Computer Science and Application 计算机科学与应用, 2023, 13(2), 236-250
Published Online February 2023 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2023.132024

Hans汉斯

基于生物信息学中DNA分子序列模式匹配算法研究实现

陈亭宇，尹国才，魏国晟

北华航天工业学院计算机学院，河北 廊坊

收稿日期：2023年1月17日；录用日期：2023年2月15日；发布日期：2023年2月22日

摘　要

生物信息学是融合先进的生物科学和计算机技术的一门综合运用数学、信息科学、计算机技术等对生物学、医学的信息进行科学的组织、整理和归纳的科学。DNA分子序列比对是生物信息学中最重要和最基础的研究方向之一，是探究基因与疾病关系的重要手段。本文研究的主要目标是在不确定的分子序列数据中找到所有与目标序列相同且出现概率大于给定阈值的序列，并给出目标序列总数及每个目标序列的起始位点。本文针对现有基于“空间换时间”的分子序列模式匹配算法仅限于次数的计算以及基于生物信息学中双DNA序列比对算法的图像立体匹配方法对于不确定的源数据具有局限性的问题，提出了一种基于加权后缀树的DNA分子序列模式匹配算法。该方法应用加权后缀树为主要数据结构，改进了不确定的源数据的匹配准确度，解决了map数据结构仅限于次数计算的问题，实验结果表明，本文提出的算法在匹配速度及灵敏度上有了一定的提高。

关键词

生物信息学，分子序列，模式匹配，算法

Research and Implementation of DNA Molecular Sequence Pattern Matching Algorithm Based on Bioinformatics

Tingyu Chen, Guocai Yin, Guosheng Wei

School of Computer Science and Engineering, North China Institute of Aerospace Engineering, Langfang Hebei

Received: Jan. 17th, 2023; accepted: Feb. 15th, 2023; published: Feb. 22nd, 2023

文章引用: 陈亭宇, 尹国才, 魏国晟. 基于生物信息学中 DNA 分子序列模式匹配算法研究实现[J]. 计算机科学与应用, 2023, 13(2): 236-250. DOI: 10.12677/csa.2023.132024

Abstract

Bioinformatics is a science that integrates advanced biological science and computer technology. It integrates mathematics, information science and computer technology to scientifically organize, sort out and conclude the information of biology and medicine. DNA sequence alignment is one of the most important and basic research directions in bioinformatics and an important means to explore the relationship between genes and diseases. The main objective of this paper is to find all sequences that are identical to the target sequence and whose occurrence probability is greater than the given threshold in the uncertain molecular sequence data and to give the total number of target sequences and the starting site of each target sequence. In this paper, a weighted suffix tree-based DNA sequence pattern matching algorithm is proposed to solve the problem that the existing molecular sequence pattern matching algorithm based on "space for time" is limited to the calculation of times, and the image stereo matching method based on the double DNA sequence alignment algorithm in bioinformatics is limited to uncertain source data. This method uses weighted suffix trees as the main data structure, improves the matching accuracy of uncertain source data, and solves the problem that map data structure is limited to number calculation. Experimental results show that the proposed algorithm has improved the matching speed and sensitivity to a certain extent.

Keywords

Bioinformatics, Molecular Sequences, Pattern Matching, Algorithm

1. 引言

生物信息学是融合先进的生物科学和计算机技术的一门综合运用数学、信息科学、计算机技术等对生物学、医学的信息进行科学的组织、整理和归纳的科学[1]。生物信息学的研究重点主要体现在基因组学和蛋白质学两方面，从核酸和蛋白质序列出发，分析序列中表达结构和功能的生物信息。近年来，随着高通量测序(NGS)技术的快速发展[2]，组学数据和表观遗传学数据呈指数增长[3]，海量的生物数据背后蕴含了大量的生物规律和知识。DNA/RNA 甲基化修饰位点识别[4]-[9]、增强子识别[10] [11]、RNA 序列编辑位点识别[12]、蛋白质功能位点识别[13]、DNA 结合蛋白识别[14]等本质上都是多分类或二分类任务。DNA 分子序列比对是生物信息学中最重要和最基础的研究方向之一，是探究基因与疾病关系的重要手段[15]，例如判断人体内是否存在致癌致病的关键 DNA 分子序列以及其存在的概率，通过概率大小判断是否需要提前介入性治疗等[16] [17]。

近年来，采用数据挖掘算法进行 DNA 分子序列模式匹配分析已成为生物信息学领域的研究热点[18] [19] [20]。目前基于 DNA 分子序列模式匹配研究的方法大致分为以下几类：一是，一种基于“空间换时间”的模式匹配算法[21]，以 map 数据结构来存储中间结果，使得扫描 DNA 序列一次即可同时计算所有元组模式在该序列中出现的次数。该算法通过分析 DNA 序列特征计算过程中的特殊性，提升了 DNA 序列模式特征计算的效率，但仅限于次数的计算，具有局限性。二是，一种基于生物信息学中双 DNA 序列

比对算法的图像立体匹配方法，但此种算法对于不确定的源数据具有局限性。

基于此，本文针对现有基于“空间换时间”的分子序列模式匹配算法仅限于次数的计算以及基于生物信息学中双 DNA 序列比对算法的图像立体匹配方法对于不确定的源数据具有局限性的问题，提出了一种基于加权后缀树的 DNA 分子序列模式匹配算法。该方法应用加权后缀树为主要数据结构，保留了顺序遍历的优点，克服了无用序列遍历的缺点，改进了不确定的源数据的匹配准确度，解决了 map 数据结构仅限于次数计算的问题，在匹配速度及灵敏度上有了一定的提高。

2. 算法

DNA 序列是由 A (腺嘌呤碱基)、T (胸腺嘧啶碱基)、C (胞嘧啶碱基)、G (鸟嘌呤碱基) 4 种碱基字符随意组合而成的字符串。在某些情况下，分子加权序列可以模拟调控蛋白结合位点的过程。由于 DNA 蛋白质复合物的特性，每个位置上个体碱基出现概率的总和与 DNA 蛋白质复合物的稳定性有关。如果总和大于给定范围，则 DNA 蛋白质复合物将被视为足够稳定的复合物。本文研究的主要目标是在不确定的分子序列数据中找到所有与目标序列 P 相同且出现概率大于给定阈值 1/K 的序列，并给出目标序列总数及每个目标序列的起始位点。由于后缀树是序列分析中重要且高效的数据结构[22]，因此本文选择后缀树和加权后缀树作为算法研究实现的主要数据结构。在处理分子加权序列时，需要找到给定范围内的所有目标序列，但是直接为文本中的每个位置构建后缀树会带来巨大的工作量，占用巨大的内存。由于不确定性和数据量巨大，为了提高效率，需要一定的阈值 1/K，所有概率高于此阈值的对象将被保留。

2.1. 加权序列

加权序列 $X = X[1]\cdots X[n]$ 是一个位置序列，其中每个位置 $X[i]$ 由一组有序对 $(\sigma, \pi_i(\sigma))$ 组成，$\pi_i(\sigma)$ 是在位置 i 有字符 σ 的概率。对于每一个位置 $X[i]$，$1 \le i \le n$，$\sum_{\forall\sigma} \pi_i(\sigma) = 1$。

例如，DNA 分子序列字母表设为$\sum$ = {A, C, G, T}，则图 1 中的加权序列可视为一串 8 个字母的字符串。位置 1 的字母可以是出现概率为 0.4 的 A、出现概率为 0.6 的 C；位置 2、3、4、5 的字母分别是 T、T、G、C；位置 6 的字母可以是 C、G、T，出现概率分别为 0.2、0.1、0.7；位置 7 和 8 的字母分别为 G 和 C。ω 代表由一个或多个$\sum$中字符组成的长度为 n 的序列。$\omega[1\cdots n] = \omega[1], \omega[2], \cdots, \omega[n]$，$\omega[i] \in \sum, 1 \le i \le n$。

序列 ω

位点	1	2	3	4	5	6	7	8
	A, 0.4	T	T	G	C	A, 0	G	C
	C, 0.6					C, 0.2		
	G, 0					G, 0.1		
	T, 0					T, 0.7		

Figure 1. An example of a weighted sequence with two weighted positions
图 1. 具有两个加权位置的加权序列实例

由图 1 可以产生下列字符串：ω_1 = ATTGCCGC，ω_2 = ATTGCGGC，ω_3 = ATTGCTGC，ω_4 = CTTGCCGC，ω_5 = CTTGCGGC，ω_6 = CTTGCTGC，等等。从位置 i 开始的字符串 ω 的出现概率是每个字符出现概率的乘积。对于上述实例，我们可以得到：

$$\pi(\omega_1)=\pi_1(\mathrm{A})*\pi_2(\mathrm{T})*\pi_3(\mathrm{T})*\pi_4(\mathrm{G})*\pi_5(\mathrm{C})*\pi_6(\mathrm{C})*\pi_7(\mathrm{G})*\pi_8(\mathrm{C})=0.4*0.2=0.08$$
$$\pi(\omega_2)=\pi_1(\mathrm{A})*\pi_2(\mathrm{T})*\pi_3(\mathrm{T})*\pi_4(\mathrm{G})*\pi_5(\mathrm{C})*\pi_6(\mathrm{G})*\pi_7(\mathrm{G})*\pi_8(\mathrm{C})=0.4*0.1=0.04$$
$$\pi(\omega_3)=\pi_1(\mathrm{A})*\pi_2(\mathrm{T})*\pi_3(\mathrm{T})*\pi_4(\mathrm{G})*\pi_5(\mathrm{C})*\pi_6(\mathrm{T})*\pi_7(\mathrm{G})*\pi_8(\mathrm{C})=0.4*0.7=0.28$$
$$\cdots$$
$$\pi(\omega_i)=\pi_1(\mathrm{A})*\pi_2(\mathrm{T})*\pi_3(\mathrm{T})*\pi_4(\mathrm{G})*\pi_5(\mathrm{C})*\pi_6(\mathrm{A})*\pi_7(\mathrm{G})*\pi_8(\mathrm{C})=0.4*0=0$$

2.2. 后缀树

后缀树是一种重要的数据结构，对于处理字符串问题十分有效[22]。用 $T(\omega)$ 表示字符串 ω 的后缀树。设 $L(v)$ 表示 $T(\omega)$ 中从根到节点 v 的路径标签。当 $L(v)=\omega[i\cdots n]$ 时， $T(\omega)$ 的叶节点 v 被标记为 i。将 v 的叶列表 $LL(v)$ 定义为 v 节点下面子树中的叶标签列表。后缀树支持 ω 上的快速精确字符串匹配：模式 P 从位置 i 开始出现当且仅当 P 是 ω_i 的前缀，则可以在 $\omega[i\cdots n]$ 中通过前缀搜索找到 P 的所有出现。由于字符串 ω 的最后一个字母为 $\$\notin\sum$，则没有一个后缀是另一个后缀的前缀，并且后缀树中的所有叶节点都是代表不同后缀的节点。示例：Let ω = banana\$。如图 2 后缀树 $T(\omega)$ 所示，顶部节点是这个后缀树的根。节点 0~6 表示字符串 ω 的所有后缀。

节点 0：banana\$
节点 1：anana\$
节点 2：nana\$
节点 3：ana\$
节点 4：na\$
节点 5：a\$
节点 6：\$

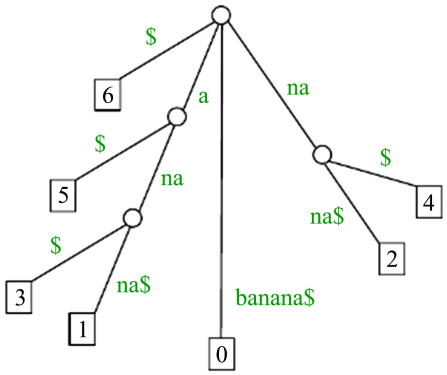

Figure 2. Suffix tree $T(\omega)$
图 2. 后缀树 $T(\omega)$

2.3. 加权后缀树

加权后缀树是本算法使用的主要数据结构，用于存储具有一定出现概率的加权序列对应的后缀。由于这些属性，加权后缀树被认为是具有存在概率概念的广义后缀树。本文中使用加权后缀树 *Weighted Suffix Tree* (WST)存储所有出现概率大于给定阈值 1/K 的目标后缀，然后求出给定模式 P 在可实现的最优时间内的所有出现次数及位点[23]。

X 是一个加权序列，给从位置 i 开始的每个后缀定义一个可能的加权子字符串列表，并且每个后缀出

现的概率大于给定阈值 1/K。$X_{i,j}$ 表示从位置 i 开始的后缀，j 表示从位置 i 开始的第 j 个加权子字符串。$WST(X)$ 表示加权序列 X 的加权后缀树。$L(v)$ 代表 $WST(X)$ 中节点 v 的路径标签。$LL(v)$ 表示 v 节点下面子树中的叶标签列表。为清楚地解释上述概念，举例找出 $X_{i,j}$ 中所有 $\pi(X_{i,j}) \geq 1/4$ 的分子序列，如图 3 所示。

1) $X[1\cdots 8]$ 的前缀：

$$X_{1,1} = \underline{\mathrm{A}}\mathrm{TTGC}\underline{\mathrm{C}}\mathrm{GC}, \pi(X_{1,1}) = 0.08;$$

$$X_{1,2} = \underline{\mathrm{A}}\mathrm{TTGC}\underline{\mathrm{G}}\mathrm{GC}, \pi(X_{1,2}) = 0.04;$$

$$X_{1,3} = \underline{\mathrm{A}}\mathrm{TTGC}\underline{\mathrm{T}}\mathrm{GC}, \pi(X_{1,3}) = 0.28;$$

$$X_{1,4} = \underline{\mathrm{C}}\mathrm{TTGC}\underline{\mathrm{C}}\mathrm{GC}, \pi(X_{1,4}) = 0.12;$$

$$X_{1,5} = \underline{\mathrm{C}}\mathrm{TTGC}\underline{\mathrm{G}}\mathrm{GC}, \pi(X_{1,5}) = 0.06;$$

$$X_{1,6} = \underline{\mathrm{C}}\mathrm{TTGC}\underline{\mathrm{T}}\mathrm{GC}, \pi(X_{1,6}) = 0.42.$$

2) $X[2\cdots 8]$ 的前缀：

$$X_{2,1} = \mathrm{TTGC}\underline{\mathrm{C}}\mathrm{GC}, \pi(X_{2,1}) = 0.2;$$

$$X_{2,2} = \mathrm{TTGC}\underline{\mathrm{G}}\mathrm{GC}, \pi(X_{2,2}) = 0.1;$$

$$X_{2,3} = \mathrm{TTGC}\underline{\mathrm{T}}\mathrm{GC}, \pi(X_{2,3}) = 0.7.$$

3) $X[3\cdots 8]$ 的前缀：

$$X_{3,1} = \mathrm{TGC}\underline{\mathrm{C}}\mathrm{GC}, \pi(X_{3,1}) = 0.2;$$

$$X_{3,2} = \mathrm{TGC}\underline{\mathrm{G}}\mathrm{GC}, \pi(X_{3,2}) = 0.1;$$

$$X_{3,3} = \mathrm{TGC}\underline{\mathrm{T}}\mathrm{GC}, \pi(X_{3,3}) = 0.7.$$

4) $X[4\cdots 8]$ 的前缀：

$$X_{4,1} = \mathrm{GC}\underline{\mathrm{C}}\mathrm{GC}, \pi(X_{4,1}) = 0.2;$$

$$X_{4,2} = \mathrm{GC}\underline{\mathrm{G}}\mathrm{GC}, \pi(X_{4,2}) = 0.1;$$

$$X_{4,3} = \mathrm{GC}\underline{\mathrm{T}}\mathrm{GC}, \pi(X_{4,3}) = 0.7.$$

5) $X[5\cdots 8]$ 的前缀：

$$X_{5,1} = \mathrm{C}\underline{\mathrm{C}}\mathrm{GC}, \pi(X_{5,1}) = 0.2;$$

$$X_{5,2} = \mathrm{C}\underline{\mathrm{G}}\mathrm{GC}, \pi(X_{5,2}) = 0.1;$$

$$X_{5,3} = \mathrm{C}\underline{\mathrm{T}}\mathrm{GC}, \pi(X_{5,3}) = 0.7.$$

6) $X[6\cdots 8]$ 的前缀：

$$X_{6,1} = \underline{\mathrm{C}}\mathrm{GC}, \pi(X_{6,1}) = 0.2;$$

$$X_{6,2}=\underline{G}GC,\ \pi\left(X_{6,2}\right)=0.1;$$

$$X_{6,3}=\underline{T}GC,\ \pi\left(X_{6,3}\right)=0.7.$$

7) $X[7\cdots 8]$的前缀：

$$X_{7,1}=GC,\ \pi\left(X_{7,1}\right)=1;$$

8) $X[8]$的前缀：

$$X_{8,1}=C,\ \pi\left(X_{8,1}\right)=1;$$

挑选出现概率大于等于 1/4 的序列：

$$X_{1,3}=\underline{A}TTGC\underline{T}GC\$$$

$$X_{1,6}=\underline{C}TTGC\underline{T}GC\$$$

$$X_{2,3}=TTGC\underline{T}GC\$$$

$$X_{3,3}=TGC\underline{T}GC\$$$

$$X_{4,3}=GC\underline{T}GC\$$$

$$X_{5,3}=C\underline{T}GC\$$$

$$X_{6,3}=\underline{T}GC\$$$

$$X_{7,1}=GC\$$$

$$X_{8,1}=C\$$$

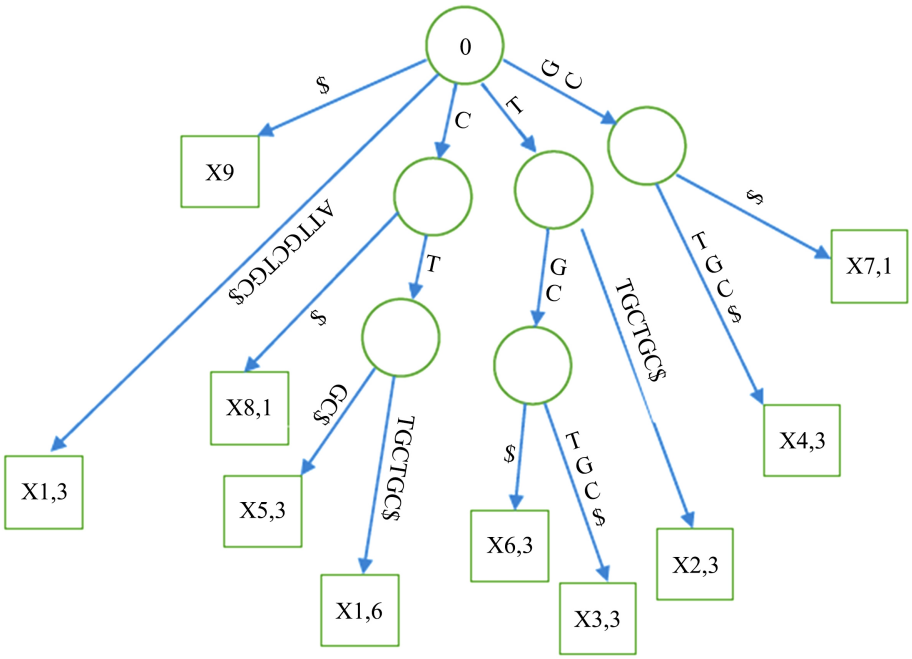

Figure 3. An example of a weighted suffix tree
图 3. 加权后缀树实例

如前所述，为所有出现概率大于等于给定阈值 1/K 的子字符串建立加权后缀树是本算法的核心，有效地建立加权后缀树主要包括三个步骤：着色，生成和构造[23]。

2.4. 着色

对于给定的加权序列 X，从左到右扫描每一个位置 i，$(1 \le i \le n)$，并按如下规则求值：情况 1：如果位置 i 所列的所有字母出现的概率都不大于阈值 1-1/K，则位置 i 着色为黑色；情况 2：如果存在一个字母，其出现在位置 i 的概率大于阈值 1/K，则将位置 i 着色为灰色；情况 3：如果位置 i 存在一个字母，其出现概率为 1，则位置 i 着色为白色。

白色的位置因为只有一个字母出现的概率等于 1，没有其他选择，因此被认为是足够稳定的位置；黑色的位置没有出现概率大于 1-1/K 的字母，而会有多于一个出现概率大于 1/K 的字母，因此将黑色的位置视为分支位置；对于灰色的位置，所有可能出现的字母出现的概率都小于 1/K，只有一个特殊字母出现的概率大于 1-1/K，这意味着最后只有一种选择。在着色阶段，为加权序列 X 的第一个位置以及所有黑色的位置生成一个队列 B。同时创建一个向量 COLOR 来存储 X 的每个位置的颜色。示例：长度 $n = 10$ 的加权序列 X 如表 1 所示。

Table 1. Weighted sequence X
表 1. 加权序列 X

	0 (A)	1 (C)	2 (G)	3 (T)
0	1	0	0	0
1	0	1	0	0
2	0.5	0.5	0	0
3	1	0	0	0
4	0.3	0.3	0.3	0.1
5	0	0	0	1
6	0.4	0.6	0	0
7	0.2	0	0.8	0
8	0.5	0	0.5	0
9	0	0.1	0.9	0

设置阈值 K = 4，1-1/K = 0.75。位置 0 有字母 A 的出现概率等于 1，因此设置 COLOR[0] = 0 (白色)。同理，位置 1, 3, 5 也是白色位置；位置 2 有一个概率为 0.5 的字母 A 和一个概率相同的字母 C，没有一个概率大于 1-1/K = 0.75 或等于 1。因此，设置 COLOR[2] = 2 (黑色)。同理，位置 4, 6, 8 也是黑色位置；位置 7 有一个概率为 0.2 的字母 A 和一个概率为 0.8 的字母 G，大于 1-1/K = 0.75，因此设置 COLOR[7] = 1 (灰色)。同理，位置 9 也是灰色位置。这样，可以得到完整的向量 COLOR[ ] = {0, 0, 2, 0, 2, 2, 1, 2, 1} 和队列 B = {0, 2, 4, 6, 8}。

2.5. 生成

生成阶段的输入是加权序列 X、队列 B 和来自着色阶段的向量 COLOR。这一阶段的主要目的是生成存在概率大于 1/K 的所有子字符串。首先，从左到右扫描队列 B 中的每个位置，然后从每个黑色位置 i 开始生成若干可能的子字符串。具体的生成方法如下：第一，从黑色位置 i 开始，向右移动；第二，找到与前一个字符相同的单个字符，并将其添加到当前未完成的子字符串的末尾；第三，当遇到一个灰色或白色的位置时，意味着只有一个选择，继续添加它直到可以完成当前的子字符串；第四，尝试下一个可能的字母，并在这个黑色的位置创建一个新的子字符串，直到生成所有可能的子字符串；第五，从下一个黑色位置开始，重复以上步骤。该过程如图 4 所示。

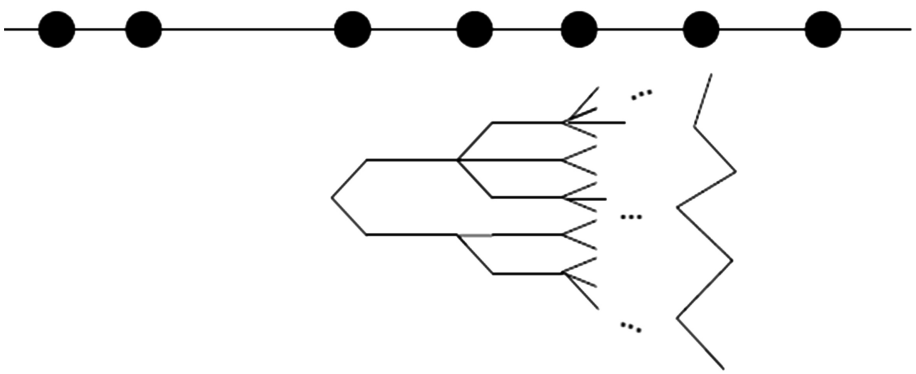

Figure 4. The process of creating all possible substrings for the black position i
图 4. 为黑色位置 i 创造所有可能的子字符串的过程

在这个阶段为每个生成的子字符串定义两个概率 π'和 π''。π'存储的是真正的子字符串的概率，而 π''存储的是将灰色位置视为白色位置时的概率。当满足以下条件时，生成过程停止：首先，当第一次遇到一个黑色位置并且 π''已经小于 1/K，记录下来；其次，如果所记录位置的下一个位置为黑色，则停止生成，并将所记录的位置标记为扩展位置。否则，如果从所记录的位置开始的下一个位置是白色或灰色位置，只需将白色或灰色的位置相加，直到再次遇到黑色的位置，最后一个白色或灰色的位置标记为扩展位置，如图 5 所示。由于 π'可能提前满足阈值 1/K，则实际子字符串可能比扩展子字符串短，因此定义了每个产生子字符串的真实结束位置与扩展结束位置之间的距离差 D。

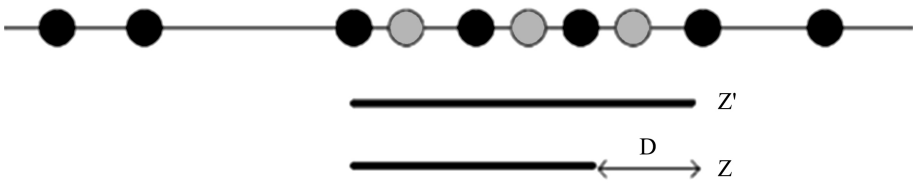

Figure 5. Extended substring Z' and actual substring Z
图 5. 扩展子字符串 Z'及实际子字符串 Z

示例：使用加权序列 X (见表 1)、向量 COLOR[ ] ={0, 0, 2, 0, 2, 2, 1, 2, 1}和着色阶段产生的队列 B = {0, 2, 4, 6, 8}来生成因子。首先，从左到右扫描队列 B，得到第一个位置 0。其次，为 0 号位置创建空的树，从 0 号位置开始扩展子字符串(如图 6 树 0 所示)。将所有的树(分别见图 6~9 和图 10)添加到队列 LT 中，将在构造阶段使用。

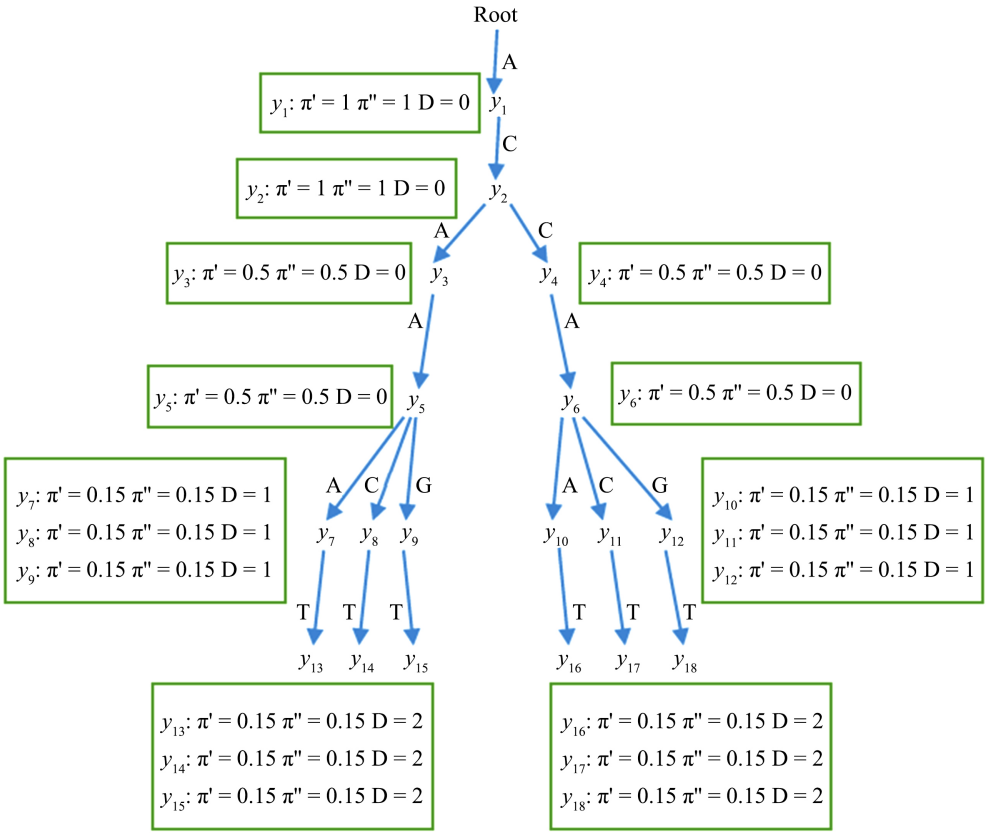

Figure 6. Tree 0
图 6. 树 0

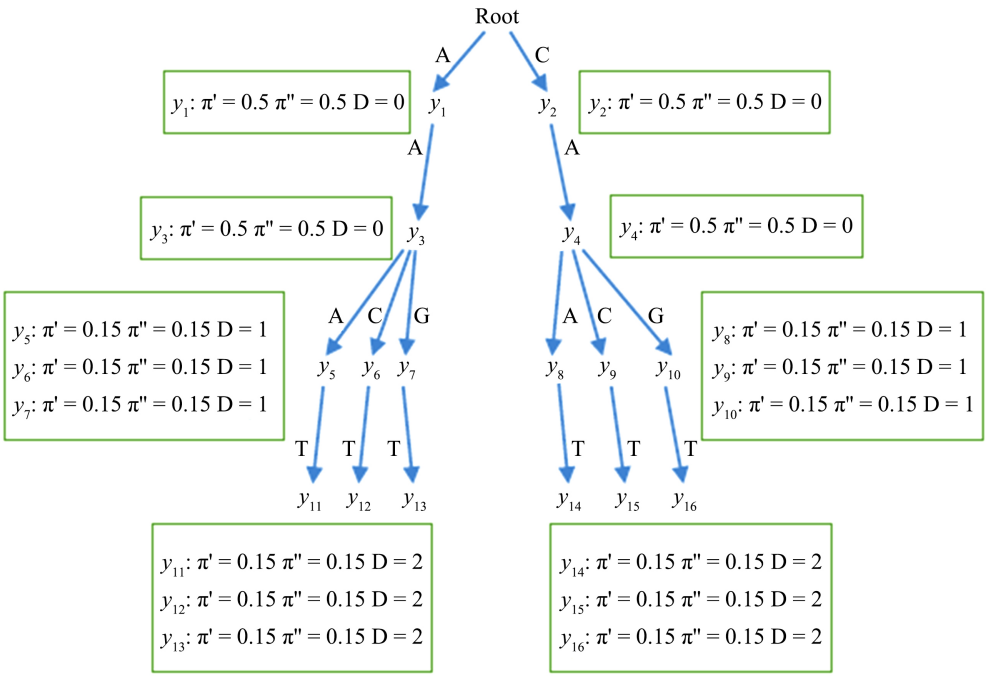

Figure 7. Tree 2
图 7. 树 2

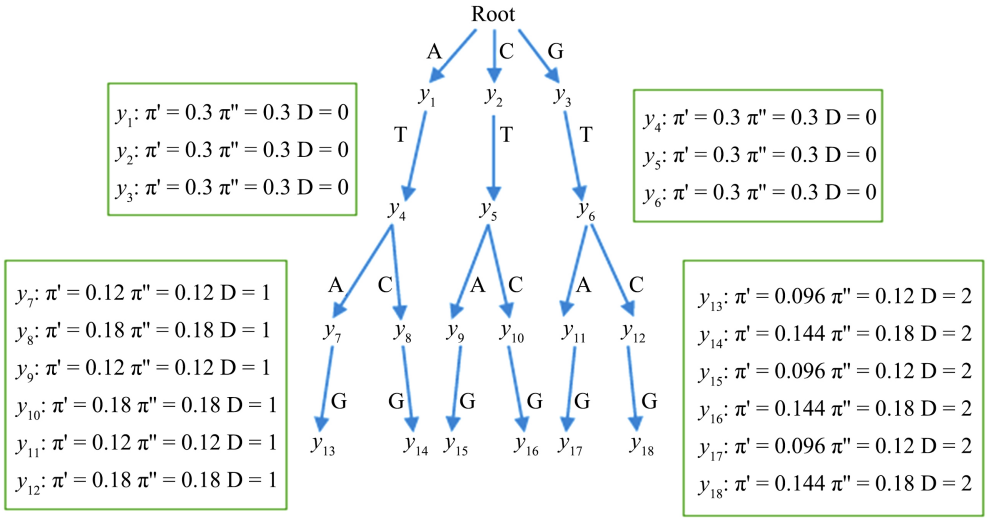

Figure 8. Tree 4
图 8. 树 4

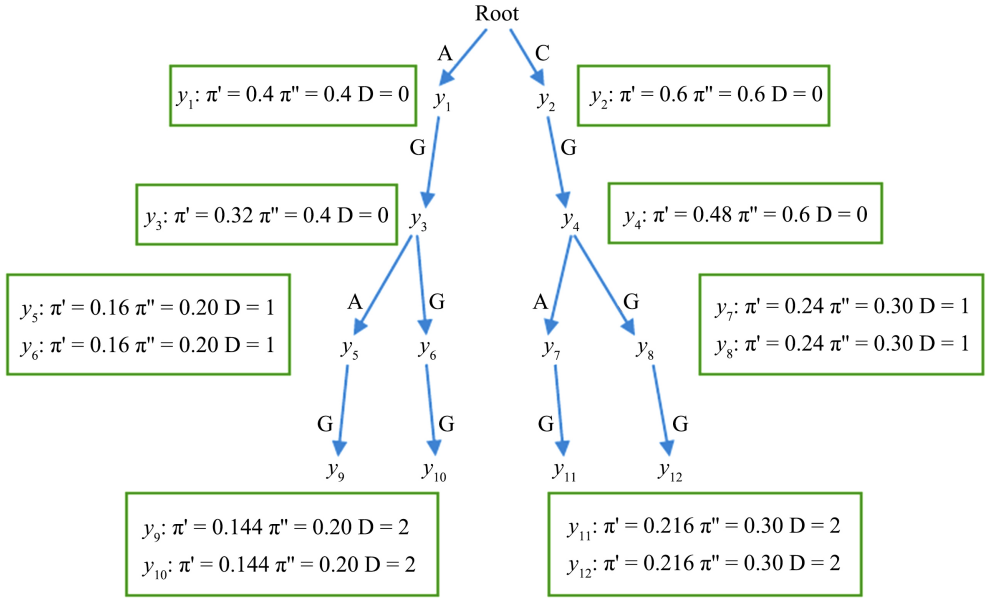

Figure 9. Tree 6
图 9. 树 6

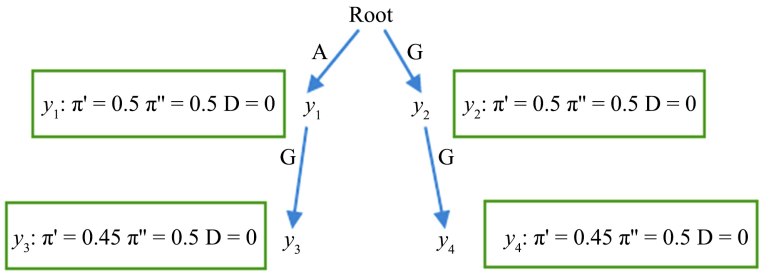

Figure 10. Tree 8
图 10. 树 8

2.6. 构造

从每个黑色位置生成所有可能的子字符串后，在广义后缀树中插入每个扩展子字符串，然后从树中删除扩展字符。示例：使用加权序列 X (见表 1)、着色阶段生成的向量 COLOR[] = {0, 0, 2, 0, 2, 1, 2, 1}和生成阶段生成的队列 LT 来构建后缀树。首先，从左向右扫描队列 LT，可以得到树 0 (如图 6 所示)。其次，对于每个树中的所有叶子，得到每个叶子的 Z'路径标签，并将 Z'插入加权后缀树中，同时插入 Z'的后缀，并计算相应的 Dj，如表 2 所示。继续插入子字符串并删除冗余部分，直到所有树的所有叶子都被检查完毕，最后我们可以得到完整的加权后缀树，如图 11 所示。

Table 2. Construction process of tree 0
表 2. 树 0 的构造过程

叶子	Z' (路径标签)	插入 WST	D	λ_1	D_1	删除冗余部分
y_{13}	ACAAAT	ACAAAT$ CAAAT$	2	0	2	ACAAAT$ → ACAA$ CAAAT$→ CAA$
y_{14}	ACAACT	ACAACT$ CAACT$	2	0	2	ACAACT$ → ACAA$ CAACT$→ CAA$
y_{15}	ACAAGT	ACAAGT$ CAAGT$	2	0	2	ACAAGT$ → ACAA$ CAAGT$→ CAA$
y_{16}	ACCAAT	ACCAAT$ CCAAT$	2	0	2	ACCAAT$ → ACCA$ CCAAT$→ CCA$
y_{17}	ACCACT	ACCACT$ CCACT$	2	0	2	ACCACT$ → ACCA$ CCACT$→ CCA$
y_{18}	ACCAGT	ACCAGT$ CCAGT$	2	0	2	ACCAGT$ → ACCA$ CCAGT$→ CCA$

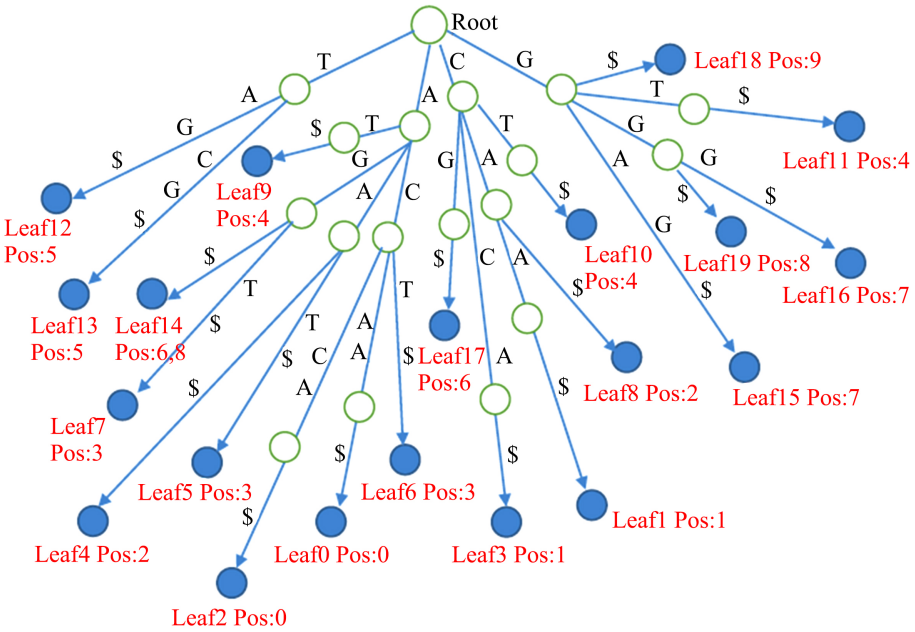

Figure 11. A complete weighted suffix tree
图 11. 完整的加权后缀树

3. 结果与分析

根据影响实验运行时间及结果的三个因素：模式 P，阈值 K 和不同长度的测试文件进行多组测试。测试数据文件：text2k.fa，text20k.fa，text50k.fa，text100k.fa，text150k.fa，text200k.fa。分别取文本长度 2000、20,000、50,000、100,000、150,000、200,000，选取 K = 2、K = 4、K = 6、K = 8、模式 P 长度等于 2、4、8、16、32 进行一系列测试。部分实验结果见表 3。

Table 3. Experimental results
表 3. 实验结果

模式 P	K 值	文件名	加权序列长度	后缀树数量	发生次数	发生位点
ATAGCAGG	2	text200k.fa	200,000	5074	1	6184
ATAGCAGG	4	text200k.fa	200,000	5074	4	6184, 16421, 83378, 105107
AACG	4	text200k.fa	200,000	5074	919	省略
AACG	8	text200k.fa	200,000	5074	934	省略
AACG	8	text50k.fa	50,000	1335	235	省略
GCCGCGTCGTAGCT CAACATCTGTCTGT TCGCGT	4	text50k.fa	50,000	1335	1	20516

基于阈值为 4 的 K，测试不同长度的模式 P 和不同长度的测试数据。从表 4 可以看出，当阈值 K 固定时，模式 P 的长度对运行时间影响不大，而测试数据长度对运行时间影响较大。

Table 4. K = 4
表 4. K = 4

模式长度 / 文件长度	(K = 4) 2	4	8	16	32
2 k (B:49)	1.750s	1.804s	1.518s	1.478s	2.228s
20 k (B:528)	6.240s	5.830s	6.242s	6.446s	6.208s
50 k (B:1335)	14.736s	14.421s	15.098s	13.764s	14.079s
100 k (B:2552)	27.677s	27.632s	28.018s	27.573s	27.380s
150 k (B:3800)	40.257s	40.176s	41.628s	40.556s	40.456s
200 k (B:5074)	57.894s	53.196s	54.380s	55.326s	54.868s

基于长度为 8 的模式 P，测试不同阈值 K 和不同长度的测试数据。从表 5 可以看出，当模式 P 长度固定时，阈值 K 和测试数据长度都对运行时间有一定的影响。

Table 5. The length of pattern P is 8
表 5. 模式 P 长度 = 8

K 值 / 文件长度 (PL = 8)	2	4	6	8
2 k (B:49)	1.248s	1.518s	1.495s	1.905s
20 k (B:528)	1.265s	6.242s	5.729s	12.331s
50 k (B:1335)	3.228s	15.098s	14.093s	28.141s
100 k (B:2552)	4.959s	28.018s	26.692s	63.693s
150 k (B:3800)	7.749s	41.628s	41.820s	93.868s
200 k (B:5074)	10.096s	54.380s	55.480s	124.825s

基于长度为 100 k 的测试数据，测试不同阈值 K 和不同长度的模式 P。从表 6 可以看出，当测试数据长度固定时，模式 P 长度对运行时间影响不大，而阈值 K 对运行时间影响较大。

Table 6. The length of test data is 100 k
表 6. 测试数据长度 = 100 k

模式长度 / K 值 (text 100 k)	2	4	8	16	32
2	6.991s	6.555s	4.959s	5.967s	5.867s
4	27.677s	27.632s	28.018s	27.573s	27.380s
6	36.010s	28.918s	26.692s	26.876s	28.765s
8	62.897s	62.859s	63.693s	58.875s	63.819s

综上所述，当阈值 K 越来越大时，黑色点位会越来越多，需要建立更多后缀树，该算法的运行时间主要与阈值 K 正相关。从实验结果可以看出，相较于常规的 Map-Based 算法和双 DNA 序列比对算法的图像立体匹配方法，本文提出的算法在模式较为复杂的情况下，匹配速度及不确定数据的位点精准度上表现优异，这主要是因为 Map-Based 算法比较适用于模式比较简单、长度较短情况下单个序列的模式计算，在数据位点定位上存在较大缺陷。在进行算法匹配前，本文提出的算法仅需要对源数据进行处理，而 Map-Based 算法需要对模式及源数据都进行预处理，此过程要进行 $m \times n$ 次，m、n 分别为序列数集和模式数集，此过程进行的辅助数据的重复计算会给模式匹配的性能带来很大的损失，预处理消耗的时间远大于加速时间。

本文提出的基于加权后缀树的模式匹配算法在应用到 DNA 分子序列特征计算时，能很好地应对 DNA 序列数目庞大、单个 DNA 序列的长度较长的情况，并精确地定位发生位点的位置，解决了仅能进行次数计算的缺点。实验结果表明，在实际应用中该算法不仅效率更高而且匹配精准度及性能稳定性更高，并且易于实现，具有一定的应用推广价值，对探究基因序列知识具有重要意义。

基金项目

廊坊市科技支撑计划项目(2020011045)，北华航天工业学院科研基金项目(KY-2020-16)，北华航天工

业学院研究生创新资助项目(YKY-2021-20)。

参考文献

[1] 应嘉, 赵睿颖, 尚彤. 生物信息学在人基因组计划中的应用[J/OL]. 北京大学学报(医学版), 2002, 34(4): 389-392. https://doi.org/1671-167X(2002)04-0389-04

[2] 鲍芸, 肖艳群, 王华梁. 高通量测序技术的临床应用及质量管理[J]. 中华检验医学杂志, 2022, 45(11): 1099-1103.

[3] 谢娟英, 王明钊, 许升全. 面向甲基化修饰位点预测的 DNA/RNA 序列特征编码算法研究进展[J/OL]. 中国科学: 生命科学, 2022, 1-35. https://doi.org/10.1360/SSV-2022-0074

[4] Hasan, M.M., Basith, S., Khatun, M.S., *et al*. (2021) Meta-i6mA: An Interspecies Predictor for Identifying DNA N6-methyladenine Sites of Plant Genomesby Exploiting Informative Features in an Integrative Machine-Learning Framework. *Briefings in Bioinformatics*, **22**, bbaa202. https://doi.org/10.1093/bib/bbaa202

[5] Dai, C., Feng, P., Cui, L., *et al*. (2021) Iterative Feature Representation Algorithm to Improve the Predictive Performance of N7-methylguanosine Sites. *Briefings in Bioinformatics*, **22**, bbaa278. https://doi.org/10.1093/bib/bbaa278

[6] Fang, T., Zhang, Z., Sun, R., *et al*. (2019) RNAm5CPred: Prediction of RNA 5-Methylcytosine Sites Based on Three Different Kinds of Nucleotidecomposition. *Molecular Therapy—Nucleic Acids*, **18**, 739-747. https://doi.org/10.1016/j.omtn.2019.10.008

[7] Liu, L., Lei, X., Meng, J., *et al*. (2020) ISGm1A: Integration of Sequence Features and Genomic Features to Improve the Prediction of Human m_1A RNA Methylation Sites. *IEEE Access*, **8**, 81971-81977. https://doi.org/10.1109/ACCESS.2020.2991070

[8] Yang, X., Ye, X., Li, X., *et al*. (2021) iDNA-MT: Identification DNA Modification Sites in Multiple Species by Using Multi-Task Learning Based a Neuralnetwork Tool. *Frontiers in Genetics*, **12**, 411. https://doi.org/10.3389/fgene.2021.663572

[9] Zhang, L., Xiao, X. and Xu, Z.C. (2020) iPromoter-5mC: A Novel Fusion Decision Predictor for the Identification of 5-Methylcytosine Sites in Genome-Wide DNA Promoters. *Frontiers in Cell and Developmental Biology*, **8**, 614. https://doi.org/10.3389/fcell.2020.00614

[10] Khanal, J., Tayara, H. and Chong, K.T. (2020) Identifying Enhancers and Their Strength by the Integration of Word Embedding and Convolution Neural Network. *IEEE Access*, **8**, 58369-58376. https://doi.org/10.1109/ACCESS.2020.2982666

[11] Cai, L., Ren, X., Fu, X., *et al*. (2021) iEnhancer-XG: Interpretable Sequence-Based Enhancers and Their Strength Predictor. *Bioinformatics*, **37**, 1060-1067. https://doi.org/10.1093/bioinformatics/btaa914

[12] Chen, W., Feng, P., Yang, H., *et al*. (2017) iRNA-AI: Identifying the Adenosine to Inosine Editing Sites in RNA Sequences. *Oncotarget*, **8**, 4208-4217. https://doi.org/10.18632/oncotarget.13758

[13] Chandra, A., Sharma, A., Dehzangi, A., *et al*. (2019) Bigram-PGK: Phosphoglycerylation Prediction Using the Technique of Bigram Probabilities of Positionspecific Scoring Matrix. *BMC Molecular and Cell Biology*, **20**, 57. https://doi.org/10.1186/s12860-019-0240-1

[14] Zhang, Y., Qiao, S., Ji, S., *et al*. (2020) DeepSite: Bidirectional LSTM and CNN Models for Predicting DNA-Protein Binding. *International Journal of Machine Learning and Cybernetics*, **11**, 841-851. https://doi.org/10.1007/s13042-019-00990-x

[15] Michalak, E.M., Burr, M.L., Bannister, A.J., *et al*. (2019) The Roles of DNA, RNA and Histone Methylation in Ageing and Cancer. *Nature Reviews Molecular Cell Biology*, **20**, 573-589. https://doi.org/10.1038/s41580-019-0143-1

[16] 陶赛群, 陈力, 李静. 乙醛脱氢酶 2 基因多态性与癌症的关系及药物治疗的研究进展[J]. 现代药物与临床, 2022, 37(3): 666-672. https://doi.org/10.7501/j.issn.1674-5515.2022.03.039

[17] 阳雪兰, 曾雷. 基因融合与癌症发生发展关系及其致病机制的研究进展[J]. 吉林大学学报(医学版), 2022, 48(2): 527-532. https://doi.org/10.13481/j.1671-587X.20220233

[18] Wahab, A., Mahmoudi, O., Kim, J. and Chong, K.T. (2020) DNC4mC-Deep: Identification and Analysis of DNA N4-methylcytosine Sites Based on Different Encodingschemes by Using Deep Learning. *Cells*, **9**, 1756. https://doi.org/10.3390/cells9081756

[19] Liu, L., Song, B., Ma, J., *et al*. (2020) Bioinformatics Approaches for Deciphering the Epitranscriptome: Recent Progress and Emerging Topics. *Computational and Structural Biotechnology Journal*, **18**, 1587-1604. https://doi.org/10.1016/j.csbj.2020.06.010

[20] Chen, Z., Zhao, P., Li, C., *et al*. (2021) iLearnPlus: A Comprehensive and Automated Machine-Learning Platform for Nucleic Acid and Protein Sequenceanalysis, Prediction and Visualization. *Nucleic Acids Research*, **49**, e60. https://doi.org/10.1093/nar/gkab122

[21] 戴胜冬, 杨昆. 计算 DNA 序列模式特征的匹配算法[J]. 杭州电子科技大学学报(自然科学版), 2015, 35(1): 88-92. https://doi.org/10.13954/j.cnki.hdu.2015.01.018

[22] Apostolico, A., Crochemore, M., Farach-Colton, M., Galil, Z. and Muthukrishnan, S. (2016) 40 Years of Suffix Trees. *Communications of the ACM*, **59**, 66-73. https://doi.org/10.1145/2810036

[23] Iliopoulos, C.S., Makris, C., Panagis, Y., Perdikuri, K., Theodoridis, E. and Tsakalidis, A. (2006) The Weighted Suffix Tree: An Efficient Data Structure for Handling Molecular Weighted Sequences and Its Applications. *Fundamenta Informaticae*, **71**, 259-277.