操作的参数都保证互不相交）。此外，我们使用定理2.1（用于检查 F的片段是否与 G的片段匹配）以及事实3.1（用于检查 G 的片段是否平衡）。计入必要的预处理后，总运行时间为 $\mathcal{O}(n + |\mathsf{D}|s^3)$。

□

引理 3.17. 存在线性时间算法，对于给定的森林 P 和整数 $k \in \mathbb{Z}_+$，可以构造一个长度至多为 $74k^3$ 的森林，该森林对于每个归一化拟度量 w 都与 P $\mathsf{ted}^w_{\le k}$-等价。

算法 6：构造一个长度至多为 $74k^3$ 且与 P $\mathsf{ted}^w_{\le k}$-等价的森林。

水平归约(P, k):
  $P' \leftarrow$ 周期性约简$(P, 4k, \{Q \in \Sigma^+ : |Q| \le 4k$ 且 Q 是一个本原森林$\})$;
  如果 $|P'| \ge 74k^3$，则返回某个 $a \in \Sigma$ 对应的 $(a^{37k^3})^{37k^3}_a$;
  否则返回 P';

证明。我们应用引理2.13，取 $e = 4k$ 和 $\mathcal{Q}$，其中 $\mathcal{Q}$ 由所有长度不超过 $4k$的本原森林组成。对于任意标签 $P'' := (a^{37k^3})^{37k^3}_a$，我们返回$= (a^{37k^3})^{37k^3}_a$；否则，根据$|P'| \le 74k^3$ 是否成立，返回 P'。

注意，$\mathcal{Q}$的选取使得 P'避免水平 k-周期性，并且根据引理3.5，P'与$\mathsf{ted}^w_{\le k}$-等价（对每个归一化拟度量）。P w $|P'| < 74k^3$ 因此，当时，算法是正确的。3 7 否则，引理。P' P'' 蕴含与 $\mathsf{ted}^w_{\le k}$-等价（对每个归一化拟度量） w（二者均避免水平 k-周期性，且长度至少为 $74k^3$）。

可使用事实3.1实现预言机：它可以在常数时间内测试 P 的给定片段是否属于 $\mathcal{Q}$。因此，根据引理2.13，总运行时间为线性。 □

引理 3.18. 存在一个线性时间算法，给定上下文 P 和一个整数 $k \in \mathbb{Z}_+$，可构造一个长度至多为 $578k^4$ 的上下文；对于每个归一化拟度量 w，该上下文都与 P $\mathsf{ted}^w_{\le k}$-等价。

算法 7：构造一个长度至多为 $578k^4$ 、与 P $\mathsf{ted}^w_{\le k}$-等价的上下文。

垂直归约(P, k):
  令 $P = P_0 \star \cdots \star P_{e-1}$，其中每个 P_i 都是深度为 1 的上下文;
  对于 $i \leftarrow 0$ 到 e，执行
    令 $P_i = \langle (_{a_i} F_i;\ G_i)_{a_i} \rangle$;
    $\mathsf{P} \leftarrow \mathsf{P} \cdot \langle (_{a_i} \cdot$ 水平归约$(F_i, k);$ 水平归约$(G_i, k)\ \cdot)_{a_i} \rangle$;
  $\mathcal{Q} \leftarrow \{\mathsf{Q} : \bigstar_{i=0}^{|\mathsf{Q}|-1} \mathsf{Q}[i]$ 是长度至多为 $8k\}$ 的本原上下文;
  $\mathsf{P}' \leftarrow$ 周期性约简$(\mathsf{P}\ 6k, \mathcal{Q})$;
  $P' \leftarrow \bigstar_{i=0}^{|\mathsf{P}'|-1} \mathsf{P}'[i]$;
  如果 $|P'| \ge 578k^4$ 那么返回 $\bigstar_{i=0}^{17k^2-1} \langle (_a((_a)_a)^i;\ ((_a)_a)^{17k^2-1-i})_a \rangle$ 对于某个 $a \in \Sigma$;
  否则返回 P';

证明。令 $P = P_0 \star \cdots \star P_{e-1}$，其中每个 P_i 都是深度为 1 的上下文，即 $P_i = \langle (_{a_i} F_i;\ G_i)_{a_i} \rangle$，其中标签为 $a_i \in \Sigma$，森林为 F_i、G_i。首先，我们的算法构造字符串 $\mathsf{P}[0\,..\,e)$，其字符是深度为 1 的上下文，并定义为：$\mathsf{P}[i] = \langle (_{a_i} F'_i;\ G'_i)_{a_i} \rangle$，其中森林 $F'_i =$ HorizontalReduction(F_i, k) 和 $G'_i =$ HorizontalReduction(G_i, k) 使用引理 3.17 构造。接下来，我们使用引理 2.13 转换 P，其中 $e = 6k$，并定义一族 $\mathcal{Q}$，使得 $\mathsf{P}[i\,..\,j) \in \mathcal{Q}$ 当且仅当 $\mathsf{P}[i] \star \cdots \star \mathsf{P}[j-1]$ 是长度至多为 $8k$ 的本原上下文（这意味着 $j - i \le 4k$）。为了将引理 2.13 应用于 P，我们使用线性时间字符串

排序 [PT87、AN94] 将 $\mathbf{P}$（深度为 1 的上下文）的字符映射为整数标识符。将对应于所得字符串 $\mathbf{P}'$ 的上下文进行复合，我们得到上下文 。我们返回 P''： $\bigstar_{i=0}^{17k^2-1}\langle\ a\ \ a\ a^{\,i}\ \ a\ a^{\,17k^2-1-i}\ a\rangle$ $a\in\Sigma$ P' (((); ()))（任意标签均可）或取决于是否$|P'|\geq 578k^4$ 成立。

注意， $|P''|=\sum_{i=0}^{17k^2-1}(1+2\cdot i+2\cdot(17k^2-1-i)+1)=17k^2\cdot 2\cdot 17k^2=578k^4$。因此，得到的上下文（ P'或 P''）的长度至多为 $578k^4$。现在来说明，对于每个归一化拟度量 w，它都与 P 在 $\mathsf{ted}^w_{\leq k}$意义下等价。根据引理 3.17，森林 F'_i和 G'_i 分别与 F_i 和 G_i在 $\mathsf{ted}^w_{\leq k}$意义下等价，因此 $\bigstar_{i=0}^{e-1}\mathbf{P}[i]$ 与 P在 $\mathsf{ted}^w_{\leq k}$意义下等价。根据引理 2.13，上下文 P'可由 $\bigstar_{i=0}^{e-1}\mathbf{P}[i]$ 反复将 Q^{6k+1} 替换为 Q^{6k}得到，其中 Q 是长度至多为 $8k$的本原上下文。根据引理 3.10， Q^{6k+1} 与Q^{6k}在 $\mathsf{ted}^w_{\leq k}$意义下等价，因此该操作保留 $\mathsf{ted}^w_{\leq k}$等价关系，也就是说， P' 也与 P在 $\mathsf{ted}^w_{\leq k}$意义下等价。此外，$\mathbf{P}'$ 中的每个深度为 1 的上下文都源自 $\mathbf{P}$，因此 P' （任意一半）中出现的每个森林长度都至多为 $74k^3$。此外，引理 2.13 保证，对于任何长度至多为 $8k$的上下文 Q， P'都不是 $C\star Q^{6k+1}\star D$的形式，因此 P' 不存在垂直 k周期性。根据构造，P'' 不存在垂直 k周期性，且其两半仅包含长度至多为 $74k^3$ 的森林（实际上，至多为 $34k^2$）。因此，只要 $|P''|\geq 578k^4$，引理 3.12 就意味着 P''与 P' 在 $\mathsf{ted}^w_{\leq k}$意义下等价（根据传递性，也与 P等价）。

关于运行时间分析，我们注意到引理 3.17 的所有应用都涉及 P中互不相交的片段，因此调用水平归约 的总代价是线性的。为上下文 $\mathbf{P}[i]$ 分配整数标识符并应用引理 2.13 也只需线性时间。最后，仅当 P'' 时才构造 $|P'|\geq 578k^4$，因此这一步的代价也可由 $\mathcal{O}(k^4)=\mathcal{O}(|P|)$ 上界。 □

定理 3.19. 存在一个运行时间为 $\mathcal{O}(n)$ 的算法，它给定大小至多为 $n\geq 12716k^5$ 的森林 F, G 和一个整数 $k\in\mathbb{Z}_+$，构造长度至多为 $\frac{n}{2}+6358k^5$ 的森林 F', G'，使得对于每个归一化拟度量 w，$\mathsf{ted}^w_{\leq k}(F,G)=\mathsf{ted}^w_{\leq k}(F',G')$ 成立。

证明。根据对称性，不失一般性，假设 $|F|\geq|G|$。我们首先应用引理 3.15，构造 F 的一个片段分解 $\mathbf{D}$，其中至多包含 $12k-1$ 个片段，且每个片段的长度至多为 $\lceil\frac{n}{2k}\rceil$。接下来，我们使用引理 3.16，找出一个最大规模集合 $S\subseteq\mathbf{D}\times\mathcal{P}(G)$：对于某个宽度至多为 $2k$ 的对齐 $\mathcal{A}\in\mathsf{A}(F,G)$，该集合仅包含完美匹配的片段配对 $\mathcal{A}$。如果 $|S|<|\mathbf{D}|-k$，我们返回 $F'=((a)a)^{k+1}$ 和某个 $a\in\Sigma$ 对应的 $G'=\varepsilon$。否则，对于 S 中的每一对匹配森林 $F[i\,..\,j)=P=G[i'\,..\,j')$，我们使用引理 3.17 构造一个长度至多为 $74k^3$ 的森林 P'，使其对于每个归一化拟度量 w 都与 P $\mathsf{ted}^w_{\leq k}$-等价。我们将 $F[i\,..\,j)$ 和 $G[i'\,..\,j')$ 处的 P 出现替换为 P' 的出现。类似地，对于 S 中的每一对匹配上下文 $\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle=P=\langle G[i'\,..\,i'+\ell);\ G[j'-r\,..\,j')\rangle$，我们使用引理 3.18 构造一个长度至多为 $578k^4$ 的上下文 P'，使其对于每个归一化拟度量 w 都与 P $\mathsf{ted}^w_{\leq k}$-等价。我们将 $\langle F[i\,..\,i+\ell);\ F[j-r\,..\,j)\rangle$ 和 $\langle G[i'\,..\,i'+\ell);\ G[j'-r\,..\,j')\rangle$ 处的 P 出现替换为 P' 的出现。

如果对于任意归一化权重函数 w，$\mathsf{ted}^w(F,G)\leq k$ 成立，那么相应最优对齐的无权代价不超过 k。因此，其宽度至多为 $2k$，并且与 $\mathbf{D}$ 的至多 k 个片段之外的所有片段都完美匹配。因此，在这种情况下， $|S|\geq|\mathbf{D}|-k$。特别地，如果对于某个 $a\in\Sigma$，我们返回 $F'=((a)a)^{k+1}$和 $G'=\varepsilon$，那么 $\mathsf{ted}^w_{\leq k}(F,G)=\infty=\mathsf{ted}^w_{\leq k}(F',G')$ 成立，正如所声称的那样。在这种情况下， $|F'|$, $|G'|\leq 2k+2\leq 4k<6358k^5$。否则，根据 $\mathsf{ted}^w_{\leq k}$-等价关系的定义，所得森林 F'和 G'满足 $\mathsf{ted}^w_{\leq k}(F,G)=\mathsf{ted}^w_{\leq k}(F',G')$。此外，$|F'|\leq k\cdot\lceil\frac{n}{2k}\rceil+(11k-1)\cdot 578k^4\leq\frac{n}{2}+k+(11k-1)\cdot 578k^4\leq\frac{n}{2}+11\cdot 578\cdot k^5=\frac{n}{2}+6358k^5$和$|G'|=|F'|+|G|-|F|\leq|F'|\leq\frac{n}{2}+6358k^5$。

引理3.15 和3.16 的代价 $\mathcal{O}(n)$ 和 $\mathcal{O}(n+k^4)=\mathcal{O}(n)$ 时间。对水平归约和垂直归约的调用涉及 F中互不相交的片段，因此它们的

根据引理 $\mathcal{O}(n)$，总代价分别为引理 3.17 和 3.18所给出的结果。 □

推论 3.20。存在一个线性时间算法，给定森林 F, G 和一个整数$k \in \mathbb{Z}_+$，可构造森林 F', G'，其长度至多为 $12717k^5$，使得对于每个归一化拟度量 w，均成立 $\mathsf{ted}^w_{\le k}(F,G) = \mathsf{ted}^w_{\le k}(F',G')$。

证明。只要 $\max(|F|,|G|) > 12717k^5$，我们就反复应用定理 3.19，并返回所得的森林对。形式化地，我们构造一个序列 $(F_i,G_i)_{i=0}^t$，使得对每个 $i \in [0\,..\,t)$，$(F_0,G_0) = (F,G)$ 且 $\mathsf{ted}^w_{\le k}(F_{i+1},G_{i+1}) = \mathsf{ted}^w_{\le k}(F_i,G_i)$ 均成立。考虑第 i 次迭代中的森林 (F_i,G_i)。如果 $n_i := \max(|F_i|,|G_i|) \le 12717k^5$，我们令 $t := i$，并返回 $(F',G') := (F_i,G_i)$。否则，我们应用定理 3.19，导出长度至多为 $n_{i+1} := \max(|F_{i+1}|,|G_{i+1}|) \le \frac{1}{2}n_i + 6358k^5$ 的森林 F_{i+1} 和 G_{i+1}，使得 $\mathsf{ted}^w_{\le k}(F_{i+1},G_{i+1}) = \mathsf{ted}^w_{\le k}(F_i,G_i)$。由于 $n_{i+1} - 12716k^5 \le \frac{1}{2}(n_i - 12716k^5)$，值 n_i 在每次迭代中都严格减小，因此该过程会终止。此外，每次迭代的运行时间为 $\mathcal{O}(n_i) = \mathcal{O}(n_i - 12716k^5)$。后面的这些值构成一个几何级数，其总和由 $i = 0$ 处的首项主导。因此，总运行时间与输入规模呈线性关系。 □

定理 1.2. 给定长度至多为 n 的森林 F, G、一个整数 $k \in \mathbb{Z}_+$ 和一个拟度量 w，可以在 $\mathcal{O}(n + k^{15})$ 时间内计算 $\mathsf{ted}^w_{\le k}(F,G)$。此外，可以在 $\mathcal{O}(n + k^7 \log k)$ 时间内计算 $\mathsf{ted}_{\le k}(F,G)$。

证明。我们首先应用推论 3.20，构造长度为 $\mathcal{O}(k^5)$ 的森林 F', G'，使得 $\mathsf{ted}^w_{\le k}(F,G) = \mathsf{ted}^w_{\le k}(F', G')$。然后，我们使用 Demaine、Mozes、Rossman 和 Weimann 的算法 [DMRW10] 计算 $\mathsf{ted}^w_{\le k}(F',G')$。这两个步骤的运行时间分别为 $\mathcal{O}(n)$ 和 $\mathcal{O}((k^5)^3)$，总计为 $\mathcal{O}(n + k^{15})$。如果 w 是离散度量（无权情形），则使用 Akmal 和 Jin 的算法 [AJ21]，计算 $\mathsf{ted}_{\le k}(F',G')$，其代价为 $\mathcal{O}(k^5 \cdot k^2 \cdot \log(k^5)) = \mathcal{O}(k^7 \log k)$ 时间。 □

4 Dyck 编辑距离

本节给出一种确定性算法，用于计算给定输入字符串的加权 Dyck 编辑距离。形式化地，我们证明如下定理。

定理 1.3. 给定一个长度为 n 的字符串 X、一个整数 $k \in \mathbb{Z}_+$ 和一个偏斜度量 w，值 $\mathsf{dyck}^w_{\le k}(X)$ 可在 $\mathcal{O}(n + k^{12})$ 时间内计算得出。

4.1 预备知识

在 Dyck 语言中，字母表 Σ 由两个互不相交的集合 T 和 $\overline{T}$ 组成，分别表示左括号和右括号，并存在一个双射 $f\colon T \to \overline{T}$，将每个左括号映射到对应的右括号。我们将此映射扩展为一个对合 $f\colon T \cup \overline{T} \to T \cup \overline{T}$，再扩展为一个对合 $f\colon \Sigma^* \to \Sigma^*$，将每个字符串 $X[0]X[1]\cdots X[|X|-1]$ 映射到其反向互补 $\overline{X[|X|-1]}\cdots\overline{X[1]}\,\overline{X[0]}$。给定两个字符串 X、Y，我们将它们的串接记作 XY 或 $X \cdot Y$。

Dyck 语言 $\mathrm{Dyck}(\Sigma) \subseteq \Sigma^*$ 由所有基于 Σ 的括号匹配表达式组成；形式上，它可以通过一个上下文无关文法来定义，该文法只有一个非终结符 S，并包含产生式$S \to SS$、$S \to \varnothing$（空串）以及对所有 $a \in T$ 均成立的 $S \to aS\overline{a}$。

定义 4.1（高度）。给定一个字母表集合 Σ，定义函数 $h\colon \Sigma \to \{-1,1\}$ where$h(a) = 1$ if $a \in \Sigma$ 是左括号，且 $h(a) = -1$ otherwise。给定一个字符串 $X \in \Sigma^n$，将位置 i（其中 $0 \le i \le n$）的高度定义为 $H(i) = \sum_{j=0}^{i-1} h(X[j])$。

这里的 $H(i)$ 是左括号数量与右括号数量之差，其中 $X[0..i)$ 。

定义 4.2（峰与谷）。给定字符串 $X \in \Sigma$，若索引 $i \in [1 \dots n)$ 满足 $H(i-1) < H(i) > H(i+1)$，则称其为峰；若满足 $H(i-1) > H(i) < H(i+1)$，则称其为谷。

4.2 Dyck语言对齐与加权Dyck编辑距离

若 $\mathcal{M} \subseteq \{(i,j) \subseteq \mathbb{Z}^2 : i < j\}$ 是非交叉匹配，则称其为非交叉匹配，当且仅当任意两个不同的配对 (i,j)、$(i',j') \in \mathcal{M}$ 满足 $i < j < i' < j'$ 或 $i < i' < j' < j$。这种匹配也可以解释为一个函数 $\mathcal{M}: \mathbb{Z} \to \mathbb{Z} \cup \{\perp\}$，其中若存在某个 $j \in \mathbb{Z}$，使得 $(i,j) \in \mathcal{M}$ 或 $(j,i) \in \mathcal{M}$，则 $\mathcal{M}(i) = j$；否则，$\mathcal{M}(i) = \perp$。对于字符串 $X \in \Sigma^*$，我们将其 Dyck 语言对齐定义为上述定义的匹配函数 $\mathcal{M}$ 。

对于 X 的两个片段 $X[p\,..\,q)$ 和 $X[p'\,..\,q')$，如果对每个 $r \in [p\,..\,q)$ 都满足 $\overline{X[p\,..\,q)} = X[p'\,..\,q') \in T^*$ and $(r, q'-r-1)$，我们记作 $X[p\,..\,q) \simeq_{\mathcal{M}} X[p'\,..\,q')$。

类似于第 2 节 2.2，我们在 $\Sigma := \Sigma \cup \{\varepsilon\}$ 上定义权重函数 w。如果该权重函数满足三角不等式，即对每个 a、b、$c \in \Sigma$，均有 $w(a,b) + w(b,c) \geq w(a,c)$；并满足反对称性，即对每个 a、$b \in \Sigma$，均有 $w(a,b) = w(b,\overline{a})$，则称该权重函数为偏斜度量。本节其余部分中，除非另有说明，我们均假设权重函数 w 是偏斜度量。

定义 4.3. 字符串 $X \in \Sigma^*$ 关于权重函数 w 的加权 Dyck 编辑距离，是 X 与某个字符串 $Y \in \mathsf{Dyck}(\Sigma)$之间的最小编辑距离 $\mathsf{ed}^w(X,Y)$。形式化地，

$$\mathsf{dyck}^w(X) = \min_{Y \in \mathsf{Dyck}(\Sigma)} \mathsf{ed}^w(X,Y).$$

对于 $k \in \mathbb{R}_{\geq 0}$，我们也记作

$$\mathsf{dyck}^w_{\leq k}(X) = \begin{cases} \mathsf{dyck}^w(X) & \text{if } \mathsf{dyck}^w(X) \leq k, \\ \infty & \text{otherwise.} \end{cases}$$

对齐 $\mathcal{M} \in \mathsf{M}(X)$ 关于权重函数 w的代价，记作 $\mathsf{dyck}^w_{\mathcal{M}}(X)$，定义为

$$\mathsf{dyck}^w_{\mathcal{M}}(X) = \sum_{(i,j) \in \mathcal{M}} \mathsf{dyck}^w(X[i]X[j]) + \sum_{i \in [0..|X|): \mathcal{M}(i) = \perp} \mathsf{dyck}^w(X[i]).$$

事实 4.4. 对于任意 字符串 X 和权重函数 w，我们有 $\mathsf{dyck}^w(X) = \min_{\mathcal{M} \in \mathsf{M}(X)} \mathsf{dyck}^w_{\mathcal{M}}(X)$.

证明。首先，我们对 $|X|$ 进行归纳，证明对于每个 $\mathcal{M} \in \mathsf{M}(X)$，$\mathsf{dyck}^w(X) \leq \mathsf{dyck}^w_{\mathcal{M}}(X)$ 成立。当 $|X| = 0$ 时，断言显然成立。若 $\mathcal{M}(0) = \perp$，则我们构造 $\mathcal{M}' := \{(i-1, j-1) : (i,j) \in \mathcal{M}\}$ 以及 $X' := X[1\,..\,|X|)$。根据归纳假设，$\mathsf{dyck}^w(X) \leq \mathsf{dyck}^w(X') + \mathsf{dyck}^w(X[0]) \leq \mathsf{dyck}^w_{\mathcal{M}'}(X') + \mathsf{dyck}^w(X[0]) = \mathsf{dyck}^w_{\mathcal{M}}(X)$。若 $\mathcal{M}(0) = |X|-1$，则我们构造 $\mathcal{M}' := \{(i-1, j-1) : (i,j) \in \mathcal{M} \setminus \{(0, |X|-1)\}$ 以及 $X' = [1\,..\,|X|-1)$。根据归纳假设，$\mathsf{dyck}^w(X) \leq \mathsf{dyck}^w(X') + \mathsf{dyck}^w(X[0]X[|X|-1]) \leq \mathsf{dyck}^w_{\mathcal{M}'}(X') + \mathsf{dyck}^w(X[0]X[|X|-1]) = \mathsf{dyck}^w_{\mathcal{M}}(X)$。否则，对于某个 $p \in [1\,..\,|X|-1)$，有 $(0,p) \in M$。在这种情况下，我们构造 $\mathcal{M}' := \{(i,j) \in \mathcal{M} : j \leq p\}$ 和 $X' := X[0\,..\,p]$，并构造 $\mathcal{M}'' := \{(i-p-1, j-p-1) : (i,j) \in \mathcal{M}$ 以及 $i > p\}$ 和 $X'' := X[p+1\,..\,|X|)$。根据归纳假设，$\mathsf{dyck}^w(X) \leq \mathsf{dyck}^w(X') + \mathsf{dyck}^w(X'') \leq \mathsf{dyck}^w_{\mathcal{M}'}(X') + \mathsf{dyck}^w_{\mathcal{M}''}(X'') = \mathsf{dyck}^w_{\mathcal{M}}(X)$；这里，最后一个等式是由以下事实得出的：$|\mathcal{M}| = |\mathcal{M}'| + |\mathcal{M}''|$：任意满足 $i \leq p$ 和 $j > p$ 的 $(i, j) \in M$ 都会违反 $\mathcal{M}$ 的非交叉性质。

接下来，我们对 $|X|$ 进行归纳，证明存在 $\mathcal{M} \in \mathrm{M}(X)$ 使得 $\mathrm{dyck}^w_{\mathcal{M}}(X) \leq \mathrm{dyck}^w(X)$；同样，当 $|X| = 0$ 时，断言显然成立。固定 $Y \in \mathrm{Dyck}(\Sigma)$ 和 $\mathcal{A} \in \mathrm{A}(X,Y)$，使得 $\mathrm{dyck}^w(X) = \mathrm{ed}^w_{\mathcal{A}}(X,Y)$。若 $\mathcal{A}$ 删除 $X[0]$，则考虑 $X' := X[1..|X|)$。归纳假设给出一个匹配 $\mathcal{M}'$，使得 $\mathrm{dyck}^w_{\mathcal{M}'}(X') \leq \mathrm{dyck}^w(X')$。在这种情况下，我们令 $\mathcal{M} := \{(i+1, j+1) : (i,j) \in \mathcal{M}'\}$，从而 $\mathrm{dyck}^w_{\mathcal{M}}(X) = \mathrm{dyck}^w(X[0]) + \mathrm{dyck}^w_{\mathcal{M}'}(X') \leq \mathrm{ed}^w(X[0], \varepsilon) + \mathrm{dyck}^w(X') \leq w(X[0], \varepsilon) + \mathrm{ed}^w(X', Y') \leq \mathrm{ed}^w_{\mathcal{A}}(X,Y) = \mathrm{dyck}^w(X)$。$\mathcal{A}$ 删除 $X[|X|-1]$ 的情况是对称的，因此我们可以假设 $\mathcal{A}$ 既不删除 $X[0]$，也不删除 $X[|X|-1]$；特别地，$Y \neq \varepsilon$。

假设对于某些非空字符串 $Y', Y'' \in$，有 $Y = Y' \cdot Y''$ $\mathrm{Dyck}(\Sigma)$。由此得到一个分解 $X = X' \cdot X''$，满足 $\mathrm{ed}^w(X,Y) = \mathrm{ed}^w(X',Y') + \mathrm{ed}^w(X'',Y'')$。此外，$Y$ 的最优性保证了 X' 和 X'' 均非空。根据归纳假设，存在匹配 $\mathcal{M}', \mathcal{M}''$，使得 $\mathrm{dyck}^w_{\mathcal{M}'}(X') \leq \mathrm{dyck}^w(X')$，且 $\mathrm{dyck}^w_{\mathcal{M}'}(X'') \leq \mathrm{dyck}^w(X'')$。在这种情况下，我们设 $\mathcal{M} := \mathcal{M}' \cup \{(i + |X'|, j + |X'|) : (i,j) \in \mathcal{M}''\}$，从而 $\mathrm{dyck}^w_{\mathcal{M}}(X) = \mathrm{dyck}^w_{\mathcal{M}'}(X') + \mathrm{dyck}^w_{\mathcal{M}'}(X'') \leq \mathrm{dyck}^w(X') + \mathrm{dyck}^w(X'') \leq \mathrm{ed}^w(X', Y') + \mathrm{ed}^w(X'', Y'') \leq \mathrm{ed}^w_{\mathcal{A}}(X,Y) = \mathrm{dyck}^w(X)$。

在剩余情况下，必有 $Y = aY'\overline{a}$ for $a \in T$ and $Y' \in \mathrm{Dyck}(\Sigma)$ $\mathcal{A}$。我们先假设 $Y[0] = a$ 与 $X[0]$ 对齐，且 $Y[|Y|-1] = \overline{a}$ 与 $X[|X|-1]$ 对齐。在这种情况下，$\mathcal{A}$ 将 $Y[1..|Y|-1) = Y'$ 与 $X' := X[1..|X|-1)$ 对齐。归纳假设给出一个匹配 $\mathcal{M}'$，使得 $\mathrm{dyck}^w_{\mathcal{M}'}(X') \leq \mathrm{dyck}^w(X')$。在这种情况下，我们设 $\mathcal{M} := \{(0, |X|-1)\} \cup \{(i+1, j+1) : (i,j) \in \mathcal{M}'\}$，从而 $\mathrm{dyck}^w_{\mathcal{M}}(X) = \mathrm{dyck}^w(X[0]X[|X|-1]) + \mathrm{dyck}^w_{\mathcal{M}'}(X') \leq \mathrm{ed}^w(X[0]X[|X|-1], a\overline{a}) + \mathrm{dyck}^w(X') \leq w(X[0], a) + w(X[|X|-1], \overline{a}) + \mathrm{ed}^w(X', Y') \leq \mathrm{ed}^w_{\mathcal{A}}(X,Y) = \mathrm{dyck}^w(X)$。接下来，假设 $\mathcal{A}$ 与 $X[0]$ 对齐，但插入了 $Y[|Y|-1] = \overline{a}$。在这种情况下，$\mathcal{A}$ 将 $Y[1..|Y|-1) = Y'$ 与 $X' := X[1..|X|)$ 对齐。归纳假设给出一个匹配 $\mathcal{M}'$，使得 $\mathrm{dyck}^w_{\mathcal{M}'}(X') \leq \mathrm{dyck}^w(X')$。在这种情况下，我们设 $\mathcal{M} := \{(i+1, j+1) : (i,j) \in \mathcal{M}'\}$，从而 $\mathrm{dyck}^w_{\mathcal{M}}(X) = \mathrm{dyck}^w(X[0]) + \mathrm{dyck}^w_{\mathcal{M}'}(X') \leq \mathrm{ed}^w(X[0], a\overline{a}) + \mathrm{dyck}^w(X') \leq w(X[0], a) + w(\varepsilon, \overline{a}) + \mathrm{ed}^w(X', Y') \leq \mathrm{ed}^w_{\mathcal{A}}(X,Y) = \mathrm{dyck}^w(X)$。$\mathcal{A}$ 插入 $Y[0] = a$ 并与 $Y[|Y|-1] = \overline{a}$ 对齐的情况是对称的。$\mathcal{A}$ 同时插入 $Y[0] = a$ 和 $Y[|Y|-1] = \overline{a}$ 的情况根据 Y 的最优性是不可能的。最后，我们注意到，由于 $\mathcal{A}$ 既不删除 $X[0]$ 也不删除 $X[|X|-1]$，对齐 $\mathcal{A}$ 不可能将 $Y[0]$ 与除 $X[0]$ 之外的任何字符对齐，也不可能将 $Y[|Y|-1]$ 与除 $Y[|Y|-1]$ 之外的任何字符对齐。因此，上述情况分析是完整的。 □

断言 4.5. 对 于任意 $x \in \Sigma$ 和斜度量权重函数 w，$\mathrm{dyck}^w(x) = w(x, \epsilon) = w(\epsilon, \overline{x})$。

证明。我们考虑以下三种不同情形。

情形 1：x 被删除。在这种情形下，$\mathrm{dyck}^w(x) = w(x, \epsilon)$。

情形 2：若 $x \in T$，则在 x 之后插入 $\overline{x}$；若 $x \in \overline{T}$，则在 x 之前插入 $\overline{x}$。在这种情形下，$\mathrm{dyck}^w(x) = w(\epsilon, \overline{x}) = w(x, \epsilon)$。最后一个等式成立是因为 w 是反对称的。因此，插入可以替换为删除。从现在起，我们假设只允许删除和替换。

情形 3：x 被替换为某个 $y \in \Sigma$。这里我们还需要插入 $\overline{y}$。因此，$\mathrm{dyck}^w(x) = w(x,y) + w(\epsilon, \overline{y}) = w(x,y) + w(y, \epsilon) \geq w(x, \epsilon)$。第二个等式成立是因为 w 是反对称的，最后一个不等式成立是因为 w 满足三角不等式。显然，$\mathrm{dyck}^w(x) \leq w(x, \epsilon)$。因此断言得证。 □

命题 4.6. 对于每个 x，$y \in \Sigma$ 以及斜度量权重函数 w，$\mathrm{dyck}^w(xy) = \min^{z \in T \cup \{\epsilon\}} w(x,z) + w(y, \overline{z})$。

证明。设 $z \in T \cup \{\epsilon\}$ 使 $w(x,z) + w(y, \overline{z})$ 最小。容易证明，$\mathrm{dyck}^w(xy) \leq w(x,z) + w(y, \overline{z})$，因为可以分别用 z、$\overline{z}$ 替换 x、y。接下来证明反向结论。根据断言 4.5，我们假设只允许进行删除和替换。

情形 1： x 和 y 均被删除。此时 $\mathsf{dyck}^w(xy) \geq w(x,\epsilon) + w(y,\epsilon)$。由 $\epsilon \in T \cup \{\epsilon\}$ 和 $\overline{\epsilon} = \epsilon$ 可得该断言。

情形 2：用 $\overline{y}$ 替换 x。这里 $\mathsf{dyck}^w(xy) \geq w(x,\overline{y}) = w(x,\overline{y}) + w(y,y)$。因此，断言由 $\overline{y} \in T$ 得证。

情形 3：用 $\overline{x}$ 替换 y。这里 $\mathsf{dyck}^w(xy) \geq w(y,\overline{x}) = w(x,x) + w(y,\overline{x})$。因此，断言由 $x \in T$ 得证。

情形 3：用 z 替换 x，并用 $\overline{z}$ 替换 y。这里 $\mathsf{dyck}^w(xy) \geq w(x,z) + w(y,\overline{z})$。因此，断言由 $z \in T$ 得证。 □

从现在起，我们假设 w 是反对称的。

4.2.1 预处理。

给定输入字符串 $X \in \Sigma^n$，按如下方式预处理 X。只要存在两个相邻的指标 i、$i+1$，使得 $X[i+1] = \overline{X[i]}$ 且 $X[i] \in T$，就将它们删除。令所得字符串为 X'。我们提出以下断言。

命题 4.7. $\mathsf{dyck}^w(X) = \mathsf{dyck}^w(X')$。

证明。令 $\mathcal{M}$ 为 X 的一个最优对齐。为导出矛盾，假设对于两个连续索引 i、$i+1$、$X[i+1] = \overline{X[i]}$、$X[i] \in T$，但 $(i, i+1) \notin \mathcal{M}$。接下来，根据 i、$i+1$ 的匹配索引，我们考虑以下三种情形。

情形 1：令 (j,i)、$(i+1,k) \in \mathcal{M}$，其中 $j \in [0 \ldots i) \cup \{\perp\}$ 且 $k \in (i+1 \ldots |X|) \cup \{\perp\}$。在这种情形下，我们构造另一个对齐 $\mathcal{M}' = \mathcal{M} \setminus \{(j,i), (i+1,k)\} \cup \{(i,i+1), (j,k)\}$。我们论证 $\mathsf{dyck}^w(X[j]X[k]) \leq \mathsf{dyck}^w(X[j]X[i]) + \mathsf{dyck}^w(X[i+1]X[k])$，从而证明 $\mathsf{dyck}^w_{\mathcal{M}'}(X) \leq \mathsf{dyck}^w_{\mathcal{M}}(X)$。根据命题 4.6，令 a、$b \in T \cup \{\epsilon\}$ 满足 $\mathsf{dyck}^w(X[j]X[i]) = w(X[j],a) + w(X[i],\overline{a})$，且 $\mathsf{dyck}^w(X[i+1]X[k]) = w(X[i+1],b) + w(X[k],\overline{b})$。因此，

$$\begin{aligned}
\mathsf{dyck}^w(X[j]X[i]) + \mathsf{dyck}^w(X[i+1]X[k]) &= w(X[j],a) + w(X[i],\overline{a}) + w(X[i+1],b) + w(X[k],\overline{b}) \\
&= w(X[j],a) + w(a,\overline{X[i]}) + w(X[i+1],b) + w(X[k],\overline{b}) \\
&\geq w(X[j],\overline{X[i]}) + w(X[i+1],b) + w(X[k],\overline{b}) \\
&\geq w(X[j],b) + w(X[k],\overline{b}) \\
&\geq \mathsf{dyck}^w(X[j],X[k])
\end{aligned}$$

第二个等式成立是因为 w 具有反对称性；因此 $w(X[i], \overline{a}) = w(a, \overline{X[i]})$。第三和第四个不等式成立是因为 w 满足三角不等式，并且 $\overline{X[i]} = X[i+1]$。最后一个不等式由断言4.6得出。

情形 2：令 (k,i)、$(j,i+1) \in \mathcal{M}$，其中 $k,j \in [0 \ldots i) \cup \{\perp\}$ 且 $j < k$。在这种情形下，我们创建另一个对齐 $\mathcal{M}' = \mathcal{M} \setminus \{(k,i), (j,i+1)\} \cup \{(i,i+1), (j,k)\}$。我们证明 $\mathsf{dyck}^w(X[j]X[k]) \leq \mathsf{dyck}^w(X[k]X[i]) + \mathsf{dyck}^w(X[j]X[i+1])$，从而证明 $\mathsf{dyck}^w_{\mathcal{M}'}(X) \leq \mathsf{dyck}^w_{\mathcal{M}}(X)$。根据命题 4.6，令 $a, b \in T \cup \{\epsilon\}$ 满足 $\mathsf{dyck}^w(X[k]X[i]) = w(X[k],a) + w(X[i],\overline{a})$，且 $\mathsf{dyck}^w(X[j]X[i+1]) =$

$w(X[j], b) + w(X[i+1], \overline{b})$。因此，

$$\begin{aligned}
\mathsf{dyck}^w(X[k]X[i]) + \mathsf{dyck}^w(X[j]X[i+1]) &= w(X[k], a) + w(X[i], \overline{a}) + w(X[j], b) + w(X[i+1], \overline{b}) \\
&= w(X[k], a) + w(a, \overline{X[i]}) + w(X[j], b) + w(X[i+1], \overline{b}) \\
&\geq w(X[k], \overline{X[i]}) + w(X[i+1], \overline{b}) + w(X[j], b) \\
&\geq w(X[k], \overline{b}) + w(X[j], b) \\
&\geq \mathsf{dyck}^w(X[j], X[k])
\end{aligned}$$

情形 3：设 $(i, k), (i+1, j) \in \mathcal{M}$ 其中 $k, j \in (i+1 \ldots |X|) \cup \{\perp\}$ 且 $j < k$。在这种情形下，我们创建另一个对齐 $\mathcal{M}' = \mathcal{M} \setminus \{(i, k), (i+1, j)\} \cup \{(i, i+1), (j, k)\}$。我们论证 $\mathsf{dyck}^w(X[j]X[k]) \leq \mathsf{dyck}^w(X[i]X[k]) + \mathsf{dyck}^w(X[i+1]X[j])$，从而证明 $\mathsf{dyck}^w_{\mathcal{M}'}(X) \leq \mathsf{dyck}^w_{\mathcal{M}}(X)$。根据命题 4.6，设 $a, b \in T \cup \{\epsilon\}$ 满足 $\mathsf{dyck}^w(X[i]X[k]) = w(X[i], a) + w(X[k], \overline{a})$ 且 $\mathsf{dyck}^w(X[i+1]X[j]) = w(X[i+1], b) + w(X[j], \overline{b})$。因此，

$$\begin{aligned}
\mathsf{dyck}^w(X[i]X[k]) + \mathsf{dyck}^w(X[i+1]X[j]) &= w(X[i], a) + w(X[k], \overline{a}) + w(X[i+1], b) + w(X[j], \overline{b}) \\
&= w(X[i], a) + w(X[k], \overline{a}) + w(X[j], \overline{b}) + w(\overline{b}, \overline{X[i+1]}) \\
&\geq w(X[i], a) + w(X[k], \overline{a}) + w(X[j], \overline{X[i+1]}) \\
&\geq w(X[j], a) + w(X[k], \overline{a}) \\
&\geq \mathsf{dyck}^w(X[j], X[k])
\end{aligned}$$

□

预处理可在 $O(n)$ 时间内完成。此外，我们可以假设，预处理后的字符串中不存在两个相邻且能够对齐的符号。由此并结合 [], 中的命题 35，我们可以得到以下断言。

命题 4.8. 设 $X \in \Sigma^n$。存在一种算法，可在 $O(n)$ 时间内对 X 进行预处理，并且要么判定 $\mathsf{dyck}^w(X) > k$，要么输出一个长度至多为 n 的字符串 X'，使得 $\mathsf{dyck}^w(X) = \mathsf{dyck}^w(X')$，且 X' 至多有 $2k$ 个谷。

因此，从现在起，我们假设 X 已经过预处理，且至多有 $2k$ 个谷。

4.3 周期性缩减

定义 4.9. 对于字符串 X 的片段 $k \in \mathbb{Z}_{\geq 0}$ 和 $X[a \ldots b)$，以及 $X[c \ldots d)$，若满足 $X[a \ldots b) \in T^*$, $X[c \ldots d) \in \overline{T}^*$, $b - a = d - c$, $b \leq c$，且 $H(b) + H(c) - 2\min_{m \in [b \ldots c]} H(m) \leq 2k$，则称它们是 k-同步的。

注意，$X[a \ldots b)$ 和 $X[c \ldots d)$ 是 0-同步的，当且仅当 (a, b, c, d) 是一个梯形。

定义 4.10。对于 $k \in \mathbb{Z}_{\geq 0}$ 和斜度量权重函数 w，字符串 P、$P' \in T^*$ 称为 $\mathsf{dyck}^w_{\leq k}$-等价，如果

$$\mathsf{dyck}^w_{\leq k}(X) = \mathsf{dyck}^w_{\leq k}(X[0 \ldots a) \cdot P' \cdot X[b \ldots c) \cdot \overline{P'} \cdot X[d \ldots |X|))$$

对于每个具有 k-同步片段 $X[a \ldots b) = P$ 和 $X[c \ldots d) = \overline{P}$ 的字符串 X，该条件均成立。

事实 4.11 (事实 36, [BO16])。设 $\mathcal{M}$ 是一个对齐，满足 $\mathsf{Dyck}^w_{\mathcal{M}}(X) \leq k$。若 $X[a \ldots b) \simeq_{\mathcal{M}} X[c \ldots d)$，则片段 $X[a \ldots b)$ 和 $X(c \ldots d]$ 是 k-同步的。

事实 4.12。考虑一个字符串 X 和一个对齐 $\mathcal{M} \in \mathsf{M}(X)$，满足 $\mathrm{dyck}_{\mathcal{M}}(X) \le k$，其中某个 $k \in \mathbb{Z}_{\ge 0}$。此外，设 $X[a..b)$ 和 $X[c..d)$ 是长度为 $\ell > 6k$ 的 k-同步片段。那么，存在长度为 $\ell' \ge \frac{\ell-6k}{k+1}$ 的 k-同步片段 $X[a'..b')$ 和 $X[c'..d')$，使得 $X[a'..b') \simeq_{\mathcal{M}} X[c'..d')$ 和 $a \le a' \le b' \le b \le c \le c' \le d' \le d$。此外，我们还有 $|(a+d) - (a'+d')| \le 4k$。

证明。由于 $\mathcal{M}$ 是非交叉的，它与 $[a..b)\times[d..|X|)$ 或 $[0..a)\times[c..d)$ 不相交。根据对称性（至多相差一个反向互补），不妨设 $\mathcal{M}$ 与 $[a..b)\times[d..|X|)$ 不相交。考虑满足 $X[x] \simeq_{\mathcal{M}} \overline{X[y]}$ 的 $x \in [a..b-4k)$。这一假设意味着 $y < d$。此外，$b - x = H(b) - H(x) > 4k$，因此 $H(x) < H(b) - 4k$。同时，$|H(y+1) - H(x)| \le 2k$，因此 $H(y+1) < H(b) - 2k$。由于 $X[a..b)$ 和 $X[c..d)$ 是 k-同步的，这意味着 $y+1 \notin [b..c]$，即 $y \in [c..d)$。考虑片段 $X[a..b-4k)$，以及 $X[c..d)$ 的最小子片段，该子片段包含与 $X[x]$ 中位置完美匹配的位置 $\mathcal{M}$，且 $x \in [a..b-4k)$。这两个片段至多包含 $2k$ 个被删除或匹配不完美的位置。剩余位置构成 $X[a..b-4k)$ 和 $X[c..d)$ 的一个公共子序列；该子序列至多被中断 k 次，因此存在一个长度至少为 $\frac{\ell-6k}{k+1}$ 的连续子序列 $X[a'..b') \simeq_{\mathcal{M}} X[c'..d')$。由于 $|H(a) - H(d)| \le 2k$ 以及 $|H(a') - H(d')| \le 2k$，我们有 $4k \ge |H(a) - H(d) - H(a') + H(d')| = |a - a' + d - d'|$。 □

引理 4.13. 令 $k \in \mathbb{Z}_+$，令 $Q \in T^*$ 为一个字符串，并令 $e, e' \in \mathbb{Z}_{\ge 8k}$。那么，对于任意斜度量权重函数 w，Q^e 和 $Q^{e'}$ 都是 $\mathrm{dyck}^w_{\le k}$-等价的。

证明. 不失一般性，假设 Q 是本原的。（如果 $Q = R^m$ 对于 $m \in \mathbb{Z}_{\ge 2}$，那么 $Q^e = R^{me}$ 和 $Q^{e'} = R^{me'}$ 可以解释为 R 的幂，而不是 Q 的幂。）令 $q = |Q|$。考虑一个字符串 X 及位置 $p_T, p_{\overline{T}}$，使得 $Q^e = X[p_T..p_T + e\cdot q)$ 和 $\overline{Q^e} = X(p_{\overline{T}} - e\cdot q..p_{\overline{T}}]$ 是 k-同步片段。记作 $X[0..p_T)\cdot Q^{e'}\cdot X[p_T + e\cdot q..p_{\overline{T}} - e\cdot q]\cdot \overline{Q^{e'}}\cdot X(p_{\overline{T}}..|X|)$。此外，令 $\mathcal{M} \in \mathsf{M}(X)$ 为一个对齐，使得 $\mathrm{dyck}^w(X,Y) = \mathrm{dyck}^w_{\mathcal{M}}(X,Y) \le k$。

断言 4.14。存在 i_T、$i_{\overline{T}} \in [0..7k]$ 使得

$$X[p_T + i_T\cdot q..p_T + (i_T+1)\cdot q) \simeq_{\mathcal{M}} \overline{X(p_{\overline{T}} - (i_{\overline{T}}+1)\cdot q..p_{\overline{T}} - i_{\overline{T}}\cdot q]}.$$

证明。考虑从位置 $p_T + i\cdot q$ 开始的 $8k$ 个 Q 的出现，对于 $i \in [0..7k]$（将此片段记为 P），以及在位置 $p_{\overline{T}} - i\cdot q$ 结束的 $8k$ 个 $\overline{Q}$ 的出现，对于 $i \in [0..7k]$（将此片段记为 $\overline{P}$）。注意，P、$\overline{P}$ 也都是 k 同步的片段。因此根据事实 4.12，Q 在 P 中至少有一次出现，从索引 ℓ 开始，使得 $\mathcal{M}$ 与 $\overline{P}$ 中的某个片段完全匹配。于是我们可以定义 $i_T \in [0..7k]$，使得 $\mathcal{M}$ 与某个片段 $X(s_{\overline{T}} - q..s_{\overline{T}}] \in \overline{P}$）完全匹配（$X[p_T + i_T\cdot q..p_T + (i_T+1)\cdot q)$）。根据 $\overline{P}$ 的定义，有 $s_{\overline{T}} \ge p_{\overline{T}} - 7kq$。此外，由于 Q 是本原的（即不同于其所有非平凡循环旋转），我们得出，对于某个 $i_{\overline{T}} \in [0..7k]$，有 $s_{\overline{T}} = p_{\overline{T}} - i_{\overline{T}}\cdot q$。

□

现在，如果将 $Q^e = X[p_T..p_T + e\cdot q)$ 替换为 $Q^{e'}$，并将 $\overline{Q^e} = X(p_{\overline{T}} - e\cdot q..p_{\overline{T}}]$ 替换为 $\overline{Q^{e'}}$（其中 $e' \ge e-1$），我们可以将其解释为将 $Q = X[p_T + i_T\cdot q..p_T + (i_T+1)\cdot q)$ 替换为 $Q^{1+e'-e}$，并将 $\overline{Q} = X(p_{\overline{T}} - (i_{\overline{T}}+1)\cdot q..p_{\overline{T}} - i_{\overline{T}}\cdot q]$ 替换为 $\overline{Q^{1+e'-e}}$。根据断言 4.14，$\mathcal{M}$ 可以在不改变其代价的情况下轻松调整，因此 $\mathrm{dyck}^w(X') \le \mathrm{dyck}^w_{\mathcal{M}}(X) = \mathrm{dyck}^w(X)$。如果 $e' < e-1$，我们重复上述论证，每次将指数 e 减小一步，仍可得出 $\mathrm{dyck}^w(X') \le \mathrm{dyck}^w(X)$。无论哪种情况，通过 (X,e) 与 (X',e') 之间的对称性，都可得到反向不等式。 □

如果字符串 $P \in T^*$ 不包含任何满足 $|Q| \in [1..4k]$ 的子串 Q^{8k+1}，我们就称它避免了 k-周期性。

引理 4.15. 设 $k \in \mathbb{Z}_+$，且 P、$P' \in T^*$ 是长度至少为 $156k^3$ 的字符串，使得 $P[0..78k^3) = P'[0..78k^3)$ 和 $P[|P| - 78k^3..|P|) = P'[|P'| - 78k^3..|P'|)$ 均不含 k-周期性。那么，对于任意斜度量权重函数 w，P 与 P' 是 $\mathsf{dyck}^w_{\le k}$-等价的。

证明. 考虑字符串 X 及位置 p_T、$p_{\overline{T}}$，使得 $P = X[p_T..p_T + |P|)$、$\overline{P} = X(p_{\overline{T}} - |P|..p_{\overline{T}}]$ 是 k-同步片段。记 $X' = X[0..p_T) \cdot P' \cdot X[p_T + |P|..p_{\overline{T}} - |P|] \cdot \overline{P'} \cdot X(p_{\overline{T}}..|X|]$。此外，设 $\mathcal{M} \in \mathsf{M}(X)$ 是一个对齐，使得 $\mathsf{dyck}^w(X) = \mathsf{dyck}^w_{\mathcal{M}}(X) \le k$。

断言 4.16。存在 d、$e \in [0..78k^3]$，使得 $(p_T + d$、$p_{\overline{T}} - d) \in \mathcal{M}$，且 $(p_T + |P| - e$、$p_{\overline{T}} - |P| + e) \in \mathcal{M}$。

证明。根据事实 4.12，$X[p_T..p_T + 78k^3)$ 包含一个长度至少为 $\frac{78k^3 - 6k}{k+1} \ge 36k^2$ 的片段，该片段与 $X(p_{\overline{T}} - 78k^3..p_{\overline{T}}]$ 的一个片段 $\mathcal{M}$ 完美匹配。因此，令 $R := X[r_T..r_T + |R|)$ 为包含于 $X[p_T..p_T + |P|)$ 中的一个长度至少为 $36k^2$ 的片段，并且该片段与 $X[r_{\overline{T}} - |R|..r_{\overline{T}}) = \overline{R}$ 完美匹配。此外，令 $r'_{\overline{T}} := p_T + p_{\overline{T}} - r_T$。如果 $r_{\overline{T}} = r'_{\overline{T}}$，则对于 $d = r_T - p_T = p_{\overline{T}} - r_{\overline{T}}$，断言成立。否则，$X[r_{\overline{T}} - |R|..r_{\overline{T}})$ 和 $X[r'_{\overline{T}} - |R|..r'_{\overline{T}})$ 都是 $\overline{R}$ 在 X 中的出现。此外，$0 < |r_{\overline{T}} - r'_{\overline{T}}| \le |(p_{\overline{T}} - r'_{\overline{T}}) - (p_{\overline{T}} - r_{\overline{T}})| \le |(r_T - p_T) - (p_{\overline{T}} - r_{\overline{T}})| + |(r_T - p_T) - (p_{\overline{T}} - r'_{\overline{T}})| \le 2\mathsf{Dyck}^w_{\mathcal{M}}(X) + 2k \le 4k$。因此，$\mathsf{per}(\overline{R}) \le |r_{\overline{T}} - r'_{\overline{T}}| \le 4k$，且 $\exp(\overline{R}) \ge \frac{|\overline{R}|}{4k} \ge 9k$。由于 $X[r'_{\overline{T}} - |\overline{R}|..r'_{\overline{T}})$ 包含于 $X(p_{\overline{T}} - |P|..p_{\overline{T}}] = \overline{P[0..78k^3)}$ 中，这与 $P[0..78k^3)$ 的假设矛盾，因此 $\overline{P[0..78k^3)}$ 避免了 k—周期性。

断言的第二部分可用类似的方法证明。 □

由于 $X[p_T + d..p_T + |P| - e) \in T$和 $X[p_T + d..p_T + |P| - e) = \overline{X(p_{\overline{T}} - |P| + e..p_{\overline{T}} - d]}$，$\mathcal{M}$ 的最优性保证了 $X[p_T + d..p_T + |P| - e) \simeq_{\mathcal{M}} X(p_{\overline{T}} - |P| + e..p_{\overline{T}} - d]$。因此，若将 $P = X[p_T..p_T + |P|) = X(p_{\overline{T}} - |P|..p_{\overline{T}}]$ 替换为 P'，我们可以将其解释为 $P[d..|P| - e) = X[p_T + d..p_T + |P| - e) = X(p_{\overline{T}} - |P| + e..p_{\overline{T}} - d]$，其中有 $P'[d..|P'| - e)$。由于 $X[p_T + d..p_T + |P| - e) \simeq_{\mathcal{M}} X(p_{\overline{T}} - |P| + e..p_{\overline{T}} - d]$，无需修改代价即可直接调整对齐 $\mathcal{M}$，因此 $\mathsf{dyck}^w(X') \le \mathsf{dyck}^w_{\mathcal{M}}(X) = \mathsf{dyck}^w(X)$。由 (X, P) 与 (X', P') 之间的对称性可得反向不等式。□

推论 4.17. 设 $k \in \mathbb{Z}_+$。对于每个字符串 $P \in T^*$，都存在一个长度至多为$156k^3$ 的字符串，该字符串对于每个偏斜度量权重函数 w都与 P $\mathsf{Dyck}^w_{\le k}$-等价。

证明。我们对 $|P|$ 进行归纳，基本情形 $|P| \le 156k^3$显然成立。若 $|P| \ge 156k^3$ 和P不含 k-周期性，则引理 4.15 表明，P与一个字符串 $P' := P[0..78k^3) \cdot P[|P| - 78k^3..|P|)$ 等价，其长度为 $156k^3$。因此，假设 P包含一个片段 $P[i..j) = Q^{8k+1}$和 $|Q| \in [1..4k]$。由引理 4.13，Q^{8k+1} 与 Q^{8k}等价，因此 P与字符串$P' := P[0..i) \cdot P[i + |Q|..|P|)$ 等价。根据归纳假设，P' 与某个长度至多为 $156k^3$的字符串 P''等价；再由所考虑等价关系的传递性，P 也与 P''等价。

□

4.4 算法

引理 4.18. 存在线性时间算法，对于给定的字符串 P 和整数 $k \in \mathbb{Z}_+$，构造一个长度至多为 $156k^3$ 的字符串 P'，使其对于每个斜度量权重函数 w 都与 P $\mathsf{Dyck}^w_{\le k}$-等价。此外，P' 避免了 k-周期性。

算法 8：构造一个长度至多为 $156k^3$ 的字符串，使其与 P 在 $\text{Dyck}^w_{\le k}$ 意义下等价。

```
1 Dyck 归约(P, k):
2 |   P' ← 周期性约简(P, 8k, {Q ∈ T^+ : |Q| ≤ 4k 且 Q 是本原的});
3 |   如果 |P'| ≥ 156k^3，则返回 P'[0..78k^3) · P'[|P'| − 78k^3..|P'|);
4 |   否则返回 P';
```

证明。我们对 T^* 中长度为 $[1..4k]$ 的所有本原字符串应用算法 1，其中 $e = 8k$ 和 $\mathcal{Q}$ 由这些字符串组成。如果所得字符串 P' 满足 $|P'| < 156k^3$，则返回 P'。根据引理 4.13 和引理 2.13，字符串 P' 与 P $\text{Dyck}^w_{\le k}$-等价，并且避免了 k-周期性。因此，如果 $|P'| \le 156k^3$，则算法是正确的。否则，我们返回 $P'[0..78k^3) \cdot P'[|P'| - 78k^3..|P'|)$。 $P'[0..78k^3)$ 和 $P'[|P'| - 78k^3..|P'|)$ 都避免了 k-周期性，因此根据引理 4.15， $P'[0..78k^3) \cdot P'[|P'| - 78k^3..|P'|)$ 与 P $\text{Dyck}^w_{\le k}$-等价。根据引理 2.13，运行时间是线性的（测试一个本原片段是否属于 $\mathcal{Q}$ 可简化为检查其长度是否不超过 $4k$）。 □

定理 4.19. 存在一个运行时间为 $O(n+k^5)$ 的算法，给定一个预处理后的字符串 X 和一个整数 $k \in \mathbb{Z}_+$，构造字符串 X'，其长度至多为 $630k^4$，使得对于每个斜度量权重函数 w，$\text{dyck}^w_{\le k}(X) = \text{dyck}^w_{\le k}(X')$ 均成立。

证明。我们的过程由算法 9 实现。首先，如果 X 的长度已经不超过$630k^4$，那么我们原样返回 X。如果 $\text{dyck}(X) > k$，我们返回字符串 a^{k+1}，其中 $a \in \Sigma$ 是任意字符。如果 $\text{dyck}(X) \le k$，我们构造一个（无权）代价至多为 k的 Dyck语言对齐 $\mathcal{M} \in \mathsf{M}(X)$。然后，我们按照以下方式使用 $\mathcal{M}$ 构造输出字符串 X'：从左向右扫描 X；如果扫描到的字符 $X[i]$ 被 $\mathcal{M}$ 编辑过，就将其追加到 X' 中（此处，如果 $j > i$，则 $\text{Order}(i,j) = 1$，否则为 0）。否则， $X[i]$ 在 $\mathcal{M}$下匹配。如果 $X[i] \in T$，我们继续扫描后续字符，以确定极大片段 $P = X[i..j) \in T^*$，使得存在片段 $X(i'..j']$，其中 $\text{dyck}(X[i..j)X(i'..j']) = 0$，且 $\mathcal{M}$与 $X[i..j$匹配），并且 $X(i'..j']$。接下来，我们对 P应用引理 4.18 中的约简，并将约简后的字符串追加到 X'。否则，$X[i] \in \overline{T}$。这里我们也继续扫描后续字符，以确定极大片段 $Q = X[i..j) \in \overline{T}^*$，使得存在片段 $X(i'..j']$，其中 $\text{dyck}(X(i'..j']X[i..j)) = 0$，且 $\mathcal{M}$与 $X[i..j$匹配），并且 $X(i'..j']$。接下来，我们考虑 $\overline{Q}$（注意$\overline{Q} \in T^*$），对 $\overline{Q}$ 应用引理 4.18 中的约简，并将约简后字符串的反向互补追加到 X'。

现在证明所得字符串 X' 满足 $\text{dyck}^w_{\le k}(X) = \text{dyck}^w_{\le k}(X')$。当算法在第 2 行返回 X 时，这显然成立。如果 $\text{dyck}(X) > k$，那么 $\text{dyck}_{\le k}(X) = \text{dyck}_{\le k}(a^{k+1}) = \infty$，因此也有 $\text{dyck}^w_{\le k}(X) = \text{dyck}^w_{\le k}(a^{k+1}) = \infty$，因为采用归一化权重函数的加权 Dyck 编辑距离至少与无权 Dyck 编辑距离一样大。在剩下的情况下，我们假设 $\text{dyck}(X) \le k$。令 $S = \{P_1, .., P_\ell\}$ 为在第 13 行使用 DyckReduction() 例程生成并处理过的 T^*的片段集合。类似地，令$\overline{S} = \{Q_1, .., Q_{\ell'}\}$ 为曾经生成，且其反向补集曾使用 DyckReduction() 例程在第 20 行处理过的 $\overline{T}^*$ 的片段集合。根据构造，不难看出：(i) 这些片段互不相交；(ii) 对所有 $i \in [0..n)$，如果 $X[i] \in T$且未被 $\mathcal{M}$编辑，那么存在某个 P_j 使得 $X[i] \in P_j$；如果 $X[i] \in \overline{T}$且未被 $\mathcal{M}$编辑，那么存在某个 Q_j 使得 $X[i] \in Q_j$；(iii) 这些片段是极大的，具体而言，若 $P_i = X[a..b) \in S$，则要么 $\mathcal{M}(b) \ne \mathcal{M}(b-1) - 1$，要么 $X[b] \ne X[\mathcal{M}(b)]$；对于 $\overline{S}$中的片段也同样成立。接下来，我们证明对于每个 $P_i = X[a..b) \in S$，$\exists Q_j = X(c..d] \in \overline{S}$，使得 $|P_i| = |Q_j|$，且对于所有$k \in [0..|P_i|)$，$(a+k, d-k) \in \mathcal{M}$ 且 $X[a+k] = \overline{X[d-k]}$。为此，我们首先断言 $\mathcal{M}(b-1)$

算法 9：构造长度至多为 $630k^4$ 的字符串 X'，使得 $\mathsf{dyck}^w_{\le k}(X) = \mathsf{dyck}^w_{\le k}(X')$

```
DyckKernel(X, k):
    如果 |X| ≤ 630k^4，则返回 (X);
    如果 dyck(X) > k，则对某个 a ∈ Σ 返回 (a^{k+1});
    令 M ∈ M(X) 为满足 dyck_M(X) ≤ k 的一个 Dyck 语言对齐;
    X', P, Q ← ε;
    对于 i ← 0 到 n − 1，执行
        如果 M(i) = ⊥ 或 Dyck(X[i]X[M(i)]) · Order(i, M(i)) = 1 或
         Dyck(X[M(i)]X[i]) · Order(M(i) i) = 1 则
            X' ← X' · X[i]
        否则，如果 X[i] ∈ T 且 M(i + 1) = M(i) − 1 且 Dyck(X[i + 1]X[M(i) − 1]) = 0
         then
            P ← P · X[i]
        否则，如果 X[i] ∈ T 则
            P ← P · X[i];
            P ← Dyck 归约(P, k);
            X' ← X' · P;
            P ← ε;
        否则，如果 X[i] ∈ T̄ 和 M(i + 1) = M(i) − 1 且 dyck(X[M(i) − 1]X[i + 1]) = 0
         then
            Q ← Q · X[i]
        else
            Q ← Q · X[i];
            Q ← overline(Dyck 归约(overline(Q), k));
            X' ← X' · Q;
            Q ← ε;
    返回 (X')
```

是某个 $Q_j \in \overline{S}$ 的起始索引。否则， $\mathcal{M}(b) = \mathcal{M}(b-1) - 1$ 和 $X[b] \neq X[\mathcal{M}(b)]$；这与 P_i 的极大性矛盾。此外，根据构造，对于所有 $k \in [a\,..\,b)$， $X[\mathcal{M}(k)] \in Q_j$。最后，我们论证 $\mathcal{M}(a)$ 是 Q_j 的结束索引。否则， $\mathcal{M}(a-1) = \mathcal{M}(a) + 1$，这与事实 a 是来自 S 的某个片段的起始索引相矛盾。类似地，我们可以证明，对于每个 $Q_j \in \overline{S}$，都存在对应的匹配 $P_i \in S$，这在来自 S 和 $\overline{S}$ 的一对片段之间建立了一一对应关系。因此，对于一个片段 $P_i \in S$，令 $\mathcal{M}(P_i)$ 表示来自 $\overline{S}$ 的对应匹配片段，并可将 $S \cup \overline{S} = \cup_{i\in[\ell]}$ 表示为 $P_i, \mathcal{M}(P_i)$。根据事实 4.11， P_i、 $\mathcal{M}(P_i)$ 是 k-同步的。接下来，在算法中，对于每一对 $(P_i, \mathcal{M}(P_i))$，我们将表示 P_i 的字符串 DyckReduction(P_i)，以及 DyckReduction(P_i)（注意 $\mathcal{M}(P_i) = P_i$），即表示 $\mathcal{M}(P_i)$ 的字符串，添加到 X' 中。根据这样一个事实：任何未包含在来自 $S \cup \overline{S}$ 的片段中的字符，都会由 $\mathcal{M}$ 编辑，因此会直接复制到 X' 中；反复应用引理 4.18，对于每一对 $(P_i, \mathcal{M}(P_i))$，我们断言 $\mathsf{dyck}^w_{\leq k}(X) = \mathsf{dyck}^w_{\leq k}(X')$。

接下来，我们证明返回的字符串长度至多为 $630k^4$。当算法在第 2 行或第 3 行终止时，这一点显然成立。否则，我们创建一个字符串 X'，并将由 $\mathcal{M}$ 编辑的字符直接复制到其中。不过， $\mathcal{M}$ 删除或替换的字符至多有 $2k$ 个。接下来，我们找出极大片段 $P = X[i\,..\,j) \in T^*$，使得存在另一个片段 $X(i'\,..\,j'] \in \overline{T}^*$，它与 P通过 $\mathcal{M}$ 匹配。 P 的极大性以及对 X 的预处理确保 $X[j]$ 和 $X[i']$ 中至少有一个被 $\mathcal{M}$ 编辑。我们将这些字符称为 P 的边界字符。注意，对于任意两个不同的片段 P、 $P' \in T^*$，它们的边界字符各不相同，并且根据构造， P、 P' 两两不相交。由于 $\mathcal{M}$ 编辑的字符至多有 $2k$ 个，因此我们得出结论：算法在 T^* 上构造的片段至多有 $2k$ 个。对于每个这样的片段，按照引理 4.18 中的归约，我们向 X' 添加一个长度为 $156k^3$ 的子串。因此，所有子串的总长度为 $312k^4$。对于片段 $Q \in \overline{T}^*$，我们也可以类似地进行论证。因此， X' 的总长度至多为 $2 \cdot 312k^4 + 2k < 630k^4$。

接下来需要分析我们这套方法的复杂度。我们使用算法 [FGK+22a] 来检查 $\mathsf{Dyck}(X) \leq k$是否成立；若成立，则构造对齐 $\mathcal{M}$。这需要 $\mathcal{O}(n+k^5$ 时间。接下来，我们从左到右扫描 X一次。在整个过程中， *if/else* 语句中的所有条件都可以在 $O(1)$ 时间内检查。此外，任何字符传入 Dyck 归约() 过程的次数至多为两次。因此，根据引理 4.18，给定 X 和 $\mathcal{M}$， X' 可以在线性时间内构造。□

定理 1.3 的证明。首先，我们按照第 4.2.1 节所述步骤，在 X 上线性时间内进行预处理，构造字符串 X'，使得 $\mathsf{dyck}^w_{\leq k}(X') = \mathsf{dyck}^w_{\leq k}(X)$。接下来，我们对 X'应用定理 4.19，构造长度为 $\mathcal{O}(k^4)$ 的字符串 X''，使得 $\mathsf{dyck}^w_{\leq k}(X'')$ =$\mathsf{dyck}^w_{\leq k}(X')$。这需要时间$O(n + k^5)$。最后，如果 $X = a^{k+1}$（这可以在时间 $O(k)$ 内检查），则输出距离为 $> k$。否则，我们使用 [Mye95] 中的动态规划算法计算 $\mathsf{dyck}^w_{\leq k}(X'')$，耗时 $O(k^{12})$。因此，总运行时间为 $O(n + k^{12})$。□

A 第 2 节的延后证明 2

接下来，我们给出第 2 节中事实的缺失证明。

¯事实 2.5. 如果 w 是 Σ 上的拟度量，那么 ed^w 是 Σ^* 上的拟度量。在这种情况下，$\mathsf{ed}^w(X, Y)$ 也可以等价地定义为将 X 转换为 Y 的编辑序列的最小代价。

证明。考虑任意字符串 $X, Y, Z \in \Sigma^*$，以及对齐 $\mathcal{A} = (x_t, y_t)_{t=0}^m \in \mathsf{A}(X, Y)$ 和 $\mathcal{B} = (\hat{}y_t, \hat{}z_t)\,\hat{}_{t=0}^m \in \mathsf{A}(Y, Z)$。我们构造一个乘积对齐 $\mathcal{A} \otimes \mathcal{B} \in \mathsf{A}(X, Z)$，使得 $\mathsf{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(X, Z) \leq \mathsf{ed}^w_{\mathcal{A}}(X, Y) + \mathsf{ed}^w_{\mathcal{B}}(Y, Z)$。记 $\mathcal{A}' = (x_t, y_t)_{t=0}^{m-1}$和 $\mathcal{B}' = (\hat{}y_t, \hat{}z_t)\,\hat{}_{t=0}^{m-1}$，以及 $X' = X[0\,..\,|X|-1)$（如果 $X \neq \varepsilon$）、 $Y' = Y[0\,..\,|Y|-1)$（如果 $Y \neq \varepsilon$）和 $Z' = Z[0\,..\,|Z|-1)$（如果 $Z \neq \varepsilon$）。

我们对 $m + \hat{}m$ 进行归纳法，并根据 $\mathcal{A}$ 和 $\mathcal{B}$ 如何处理 X、Y和 Z的尾随字符，分别讨论几种情况。

1. $m = \hat{}m = 0$。在这种情况下， $X = Y = Z = \varepsilon$，我们定义 $\mathcal{A} \otimes \mathcal{B} := (0, 0)$。显然，$\mathrm{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(X, Z) = 0 = \mathrm{ed}^w_{\mathcal{A}}(X, Y) + \mathrm{ed}^w_{\mathcal{B}}(Y, Z)$。2. $(x_{m-1}, y_{m-1}) = (|X| - 1, |Y|)$，也就是说， $\mathcal{A}$ 删除 $X[|X| - 1]$。在这种情况下， $\mathcal{A}' \in \mathsf{A}(X', Y)$，我们定义 $\mathcal{A}\otimes\mathcal{B} := (\mathcal{A}'\otimes\mathcal{B})\odot(|X|, |Z|)$，其中 $\odot$表示串接，因此 $\mathcal{A}\otimes\mathcal{B}$删除$X[|X| - 1]$。根据归纳假设，$\mathrm{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(X, Z) = \mathrm{ed}^w_{\mathcal{A}'\otimes\mathcal{B}}(X', Z) + w(X[|X| - 1], \varepsilon) \le \mathrm{ed}^w_{\mathcal{A}'}(X', Y) + \mathrm{ed}^w_{\mathcal{B}}(Y, Z) + w(X[|X| - 1], \varepsilon) = \mathrm{ed}^w_{\mathcal{A}}(X, Y) + \mathrm{ed}^w_{\mathcal{B}}(Y, Z)$。3. $(\hat{}y\hat{}_{m-1}, \hat{}z_{\hat{m}-1}) = (|Y| - 1, |Z|)$，也就是说， $\mathcal{B}$插入 $Z[|Z| - 1]$。在这种情况下， $\mathcal{B}' \in \mathsf{A}(Y, Z')$，我们定义 $\mathcal{A} \otimes \mathcal{B} := (\mathcal{A} \otimes \mathcal{B}') \odot (|X|, |Z|)$，使得 $\mathcal{A} \otimes \mathcal{B}$ 插入 $Z[|Z| - 1]$。根据归纳假设，$\mathrm{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(X, Z) = \mathrm{ed}^w_{\mathcal{A}\otimes\mathcal{B}'}(X, Z') + w(\varepsilon, Z[|Z| - 1]) \le \mathrm{ed}^w_{\mathcal{A}}(X, Y) + \mathrm{ed}^w_{\mathcal{B}'}(Y, Z') + w(\varepsilon, Z[|Z| - 1]) = \mathrm{ed}^w_{\mathcal{A}}(X, Y) + \mathrm{ed}^w_{\mathcal{B}}(Y, Z)$。4. $(x_{m-1}, y_{m-1}) = (|X|, |Y| - 1)$ 且 $(\hat{}y\hat{}_{m-1}, \hat{}z_{\hat{m}-1}) = (|Y| - 1, |Z|)$，也就是说， $\mathcal{A}$插入 $Y[|Y| - 1]$并删除 $Y[|Y| - 1]$。在这种情况下， $\mathcal{A}' \in \mathsf{A}(X, Y')$ 且 $\mathcal{B}' \in \mathsf{A}(Y', Z)$，我们定义$\mathcal{A}\otimes\mathcal{B} := \mathcal{A}'\otimes\mathcal{B}'$。根据归纳假设，$\mathrm{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(X, Z) = \mathrm{ed}^w_{\mathcal{A}'\otimes\mathcal{B}'}(X, Z) \le \mathrm{ed}^w_{\mathcal{A}'}(X, Y')+\mathrm{ed}^w_{\mathcal{B}'}(Y', Z) \le \mathrm{ed}^w_{\mathcal{A}}(X, Y) + \mathrm{ed}^w_{\mathcal{B}}(Y, Z)$。5. $(x_{m-1}, y_{m-1}) = (|X|, |Y|-1)$ 且 $(\hat{}y\hat{}_{m-1}, \hat{}z_{\hat{m}-1}) = (|Y|-1, |Z|-1)$，也就是说， $\mathcal{A}$插入 $Y[|Y|-1]$，并且 $\mathcal{B}$将 $Y[|Y| - 1]$与 $Z[|Z| - 1]$对齐。在这种情况下， $\mathcal{A}' \in \mathsf{A}(X, Y')$ 且 $\mathcal{B}' \in \mathsf{A}(Y', Z')$，我们定义 $\mathcal{A} \otimes \mathcal{B} := (\mathcal{A}' \otimes \mathcal{B}') \odot (|X|, |Z|)$，使得 $\mathcal{A} \otimes \mathcal{B}$ 插入 $Z[|Z| - 1]$。根据归纳假设，$\mathrm{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(X, Z) = \mathrm{ed}^w_{\mathcal{A}'\otimes\mathcal{B}'}(X, Z') + w(\varepsilon, Z[|Z| - 1]) \le \mathrm{ed}^w_{\mathcal{A}'}(X, Y') + \mathrm{ed}^w_{\mathcal{B}'}(Y', Z') + w(\varepsilon, Y[|Y| - 1]) + w(Y[|Y| - 1], Z[|Z| - 1]) = \mathrm{ed}^w_{\mathcal{A}}(X, Y) + \mathrm{ed}^w_{\mathcal{B}}(Y, Z)$。6. $(x_{m-1}, y_{m-1}) = (|X|-1, |Y|-1)$ 且 $(\hat{}y\hat{}_{m-1}, \hat{}z_{\hat{m}-1}) = (|Y|-1, |Z|)$，也就是说， $\mathcal{A}$将 $X[|X|-1]$与 $Y[|Y| - 1]$ 对齐，并删除 $Y[|Y| - 1]$。在这种情况下， $\mathcal{A}' \in \mathsf{A}(X', Y')$ 且 $\mathcal{B}' \in \mathsf{A}(Y', Z)$，我们定义 $\mathcal{A} \otimes \mathcal{B} := (\mathcal{A}' \otimes \mathcal{B}') \odot (|X|, |Z|)$，使得 $\mathcal{A} \otimes \mathcal{B}$删除 $X[|X| - 1]$。根据归纳假设，$\mathrm{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(X, Z) = \mathrm{ed}^w_{\mathcal{A}'\otimes\mathcal{B}'}(X', Z) + w(X[|X| - 1], \varepsilon) \le \mathrm{ed}^w_{\mathcal{A}'}(X', Y') + \mathrm{ed}^w_{\mathcal{B}'}(Y', Z) + w(X[|X| - 1], Y[|Y| - 1]) + w(Y[|Y| - 1], \varepsilon) = \mathrm{ed}^w_{\mathcal{A}}(X, Y) + \mathrm{ed}^w_{\mathcal{B}}(Y, Z)$。7. $(x_{m-1}, y_{m-1}) = (|X| - 1, |Y| - 1)$ 且 $(\hat{}y\hat{}_{m-1}, \hat{}z_{\hat{m}-1}) = (|Y| - 1, |Z| - 1)$，也就是说， $\mathcal{A}$ 将 $X[|X|-1]$ 与 $Y[|Y|-1]$对齐，并且 $\mathcal{B}$将 $Y[|Y|-1]$与 $Z[|Z|-1]$对齐。在这种情况下， $\mathcal{A}' \in \mathsf{A}(X', Y')$ 且 $\mathcal{B}' \in \mathsf{A}(Y', Z')$，我们定义 $\mathcal{A} \otimes \mathcal{B} := (\mathcal{A}' \otimes \mathcal{B}') \odot (|X|, |Z|)$，使得 $\mathcal{A} \otimes \mathcal{B}$将$X[|X| - 1]$与 $Z[|Z| - 1]$对齐。根据归纳假设，$\mathrm{ed}^w_{\mathcal{A}\otimes\mathcal{B}}(X, Z) = \mathrm{ed}^w_{\mathcal{A}'\otimes\mathcal{B}'}(X', Z') + w(X[|X| - 1], Z[|Z|-1]) \le \mathrm{ed}^w_{\mathcal{A}'}(X', Y')+\mathrm{ed}^w_{\mathcal{B}'}(Y', Z) + w(X[|X|-1], Y[|Y|-1])+w(Y[|Y| - 1], Z[|Z| - 1]) = \mathrm{ed}^w_{\mathcal{A}}(X, Y) + \mathrm{ed}^w_{\mathcal{B}}(Y, Z)$。

容易验证，上述情形涵盖了所有可能性。特别是，情形 2 涵盖了 $\hat{}m = 0 < m$ 的情况，而情形 3 涵盖了 $m = 0 < \hat{}m$的情况。我们还要指出，情形 2和 3有时可能同时适用；按照约定，此时我们采用情形 2。最后，我们注意到，情形 5–7 依赖于 w 满足三角不等式这一假设。至此，事实第一部分的证明完成。

为了说明 $\mathrm{ed}^w(X, Y)$ 可以等价地定义为将 X 转换为 Y的编辑序列的最小代价，我们首先分别考察 $\mathcal{A}$ 中 X和 Y 的最小对齐里的每个操作，以构造从 $\mathcal{A}$出发的编辑序列 S。我们从右向左遍历 $\mathcal{A}$ 中的所有配对 (x_t, y_t)，从 $t = m - 1$开始，在处理完 $t = 0$ 后停止，并根据对齐的定义构造S：

1. 如果 $(x_t, y_t) = (x_{t+1} - 1, y_{t+1} - 1)$ 且 $X[x_t] = Y[y_t]$，则不向 S 添加任何内容。2. 如果 $(x_t, y_t) = (x_{t+1} - 1, y_{t+1} - 1)$ 且 $X[x_t] \ne Y[y_t]$，则向 S 添加将 $X[x_t]$ 替换为 $Y[y_t]$的操作。3. 如果 $(x_t, y_t) = (x_{t+1} - 1, y_{t+1})$，则向 S 添加删除 $X[x_t]$ 的操作。

4. 如果 $(x_t, y_t) = (x_{t+1}, y_{t+1} - 1)$，则向 S 添加在 X 的位置 x_t 插入 $Y[y_t]$ 的操作。

在所有情况下，我们都将 t 减 1。显然，所得编辑序列的代价与 $\mathcal{A}$相同；根据对齐的定义，S 将 X 转换为 Y。现在考虑将 X 转换为 Y的最小编辑序列 S，并据此构造对齐 $\mathcal{A} \in \mathsf{A}(X, Y)$，使得 $\mathsf{ed}^w_{\mathcal{A}}(X, Y) \leq cost(S)$（我们用 $cost(S)$ 表示 S所执行编辑的总代价）。我们沿用之前的记号 $\mathcal{A}'$、X'、Y'，并通过归纳法构造 $\mathcal{A}$:

1. 如果 $X[|X|-1]$ 被 S删除，且在 X末尾插入一个字符，那么 $\mathcal{A}' \in \mathsf{A}(X', Y')$，并令 $\mathcal{A} = \mathcal{A}' \odot (|X|, |Y|)$。注意，插入的字符 c 可以替换为 $Y[|Y|-1]$。令 S' 为序列 S，即从 X 末尾字符上去除插入、删除操作，并在可能的情况下去除替换操作后所得的序列。根据归纳假设和三角不等式，$\mathsf{ed}_{\mathcal{A}}(X, Y) = \mathsf{ed}_{\mathcal{A}'}(X', Y') + w(X[|X|-1], Y[|Y|-1]) \leq \mathsf{ed}_{\mathcal{A}'}(X', Y') + w(X[|X|-1], \varepsilon) + w(\varepsilon, Y[|Y|-1]) \leq \mathsf{ed}_{\mathcal{A}'}(X', Y') + w(X[|X|-1], \varepsilon) + w(\varepsilon, c) + w(c, Y[|Y|-1]) \leq cost(S') + w(X[|X|-1], \varepsilon) + w(\varepsilon, c) + w(c, Y[|Y|-1]) = cost(S)$。
2. 如果 $X[|X|-1]$ 被 S 删除，且在 X末尾没有插入字符，那么 $\mathcal{A}' \in \mathsf{A}(X', Y)$，并且我们设 $\mathcal{A} = \mathcal{A}' \odot (|X|, |Y|)$。令 S' 为序列 S，即未删除 $X[|X|-1]$时的序列。根据归纳假设，$\mathsf{ed}_{\mathcal{A}}(X, Y) = \mathsf{ed}_{\mathcal{A}'}(X', Y) + w(X[|X|-1], \varepsilon) \leq cost(S') + w(X[|X|-1], \varepsilon) = cost(S)$。3. 如果在 X末尾插入了一个字符，那么 $\mathcal{A}' \in \mathsf{A}(X, Y')$，并且我们设 $\mathcal{A} = \mathcal{A}' odot (|X|, |Y|)$。令 S' 为序列 S，即未进行这次插入时的序列。根据归纳假设，$\mathsf{ed}_{\mathcal{A}}(X, Y) = \mathsf{ed}_{\mathcal{A}'}(X, Y') + w(\varepsilon, Y[|Y|-1], \varepsilon) \leq cost(S') + w(\varepsilon, Y[|Y|-1]) = cost(S)$。4. 如果用 S替换了 $X[|X|-1]$，那么 $\mathcal{A}' \in \mathsf{A}(X', Y')$，并且我们设 $\mathcal{A} = \mathcal{A}' \odot (|X|, |Y|)$。令 S' 为 S 中未替换任何 $X[|X|-1]$ 时的序列，并令 C 为在 $X[|X|-1]$处由 S替换的字符所组成的有序列表。于是，根据归纳假设，$\mathsf{ed}_{\mathcal{A}}(X, Y) = \mathsf{ed}_{\mathcal{A}'}(X', Y') + w(X[|X|-1], Y[|Y|-1]) \leq \mathsf{ed}_{\mathcal{A}'}(X', Y') + \sum_{i=0}^{|C|-1} w(c_i, c_{i+1}) \leq cost(S') + \sum_{i=0}^{|C|-1} w(c_i, c_{i+1}) = cost(S)$。由归纳法可知，存在一个对齐 $\mathcal{A}$，其代价至多为 S 的代价。

□

事实 2.6. 考虑字符串 X及其片段 $X[i\,..\,j)$。那么，对于每个拟度量 w，我们有 $\mathsf{ed}^w(X, X[i\,..\,j)) = \mathsf{ed}^w(X[0\,..\,i) \cdot X[j\,..\,|X|), \varepsilon)$。

证明。$\mathsf{A}(X[0\,..\,i) \cdot X[j\,..\,|X|), \varepsilon)$ 中唯一的对齐会删除所有字符，因此 $\mathsf{ed}^w(X[0\,..\,i) \cdot X[j\,..\,|X|), \varepsilon) = \sum_{u \in [0\,..\,i) \cup [j\,..\,|X|)} w(X[u], \varepsilon)$。

接下来，考虑对齐 $\mathcal{A} \in \mathsf{A}(X, X[i\,..\,j))$，它会删除 $X[0\,..\,i)$、完美匹配 $X[i\,..\,j)$，并删除 $X[j\,..\,|X|)$。因此，$\mathsf{ed}^w(X, X[i\,..\,j)) \leq \mathsf{ed}^w_{\mathcal{A}}(X, X[i\,..\,j)) = \sum_{u \in [0\,..\,i) \cup [j\,..\,|X|)} w(X[u], \varepsilon) = \mathsf{ed}^w(X[0\,..\,i) \cdot X[j\,..\,|X|), \varepsilon)$。

现在，令 $Y = X[i\,..\,j)$，并考虑任意对齐 $\mathcal{B} \in \mathsf{A}(X, Y)$。对于每个 $u \in [0\,..\,i) \cup [j\,..\,|X|)$，我们递归地定义一个序列 $(c_{u,t})_{t=0}^{m_u}$，使得 $c_{u,0} = u$，对齐 $\mathcal{B}$与$X[c_{u,t}]$ 对齐至 $Y[c_{u,t+1} - i] = X[c_{u,t+1}]$（当 $t \in [0\,..\,m_u)$ 时），且 $\mathcal{B}$ 删除 $X[c_{u,m_u}]$。根据构造，序列 $(c_{u,t})_{t=0}^{m_u}$ 是有限的，并且 X 中的每个位置至多属于一个这样的序列。此外，由于 w 是拟度量，对于每个 $t \in [0\,..\,m_u)$，都有 $w(X[c_{u,t}], \varepsilon) \leq w(X[c_{u,t}], X[c_{u,t+1}]) + w(X[c_{u,t+1}], \varepsilon)$，因此 $w(X[c_{u,0}], \varepsilon) \leq w(X[c_{u,m_u}], \varepsilon) + \sum_{t \in [0\,..\,m_u)} w(X[c_{u,t}], X[c_{u,t+1}])$。由于

$\mathcal{B}$ 删除 $X[c_{u,m_u}]$，且 $X[c_{u,t}] \sim_{\mathcal{B}} Y[c_{u,t+1}-i] = X[c_{u,t+1}]$ 对 $t \in [0\,..\,m_u)$ 成立，因此得到

$$\begin{aligned}\mathsf{ed}^w_{\mathcal{B}}(X,Y) &\geq \sum_{u\in[0\,..\,i)\,..\,[j\cup|X|)} \left(w(X[c_{u,m_u}],\varepsilon) + \sum_{t\in[0\,..\,m_u)} w(X[c_{u,t}],X[c_{u,t+1}]) \right) \\ &\geq \sum_{u\in[0\,..\,i)\,..\,[j\cup|X|)} w(X[u],\varepsilon) \\ &= \mathsf{ed}^w(X[0\,..\,i)\cdot X[j\,..\,|X|),\varepsilon).\end{aligned}$$

由于 $\mathcal{B}$ 是任意选取的，我们得出结论：$\mathsf{ed}^w(X,Y) \geq \mathsf{ed}^w(X[0\,..\,i)\cdot X[j\,..\,|X|),\varepsilon)$。 □

参考文献

[Abb14] Amir Abboud。《简单问题的困难性》，2014年。发表于 ICALP 卫星研讨会（YR-ICALP）。URL：https://www.dropbox.com/s/jt9uzljjmormkb7/EasyHardness.pdf。[ABW18] Amir Abboud、Arturs Backurs 和 Virginia Vassilevska Williams。如果当前的团算法是最优的，那么 Valiant 解析器也是如此。《SIAM计算杂志》，47(6)：2527–2555，2018年。doi:10.1137/16M1061771。[AJ21] Shyan Akmal 和 Ce Jin。更快的有界树编辑距离算法。载于 Nikhil Bansal、Emanuela Merelli 和 James Worrell（编），《第48届国际自动机、语言与程序设计研讨会，ICALP 2021，2021年7月12日至16日，苏格兰格拉斯哥（线上会议）》，LIPIcs 第198卷，第12:1–12:15页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2021年。doi:10.4230/LIPIcs.ICALP.2021.12。[AKO10] Alexandr Andoni、Robert Krauthgamer 和 Krzysztof Onak。编辑距离与非对称查询复杂度的多对数近似。载于《2010年 IEEE 第51届计算机科学基础年会论文集》，FOCS '10，第377–386页，美国，2010年。IEEE计算机学会。doi:10.1109/FOCS.2010.43。[AN94] Arne Andersson 和 Stefan Nilsson。一种新的高效基数排序。载于《第35届计算机科学基础年会》，美国新墨西哥州圣菲，1994年11月20日至22日，第714–721页。IEEE计算机学会，1994年。doi:10.1109/SFCS.1994.365721。[AN20] Alexandr Andoni 和 Negev Shekel Nosatzki。近线性时间的编辑距离：这是一个常数因子。载于 Sandy Irani（编），《第61届 IEEE 计算机科学基础年会，FOCS 2020，美国北卡罗来纳州达勒姆，2020年11月16日至19日》，第990–1001页。IEEE，2020年。doi:10.1109/FOCS46700.2020.00096。[AO09] Alexandr Andoni 和 Krzysztof Onak。近线性时间内近似计算编辑距离。载于《第41届 ACM 计算理论研讨会论文集》，STOC '09，美国纽约州纽约市，2009年，第199–204页。计算机协会。doi:10.1145/1536414.1536444。[AP72] Alfred V Aho 和 Thomas G Peterson。面向上下文无关语言的最小距离纠错解析器。《SIAM计算杂志》，1(4)：305–312，1972年。

[BEG+21] Mahdi Boroujeni、Soheil Ehsani、Mohammad Ghodsi、MohammadTaghi Hajiaghayi 和 Saeed Seddighin。在真正亚二次时间内近似计算编辑距离：量子方法与 MapReduce。《《ACM 杂志》》，68(3)：19:1–19:41，2021年。doi:10.1145/3456807。[BES 06] Tu˘gkan Batu、Funda Ergun 和 Cenk Sahinalp。无意识字符串嵌入与编辑距离近似。载于《第17届 ACM-SIAM 离散算法研讨会论文集》，SODA '06，美国，2006年，第792–801页。工业与应用数学学会。doi:10.1145/1109557.1109644。[BGHS19] Mahdi Boroujeni、Mohammad Ghodsi、MohammadTaghi Hajiaghayi 和 Saeed Sed $1+\epsilon$dighin。在二次时间内近似树编辑距离。载于 Moses Charikar 和 Edith Cohen（编），《第51届 ACM SIGACT 计算理论年会论文集》，STOC 2019，美国亚利桑那州凤凰城，2019年6月23日至26日，第709–720页。ACM，2019年。doi:10.1145/3313276.3316388。[BGK03] Peter Buneman、Martin Grohe 和 Christoph Koch。压缩 XML 上的路径查询。载于《第29届超大型数据库国际会议论文集——第29卷》，VLDB '03，第141–152页。VLDB基金会，2003年。doi:10.1016/b978-012722442-8/50021-5。[BGMW20] Karl Bringmann、Pawe l Gawrychowski、Shay Mozes 和 Oren Weimann。树编辑距离无法在强亚立方时间内计算（除非全点对最短路径可以）。《ACM算法汇刊》，16(4)，2020年7月。doi:10.1145/3381878。[BGSW19] Karl Bringmann、Fabrizio Grandoni、Barna Saha 和 Virginia Vassilevska Williams。通过快速有界差分最小加乘积实现语言编辑距离与 RNA折叠的真正亚立方算法。《SIAM计算杂志》，48(2)：481–512，2019年。doi:10.1137/17M112720X。[BI18] Arturs Backurs 和 Piotr Indyk。编辑距离无法在强亚二次时间内计算（除非 SETH 不成立）。《SIAM计算杂志》，47(3)：1087–1097，2018年。doi:10.1137/15M1053128。[BII+17] Hideo Bannai、Tomohiro I、Shunsuke Inenaga、Yuto Nakashima、Masayuki Takeda 和 Kazuya Tsuruta。"runs"定理。《SIAM计算杂志》，46(5)：1501–1514，2017年。doi:10.1137/15m1011032。[Bil05] Philip Bille。树编辑距离及相关问题综述。《理论计算机科学》，337(1–3)：217–239，2005年6月。doi:10.1016/j.tcs.2004.12.030。[BO16] Arturs Backurs 和 Krzysztof Onak。快速解析错误较少的括号序列。载于 Tova Milo 和 Wang-Chiew Tan（编），《第35届 ACM SIGMOD-SIGACT-SIGAI 数据库系统原理研讨会》，PODS 2016，第477–488页。ACM，2016年。doi:10.1145/2902251.2902304。[BR20] Joshua Brakensiek 和 Aviad Rubinstein。在近线性时间内对近线性编辑距离进行常数因子近似。载于《第52届 ACM SIGACT 计算理论年会》，STOC 2020，第685–698页。ACM，2020年。doi:10.1145/3357713.3384282。[BS98] Horst Bunke 和 Kim Shearer。基于最大公共子图的图距离度量。《模式识别快报》，19(3–4)：255–259，1998年3月。doi:10.1016/S0167-8655(97)00179-7。

[BW08] Miko laj Boja´nczyk 和 Igor Walukiewicz。森林代数。载于 J¨org Flum、Erich Gr¨adel 和 Thomas Wilke（编），《逻辑与自动机：历史与展望——献给 Wolfgang Thomas》 [第2卷，逻辑与博弈文丛，第107–132页。阿姆斯特丹大学出版社，2008年。[BYJKK04] Ziv Bar-Yossef、T. S. Jayram、Robert Krauthgamer 和 Ravi Kumar。高效近似编辑距离。载于《第45届IEEE计算机科学基础年会论文集》，FOCS '04，第550–559页，美国，2004年。IEEE计算机学会。doi:10.1109/FOCS.2004.14。[CDG+20] Diptarka Chakraborty、Debarati Das、Elazar Goldenberg、Michal Kouck´y 和 Michael Saks。以真正的次二次时间将编辑距离近似到常数因子。《ACM 杂志》，67(6)，2020年10月。doi:10.1145/3422823。[CDX22] Shucheng Chi、Ran Duan 和 Tianle Xie。有界差分最小加乘积的更快算法。载于《第33届ACM-SIAM离散算法年会》，SODA 2022，第1435–1447页。工业与应用数学学会，2022年1月。doi:10.1137/1.9781611977073.60。[Cha99] Sudarshan S. Chawathe。在外存中比较层次数据。载于 Malcolm P. Atkinson、Maria E. Orlowska、Patrick Valduriez、Stanley B. Zdonik 和 Michael L. Brodie（编），VLDB'99，《第25届超大型数据库国际会议论文集》，1999年9月7日至10日，英国苏格兰爱丁堡，第90–101页。摩根·考夫曼出版社，1999年。网址：http://www.vldb.org/conf/1999/P8.pdf。[CIG22] Kateryna Chumachenko、Alexandros Iosifidis 和 Moncef Gabbouj。用于车牌国家代码识别的加权编辑距离。载于《第30届欧洲信号处理会议》，EUSIPCO 2022，第1111–1115页。IEEE，2022年。网址：https://ieeexplore.ieee.org/document/9909869。[DGH+22] Debarati Das、Jacob Gilbert、MohammadTaghi Hajiaghayi、Tomasz Kociumaka、Barna˜Saha 和 Hamed Saleh。用于有界树编辑距离的 $O(n + poly(k))$时间算法。载于《第63届IEEE计算机科学基础年会》，FOCS 2022。IEEE，2022年。arXiv:2209.07524。[DKS22] Debarati Das、Tomasz Kociumaka 和 Barna Saha。Dyck 编辑距离和 RNA 折叠的改进近似算法。载于 Mikolaj Bojanczyk、Emanuela Merelli 和 David P. Woodruff（编），《第49届自动机、语言与程序设计国际研讨会》，ICALP 2022，LIPIcs第229卷，第49:1–49:20页。Schloss Dagstuhl 信息学中心，2022年。doi:10.4230/LIPIcs.ICALP.2022.49。[DMRW10] Erik D. Demaine、Shay Mozes、Benjamin Rossman 和 Oren Weimann。树编辑距离的最优分解算法。《ACM 算法汇刊》，6(1)，2010年12月。doi:10.1145/1644015.1644017。[D¨ur22] Anita D¨urr。矩形单调最小加乘积的改进界限，2022年。doi:10.48550/arXiv.2208.02862。[FFF+16] Lionel Fontan、Isabelle Ferran´e、J´erˆome Farinas、Julien Pinquier 和 Xavier Aumont。使用音系加权莱文斯坦距离进行预测

微观层面的可懂度。载于国际语音通信协会第17届年会，Interspeech 2016，第650–654页。ISCA，2016年。doi:10.21437/Interspeech.2016-431。[FFM00] Martin Farach-Colton、Paolo Ferragina 和 S. Muthukrishnan。关于后缀树构造的排序复杂度。《ACM 杂志》，47(6):987–1011，2000年。doi:10.1145/355541.355547。[FGK+22a] Dvir Fried、Shay Golan、Tomasz Kociumaka、Tsvi Kopelowitz、Ely Porat 和 Tatiana Starikovskaya。 k-Dyck 编辑距离问题的改进算法。载于 Joseph (Seffi) Naor 和 Niv Buchbinder（编），《第33届ACM-SIAM离散算法年会》，SODA 2022，第3650–3669页。SIAM，2022年。doi:10.1137/1.9781611977073.144。[FGK+22b] Dvir Fried、Shay Golan、Tomasz Kociumaka、Tsvi Kopelowitz、Ely Porat 和 Tatiana Starikovskaya。 k-Dyck 编辑距离问题的改进算法，2022年。arXiv:2111.02336v2。[FLMM09] Paolo Ferragina、Fabrizio Luccio、Giovanni Manzini 和 S. Muthukrishnan。带标签树的压缩与索引及其应用。《ACM 杂志》，57(1)，2009年11月。doi:10.1145/1613676.1613680。[FW65] Nathan J. Fine 和 Herbert S. Wilf。周期函数的唯一性定理。《美国数学会会刊》，16(1):109–114，1965年。doi:10.2307/2034009。[GKS19] Elazar Goldenberg、Robert Krauthgamer 和 Barna Saha。间隙编辑距离的次线性算法。载于 David Zuckerman（编），《第60届IEEE计算机科学基础年会》，FOCS 2019，美国马里兰州巴尔的摩，2019年11月9日至12日，第1101–1120页。IEEE计算机学会，2019年。doi:10.1109/FOCS.2019.00070。[GRS20] Elazar Goldenberg、Aviad Rubinstein 和 Barna Saha。预处理是否有助于快速序列比较？载于《第52届ACM SIGACT 计算理论研讨会年会》，STOC 2020，第657–670页。ACM，2020年。doi:10.1145/3357713.3384300。[Gus97] Dan Gusfield。《字符串、树与序列算法：计算机科学与计算生物学》。剑桥大学出版社，美国，1997年。doi:10.1017/cbo9780511574931。[GWK21] Andrew Gerlach、Adam Wiemerslage 和 Katharina Kann。使用加权编辑距离进行范式聚类。载于第18届SIGMORPHON 语音学、音系学与形态学计算研究研讨会，第107–114页。计算语言学协会，2021年。doi:10.18653/v1/2021.sigmorphon-1.12。[Har78] Michael A. Harrison。《形式语言理论导论》。Addison-Wesley Longman 出版公司，美国马萨诸塞州波士顿，第一版，1978年。[HRS19] Bernhard Haeupler、Aviad Rubinstein 和 Amirbehshad Shahrasbi。近线性时间插入删除码，以及通过索引近似编辑距离至$1+\epsilon$。载于《第51届ACM SIGACT 计算理论研讨会年会论文集》，STOC 2019，第697–708页，美国纽约州纽约市，2019年。计算机协会。doi:10.1145/3313276.3316371。

[HT84] Dov Harel 和 Robert Endre Tarjan. 查找最近公共祖先节点的快速算法. SIAM J. Comput., 13(2):338–355, 1984年5月. doi:10.1137/0213024.[Ind01] P. Indyk. 低失真几何嵌入的算法应用. 收录于第42届IEEE计算机科学基础研讨会论文集，FOCS '01，第10页，美国，2001年. IEEE计算机学会. doi:10.1109/sfcs.2001.959878.[JM09] Dan Jurafsky 和 James H. Martin. 语音与语言处理：自然语言处理、计算语言学和语音识别导论，第2版. 人工智能系列丛书. Prentice Hall, Pearson Education International, 2009. URL: https://www.worldcat.org/oclc/315913020.[Kle98] Philip N. Klein. 无根有序树之间编辑距离的计算. 收录于第6届欧洲算法研讨会论文集，ESA '98，第91–102页，柏林，海德堡，1998年. Springer-Verlag. doi:10.1007/3-540-68530-8_8.[Koz97] Dexter C. Kozen. 自动机与可计算性. Springer New York, 1997. doi:10.1007/978-1-4612-1844-9.[KRRW15] Tomasz Kociumaka, Jakub Radoszewski, Wojciech Rytter 和 Tomasz Wale´n. 文本中的内部模式匹配查询及其应用. 收录于 Piotr Indyk 主编的第 26 届 ACM-SIAM 离散算法研讨会，SODA 2015，第 532–551 页 . SIAM, 2015. doi:10.1137/1.9781611973730.36.[KS20a] Tomasz Kociumaka 和 Barna Saha. 用于计算和嵌入间隙编辑距离的次线性时间算法. 收录于 Sandy Irani 主编的第61届IEEE计算机科学基础年会，FOCS 2020，美国北卡罗来纳州达勒姆，2020年11月16日至19日，第1168–1179页. IEEE, 2020. doi:10.1109/FOCS46700.2020.00112.[KS20b] Michal Kouck´y 和 Michael E. Saks. 近线性时间内对远距离输入对的编辑距离进行常数因子近似. 收录于第52届 ACM SIGACT 计算理论研讨会，STOC 2020，第699–712页. ACM, 2020. doi:10.1145/3357713.3384307.[Kur96] Stefan Kurtz. 加权编辑距离下的近似字符串搜索. 收录于第3届南美字符串处理研讨会，WSP 1996，第156–170页. 卡尔顿大学出版社，1996年.[Kus19] William Kuszmaul. 强亚二次时间的动态时间规整：低距离区间的算法与近似计算. 收录于 Christel Baier、Ioannis Chatzigiannakis、Paola Flocchini 和 Stefano Leonardi 主编的第46届国际自动机、语言与程序设计研讨会，ICALP 2019，第132卷 LIPIcs，第80:1–80:15页. Schloss Dagstuhl - 莱布尼茨信息学中心，2019年. doi:10.4230/LIPIcs.ICALP.2019.80.[KXI20] Satoshi Koide、Chuan Xiao 和 Yoshiharu Ishikawa. 加权编辑距离约束下道路网络中的快速子轨迹相似性搜索. VLDB基金会会刊，13(12):2188–2201，2020年. doi:10.14778/3407790.3407818.[Lev65] Vladimir I. Levenshtein. 可纠正删除、插入和反转的二进制码. Soviet physics. Doklady, 10:707–710, 1965.

[LMS98] Gad M. Landau、Eugene W. Myers 和 Jeanette P. Schmidt. 增量式字符串比较. SIAM J. Comput., 27(2):557–582, 1998年4月. doi:10.1137/s0097539794264810.[LV88] Gad M. Landau 和 Uzi Vishkin. 使用 k 差异进行快速字符串匹配. Journal of Computer and System Sciences, 37(1):63–78, 1988年. doi:10.1016/0022-0000(88)90045-1.[Mao21] Xiao Mao. 突破（无权）树编辑距离的三次时间障碍. 收录于第62届IEEE计算机科学基础年会，FOCS 2021，美国科罗拉多州丹佛，2022年2月7日至10日，第792–803页. IEEE, 2021. doi:10.1109/FOCS52979.2021.00082.[Mye86] Eugene W. Myers. 一种 O(ND)差异算法及其变体. Algorithmica, 1(2):251–266, 1986 年 . doi:10.1007/BF01840446.[Mye95] Gene Myers. 上下文无关语言的近似匹配. 信息处理快报，54(2):85–92, 1995年. doi:10.1016/0020-0190(95)00007-y.[NW70] Saul B. Needleman 和 Christian D. Wunsch. 一种适用于搜索两种蛋白质氨基酸序列相似性的一般方法. 分子生物学杂志，48(3):443–453, 1970年3月. doi:10.1016/0022-2836(70)90057-4.[PM02] Guillermo Peris 和 Andr´es Marzal. 分类中的加权编辑代价快速循环编辑距离计算. 收录于第16届国际模式识别会议，ICPR 2002，第184–187页. IEEE, IEEE计算机学会，2002年. doi:10.1109/ICPR.2002.1047428.[PR05] Seth Pettie 和 Vijaya Ramachandran. 实数加权无向图的最短路径算法. SIAM计算杂志，34(6):1398–1431, 2005年. doi:10.1137/s0097539702419650.[PT87] Robert Paige 和 Robert Endre Tarjan. 三种划分细化算法. SIAM J. Comput., 16(6):973–989, 1987年. doi:10.1137/0216062.[Sah14] Barna Saha. 近线性时间的 Dyck 语言编辑距离问题. 收录于第55届IEEE计算机科学基础年会，FOCS 2014，第611–620页. IEEE计算机学会，2014年. doi:10.1109/focs.2014.71.[Sah17] Barna Saha. 语言编辑距离和 RNA折叠的快速且节省空间的近似算法：一种遗忘式动态规划方法. 收录于2017年IEEE第58届计算机科学基础年会（FOCS），第295–306页. IEEE，2017年.[Sel77] Stanley M. Selkow. 树到树编辑问题. 信息处理快报，6(6):184–186, 1977年. doi:10.1016/0020-0190(77)90064-3.[Ski20] Steven Skiena. 算法设计手册，第三版. 计算机科学丛书. Springer, 2020. doi:10.1007/978-3-030-54256-6.[SS22] Masoud Seddighin 和 Saeed Seddighin. $3+\epsilon$ 真正次二次时间内树编辑距离的近似. 收录于 Mark Braverman 主编的第13届理论计算机科学创新会议，ITCS 2022，2022年1月31日至2月3日，

美国加利福尼亚州伯克利，LIPIcs 第 215 卷，第 115:1–115:22 页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2022 年。doi:10.4230/LIPIcs.ITCS.2022.115.[SZ90] Bruce A. Shapiro 和 Kaizhong Zhang。使用树比较多个 RNA 二级结构。Comput. Appl. Biosci., 6(4):309–318，1990 年。doi:10.1093/bioinformatics/6.4.309.[Tai79] Kuo-Chung Tai。树到树校正问题。《ACM 杂志》，26(3):422–433，1979 年 7 月。doi:10.1145/322139.322143.[Tou05] H´el`ene Touzet。用于相似有序树的线性树编辑距离算法。载于第 16 届年度组合模式匹配会议论文集，CPM'05，第 334–345 页，柏林、海德堡，2005 年。Springer-Verlag。doi:10.1007/11496656_29.[WF74] Robert A. Wagner 和 Michael J. Fischer。字符串到字符串校正问题。《ACM 杂志》，21(1):168–173，1974 年。doi:10.1145/321796.321811.[ZS89] Kaizhong Zhang 和 Dennis E. Shasha。树之间编辑距离及相关问题的简单快速算法。SIAM J. Comput., 18(6):1245–1262，1989 年。doi:10.1137/0218082。