МЕТОДЫ ОБРАБОТКИ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ И СИГНАЛОВ
(Вычислительные системы)
1989 год Выпуск 132

УДК 519.764

АЛГОРИТМ ПОСТРОЕНИЯ СЛОЖНОСТНОГО ПРОФИЛЯ СИМВОЛЬНОЙ ПОСЛЕДОВАТЕЛЬНОСТИ

О.М. Чупахина

Введение

Сложностные профили являются эффективным и перспективным средством выявления локальных структурных закономерностей в символьных последовательностях различной природы [1]. Сложность конечной последовательности [2] оказывается весьма универсальной характеристикой, аккумулирующей в себе множество частных закономерностей. Особенно ярко ее преимущества проявляются при анализе первичных структур ДНК-молекул [3,4].

В данной работе описан эффективный алгоритм вычисления сложностного профиля для текстов большой длины (10^4-10^6 символов). Именно такой порядок имеют длины геномов многих простейших микроорганизмов. Трудоемкость алгоритма составляет $O(N \cdot \log^2 D / |\Sigma|)$, где N - длина текста, $|\Sigma|$ - число элементов алфавита, D - размер окна анализа.

В основу алгоритма положена конструкция, называемая "деревом префикс-идентификаторов". В алгоритме можно выделить три блока, представляющих самостоятельный интерес: 1) блок вычисления меры сложности конечной последовательности с помощью префиксного дерева (вариант, основанный на хешировании, описан в [1]); 2) блок корректировки префиксного дерева при продвижении окна анализа вдоль последовательности (этот блок

может использоваться независимо в алгоритме сжатия текста); 3) блок корректировки меры сложности при движении окна вдоль последовательности (этап вычисления сложностного профиля).

Элементы новизны содержатся во всех трех блоках. Относительно первого блока можно заметить, что в [5] лишь отмечалась возможность использования префиксных деревьев для вычисления меры сложности, алгоритм не был опубликован. Схема коррекции префиксного дерева в окне анализа, реализованная во втором блоке, в определенном смысле "обратна" той, что описана в [6] (трудоемкость обоих алгоритмов одинакова). Следует заметить, что для родственной (по отношению к префиксному дереву) конструкции, называемой "направленным ациклическим графом слова", также разработан алгоритм корректировки графа для движущегося окна с тем же порядком трудоемкости, что и для префиксного дерева [7]. Задача корректировки меры сложности в движущемся окне (третий блок) впервые возникла в связи с введением понятия "сложностной профиль". Схема ее решения с использованием префиксных деревьев является новой.*)

1. Исходные обозначения и понятия

Введем следующие обозначения: Σ - конечный алфавит; строка - конечная последовательность символов из Σ; Σ^* - множество всех строк над алфавитом Σ; λ - пустая строка, т.е. строка, не содержащая ни одного символа; $l(S)$ - длина строки S; $S = QR$ - конкатенация строк Q и R (если $l(Q) = N_1$, $l(R) = N_2$, то $l(S)=N_1+N_2$, $Q = S[1:N_1]$, $R = S[N_1+1: N_1+N_2]$). Будем говорить, что элемент a расположен в i-й позиции строки S $(a = S[i])$, если $S = QaR$

*) В дипломной работе студентки механико-математического факультета НГУ Запрягаевой Е.В., выполненной в 1985 г. под руководством Гусева В.Д., рассматривался альтернативный (и более трудоемкий) вариант с использованием идей хеширования.

и $l(Q) = i-1$; соответственно через $S[i:j]$ обозначим фрагмент S , включающий элементы с i-го по j-й включительно , $1 \le i < j \le N$. Если $W = SQR$, где $S, Q, R \in \Sigma^*$, то будем называть Q подстрокой W, S - префиксом W , а R - суффиксом строки W .

1.1. Префиксное дерево. Пусть θ - символ, отсутствующий в Σ. Для строки $S\theta$ определим префикс-идентификатор позиции i как самую короткую подстроку Q из $S\theta$,которая начинается с позиции i и встречается в $S\theta$ только один раз. Символ θ назовем концевым маркером. Он используется для того, чтобы гарантировать существование префикс-идентификатора для каждой позиции. Совокупность всех префикс-идентификаторов строки $S\theta$ можно представить в виде дерева T ,которое называется деревом префикс-идентификаторов или префиксным деревом строки $S\theta$ [8]. В [6,9] эта конструкция названа позиционным деревом. Ребра дерева помечены символами из множества $(\Sigma \cup \{\theta\})$. Если ребро (M,K), соединяющее вершины M и K, помечено символом a , будем называть вершину K a-сыном вершины M. Дерево префикс-идентификаторов удовлетворяет следующим свойствам:

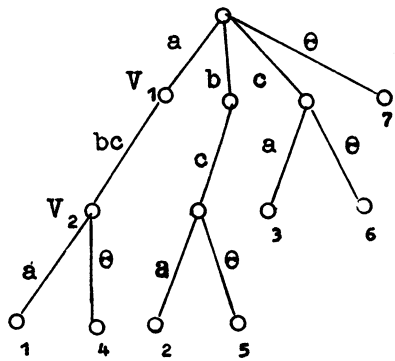
1. T имеет ровно N листьев, помеченных цифрами от 1 до N . Лист с номером i соответствует i-й позиции;

2. последовательность меток ребер, лежащих на пути из корня T в лист с меткой i , образует префикс-идентификатор позиции i.

Каждой строке соответствует единственное дерево префикс-идентификаторов (см. [8]). В общем случае префиксное дерево для строки S длины N может содержать $O(N^2)$ узлов. Однако можно "уплотнить" префиксное дерево,выявив в нем все цепные участки (цепью называется путь, каждый узел которого обладает в точности одним сыном) и сжав все вершины каждой цепи в один узел. Уплотненное таким образом дерево называется компакт-

ным префиксным деревом. Оно удовлетворяет следующему свойству [8]: компактное префиксное дерево содержит не более $3N-2$ узлов. Там же показано, что если M - это вершина в T, а M' - отец M, то либо M, либо M' имеет не менее двух сыновей.

Для примера на рис.1 изображено компактное префиксное дерево и префикс-идентификаторы для строки $S\theta = abcabc\theta$.



Позиция	Префикс-идентификатор
1	abca
2	bca
3	ca
4	abcθ
5	bcθ
6	cθ
7	θ

Рис. 1

Отметим, что вершины V_1 и V_2 не образуют цепь, поскольку вершина V_2 имеет двух сыновей.

1.2. Сложность конечной последовательности. Пусть $S = a_1 \ldots a_N$ - заданная строка длины N. Определим ее сложность $C(S)$ [2] как число шагов некоторого процесса, последовательно разбивающего данную строку на фрагменты (компоненты) при помощи фиксированного множества допустимых операций. В качестве допустимых используются две операции: операция порождения нового символа и операция копирования любого фрагмента из предыстории, т.е. из префикса строки S, уже разбитого на компоненты.

Предположим, что за k шагов префикс $S[1:i_k]$ разбит на k фрагментов. Тогда на $(k+1)$-м шаге процесса отыски-

ваем в префиксе $S[1:i_k]$ фрагмент, либо целиком лежащий в нем, либо выходящий за его пределы и совпадающий с некоторым префиксом $S[i_k+1:i_{k+1}]$ подстроки $S[i_k+1:N]$. Если $S[i_k+1:i_{k+1}] \neq \lambda$, то, используя правило копирования фрагмента, удлиняем префикс $S[1:i_k]$ ровно на $1\,(S[i_k+1:i_{k+1}])$ символов и получаем $S[i_k+1:i_{k+1}]$ - новый $(k+1)$-й компонент. Если же $S[i_k+1:i_{k+1}] = \lambda$, то это означает, что символ $S[i_k+1]$ в префиксе $S[1:i_k]$ отсутствует. В этом случае, используя правило генерации нового символа, удлиняем $S[1:i_k]$ ровно на один символ, являющийся отдельным компонентом.

Представление строки S в виде конкатенации компонентов назовем историей формирования $H(S)$ строки S :

$$H(S)=S[1:i_1]S[i_1+1:i_2]\ldots S[i_{m-1}+1:N],$$

где $S[i_{k-1}+1:i_k]$ - k-й компонент истории, а m - число шагов процесса.

Заметим, что историй формирования строки S может быть несколько. Обозначим через $C_H(S)$ число компонентов в истории H строки S. Тогда меру сложности строки S определим как минимум по всем историям $H(S)$ строки S :

$$C(S) = \min_H \{ C_H(S) \} .$$

В [2] показано, что минимум достигается при таком процессе, когда каждый копируемый компонент истории имеет максимально возможную длину. Такая история определяется для S единственным образом.

1.3. Сложностной профиль текста. Понятие сложностного профиля текста возникает в связи с вычислением сложности отдельных фрагментов текста. Если сканировать текст от начала к концу окном фиксированной длины и вычислять сложность для каждого фрагмента, выделяемого окном, получим последовательность зна-

чений $P(S,D) = C_1 C_2 \ldots C_{N-D+1}$, где C_i - это сложность фрагмента $S_i = S[i:i+D-1]$, D -размер окна, N - длина текста S . Последовательность $P(S,D)$ назовем сложностным профилем текста S . Ниже представлен быстрый алгоритм вычисления сложностного профиля,в котором существенно используется "зацепленность" соседних фрагментов.

2. Схема вычисления сложностного профиля

Поскольку основой для вычисления префиксного дерева для нас будет служить алгоритм Вайнера [8], в котором анализ последовательности ведется справа налево, удобно и сложностной профиль вычислять, начиная со значения C_{N-D+1} .

Выделим фрагмент $S_{N-D+1} = S[N-D+1:N]$, дополним его концевым маркером Θ справа, построим для него компактное префиксное дерево T_{N-D+1} и вычислим меру сложности (см.алгоритм А1).

Для значений $1 \le j \le N-D$ префиксное дерево T_j не строится каждый раз заново, а получается путем соответствующей корректировки префиксного дерева T_{j+1} , использовавшегося на предыдущем шаге алгоритма (см.алгоритм А2). Мера сложности C_j также не вычисляется заново по префиксному дереву T_j ,а определяется путем коррекции (с помощью дерева T_j) меры сложности C_{j+1} , полученной на предыдущем шаге (см.алгоритм А3).Таким образом, для каждого положения окна проводятся две корректировки: дерева префикс-идентификаторов и меры сложности.

3. Алгоритм вычисления меры сложности с использованием префиксного дерева (алгоритм А1)

Пусть $S = a_1, \ldots, a_N$ и для строки $S\Theta$ построено префиксное дерево. Свяжем с каждой вершиной V префиксного дерева числовую величину $J(V)$, соответствующую самой левой позиции в S , с которой начинается подстрока $x(V)$, со-

ставленная из реберных меток, лежащих на пути из корня до вершины V. Пусть $l(V)$ - длина подстроки $x(V)$, а P_i - префикс-идентификатор позиции i.

Способ вычисления величины $l(V)$ заложен в алгоритме Вайнера. Начальные значения $J(V)$, соответствующие первому этапу построения сложностного профиля $(i = N-D+1)$, вычисляются по алгоритму, приведенному ниже. Для остальных значений i, $1 \le i < N-D+1$, величины $J(V)$ не вычисляются заново, а лишь корректируются с помощью алгоритма А3.

Схема вычисления начальных значений $J(V)$ такова. Последовательно просматриваем все цепи, ведущие из каждого листа дерева в его корень. Просмотр начинаем с листа с меткой 1. Пусть $(V_i = V_{i_1}), V_{i_2}, \ldots, (V_{i_l} = V_0)$ - последовательность вершин, составляющих путь из листа с меткой i в корень дерева. Поскольку строка, соответствующая листу V_i, встречается в тексте только один раз, то для нее значение $J(V_i) = i$. Далее последовательно просматриваем вершины $V_{i_2}, \ldots, V_{i_l}$. Если значение $J(V_{i_k})$ не определено, это означает, что подстрока $x(V_{i_k})$ не встречалась левее i-й позиции (иначе вершина V_{i_k} была бы помечена при просмотре предыдущих цепей). Поэтому полагаем $J(V_{i_k}) = i$ и продолжаем просмотр цепи.

Если найдется вершина V_{i_m} такая, что величина $J(V_{i_m})$ определена, это означает, что в данной вершине мы побывали раньше (при просмотре цепи, соответствующей одному из листьев с номером $i' < i$). Поскольку из любой вершины дерева в его корень ведет единственный путь, цепочки вершин, образующие пути из листьев i и i' в корень дерева, совпадают, начиная с вершины V_{i_m}. Так как просмотр каждой цепи гарантирует присвоение меток всем ее вершинам, то вершины $V_{i_{m+1}}, \ldots$

..., V_{i_1}, лежащие выше вершины V_{i_m}, уже помечены на шаге i. Значения этих меток уже не могут измениться исходя из самого смысла величин $J(V)$. Поэтому, начиная с вершины V_{i_m}, просмотр оставшихся вершин цепи прекращается и начинается анализ цепочки, соответствующей листу с номером $i+1$. Описанная процедура гарантирует просмотр всех вершин префиксного дерева. Так как число вершин в компактном префиксном дереве не превышает $3N-2$ и никакая вершина не просматривается дважды, то трудоемкость вычисления величин $J(V)$ составляет $O(N)$.

Изложим теперь схему вычисления меры сложности с использованием префиксного дерева, помеченного описанным выше способом. Первый символ в цепочке S всегда является первым компонентом сложности, так как он не может быть скопирован (используется операция генерации). Предположим, что для $a_1,\ldots,\ldots, a_{i-1}$ получено разложение на компоненты сложности, причем символ a_{i-1} является последним элементом соответствующего компонента сложности (т.е. добавление a_i не приведет к удлинению этого компонента). Покажем, как определяется очередной компонент, начинающийся в i-й позиции.

Рассмотрим лист V_i с меткой i. Последовательно просматриваем все вершины на пути из листа V_i в корень префиксного дерева, сравнивая каждый раз значение $J(V_{i_k})$ со значением i. Если встречается вершина V, у которой $J(V)<i$, это означает, что существует подстрока, соответствующая вершине V, начало которой лежит левее позиции i. Следовательно, компонент сложности, начинающийся с i-й позиции, может быть скопирован с этой подстроки. Эта подстрока является максимальной, поскольку все строки, соответствующие вершинам, лежащим "выше" вершины V, короче подстроки, соответствующей вершине V, а все те вершины, которые лежат "ниже" вершины

V , соответствуют строкам, начинающимся с позиции i . Следовательно, длина нового компонента сложности равна $l(V)$. Прибавляя к i величину $l(V)$, определяем начало очередного компонента сложности $i' = i+l(V)$ и повторяем процесс. Если в какой-то момент значение i' превысит N , то разложение на компоненты сложности считаем законченным.

Может оказаться, что на пути из листа i в корень не встретится ни одной вершины, у которой значение $J(V) < i$. Такое возможно только тогда, когда в позиции i находится символ, не встречавшийся ранее. В этом случае полагаем длину компонента сложности равной единице, значение сложности увеличиваем на единицу, поиск нового компонента начинаем с позиции $i+1$.

Приведем структурную схему алгоритма вычисления меры сложности с использованием префиксного дерева.

Алгоритм А1.

I. Обозначим через C меру сложности, I - текущий индекс. Полагаем $C = 1$, $I = 2$;

П. Если $I \leq N$, то выполняем следующие действия:

1. Обозначим через $V(I)$ лист с номером I ; через V и V' - вершины префиксного дерева.

2. $V = V(I)$; V' - это отец вершины V .

3. Если V' не корень префиксного дерева, то выполняем следующие действия:

а) Если $J(V) < I$, то выполняем: $I = I + l(V)$; $C = C + 1$; возвращаемся на шаг П.

б) Иначе полагаем: $V = V'$, V' - отец вершины V ; возвращаемся на шаг П.3.

4. Иначе $I = I+1$; $C = C+1$.

4. Алгоритм корректировки компактного префиксного дерева при сканировании последовательности справа налево окном фиксированной длины (алгоритм A2)

4.1. Обоснование. Алгоритм корректировки компактного префиксного дерева при сдвиге окна на один символ влево состоит из двух последовательно выполняемых блоков - $(A2)_1$ и $(A2)_2$.

Пусть для строки $S_j = a_j, \ldots, a_{j+D-1}\theta$ построено компактное префиксное дерево T_j. Сначала добавляем к S_j один символ слева и достраиваем по алгоритму Вайнера [8] T_j до дерева T'_{j-1}, соответствующего строке $S'_{j-1} = a_{j-1}a_j \ldots a_{j+D-1}\theta$ (блок $(A2)_1$), затем устраняем из S'_{j-1} крайний (отличный от θ) символ справа, корректируем дерево T'_{j-1} (блок $(A2)_2$) и получаем компактное префиксное дерево T_{j-1} для строки $S_{j-1} = a_{j-1}a_j \ldots a_{j+D-2}\theta$.

Опишем лишь новый алгоритм коррекции компактного дерева при укорочении строки на один символ справа (блок $(A2)_2$), отсылая за подробностями реализации блока $(A2)_1$ к работе [8]. Пусть $S = a_1 \ldots a_{N-1}$, где $a_j \in \Sigma$, $1 \le j \le N-1$. Образуем конкатенацию $Sa\theta$, где $a \in \Sigma$, $\theta \notin \Sigma$. Покажем, какого рода коррекция необходима, чтобы перевести префиксное дерево, соответствующее строке $Sa\theta$, в префиксное дерево для строки $S\theta$. Как и раньше, под P_j будем понимать префикс-идентификатор для позиции j, а под $l(P_j)$ - его длину.

Вначале сформируем два списка префикс-идентификаторов, подлежащих корректировке. Первый из них (обозначим его $L1$) включает все префикс-идентификаторы, заканчивающиеся концевым маркером θ (кроме самого θ). Все они должны измениться, поскольку содержат устраняемый символ "a". Пусть наиболее длинный из этих префикс-идентификаторов соответствует позиции j_0 (обозначим его P_{j_0}). Тогда из определения префикс-идентификатора непосредственно следует, что префикс-идентифи -

каторы для позиций $j_0+1, j_0+2, \ldots, N+1$ (и только они) так - же заканчиваются концевым маркером, т.е. список $L1 = \{P_N, P_{N-1}, \ldots, P_{j_0}\}$.

Действит льно, предположим, что это не так, т.е. сущест - вует такой префикс-идентификатор для позиции j, $j_0 < j \le N$, который заканчивается не концевым маркером θ, а некоторым символом a_k, $j \le k \le N$. Тогда подстрока $S[j:k]$ встреча- ется в S всего один раз и префикс-идентификаторы $P_{j_0}, \ldots$ $\ldots, P_{j-1}$ должны заканчиваться символом a_k, а не θ, иначе нарушается требование минимальности длины в определении префикс-идентификатора. Возникшее противоречие свидетельству- ет о том, что предполагаемая ситуация невозможна.

Второй список (обозначим его $L2$) состоит из тех пре - фикс-идентификаторов, которые заканчиваются символом "a", под- лежащим устранению. Формируем его следующим образом: для по- зиций $j_0-1, j_0-2,$ и т.д. проверяем выполнение условия

$$j_0 - k+1(P_{j_0-k}) = N, \quad k = 1,2,\ldots, m, \qquad (1)$$

до тех пор, пока при некотором $m \ge 1$ оно не перестанет выпол- няться. Тогда $L2=\{P_{j_0-1}, P_{j_0-2}, \ldots, P_{j_0-m+1}\}$. Заметим, что списки $L1$ и $L2$ могут оказаться и пустыми, но не одновре- менно, так как префикс-идентификатор для позиции N равен либо "a", либо "$a\theta$".

Кроме концевых префикс-идентификаторов, вошедших в спис- ки $L1$ и $L2$, изменению могут подвергнуться также некото- рые внутренние префикс-идентификаторы, которые перестают быть таковыми при устранении символа "a". Формально их можно было бы выделить в отдельный список, но мы не будем этого делать, поскольку их корректировка увязана с корректировкой списка $L1$ (см. случай 2).

Рассмотрим, как изменяются концевые префикс-идентификато- ры, в состав которых входит удаляемый символ "a". Заметим,

что все такие префикс-идентификаторы и только они составляют списки $L1$ и $L2$.

Пусть P_j - префикс-идентификатор из списка $L1$. Тогда его можно представить в виде $P_j = Ra\theta$, где $R = a_j, \ldots, a_{N-1}$. Поскольку θ является концевым маркером, то после удаления символа "a" цепочка $R\theta$ встречается в строке $S\theta$ один раз. А так как цепочка R встречается в $S\theta$, как минимум, два раза, $P_j = R\theta$ является префикс-идентификатором для позиции j в строке $S\theta$. По этой схеме корректируются все элементы списка $L1$.

Поясним теперь, какие внутренние префикс-идентификаторы могут измениться в связи с устранением символа "a" и как они связаны с элементами списка $L1$. Если $P_j \in L1$, то по определению префикс-идентификатора существуют одна или несколько позиций j_k, $1 \le k \le h$, префикс-идентификаторы которых имеют вид: $P_{j_k} = RaQ_{j_k}$, где Q_{j_k} - подстрока $Sa\theta$. Рассмотрим случай, когда $h = 1$, т.е. подстрока Ra в строке $Sa\theta$ встречается дважды: в позициях j и j_k. Из определения префикс-идентификатора непосредственно следует, что подстрока Q_{j_k} в данном случае состоит из одного символа $S[j_k + l(Ra)]$. Назовем его b. Тогда идентификатор P_{j_k} можно записать в виде $P_{j_k} = Rab$. Так как после удаления символа "a" подстрока Ra встречается в строке $S\theta$ один раз и она короче подстроки Rab, то префикс-идентификатором позиции j_k в строке $S\theta$ по определению становится строка Ra. Нетрудно видеть, что рассмотренный случай исчерпывает все возможности изменения внутренних префикс-идентификаторов, связанные с устранением символа "a". Иначе говоря, при $h > 1$ префикс-идентификаторы P_{j_k}, $1 \le k \le h$, не меняются при устранении концевого символа.

Пусть теперь префикс-идентификатор $P_i \in L2$, тогда он представим в виде $P_i = Ra$ и цепочка Ra в строке $Sa\theta$ встречается только один раз. Образующаяся после удаления символа "a" цепочка $R\theta$ также встречается один раз в строке $S\theta$, так как θ - это концевой маркер. Поскольку цепочка R встречается в $S\theta$, как минимум, 2 раза, то $P_i = R\theta$ является префикс-идентификатором для позиции i в строке $S\theta$.

Рассмотрим теперь, каких изменений в дереве префикс-идентификаторов потребует описанная выше схема коррекции. Применительно к списку $L1$ классифицируем вначале все случаи расположения сыновей у вершины, имеющей θ-сына.

Пусть V_1 - это вершина, имеющая θ-сына V. Тогда V - это лист префиксного дерева и соответствующая ему строка является префикс-идентификатором, заканчивающимся θ. Обозначим через V_2 отца вершины V_1. Положим для простоты, что ребро V_2V_1 помечено одним символом "a", подлежащим устранению. Обозначим через R строку, соответствующую вершине V_2. Будем различать следующие 4 варианта расположения сыновей у вершины V_1.

1. Кроме θ-сына, вершина V_1 имеет еще не менее двух сыновей (рис.2а).

2. Кроме θ-сына, вершина V_1 имеет еще одного сына - вершину A, являющуюся листом в префиксном дереве (рис.3а).

3. Кроме θ-сына, вершина V_1 имеет еще одного сына A, и вершина A имеет не менее 2-х сыновей (рис.4а).

4. Кроме θ-сына, вершина V_1 имеет еще одного сына A, и вершина A имеет единственного сына A_1. В этом случае по определению компактного префиксного дерева, вершина A_1 имеет не менее 2-х сыновей (рис.5а).

Отметим, что вариант, когда вершина V_1 имеет единственного θ-сына V, невозможен (если только $V_1 \neq V_0$), поскольку в этом случае как строка, соответствующая вершине V_1,

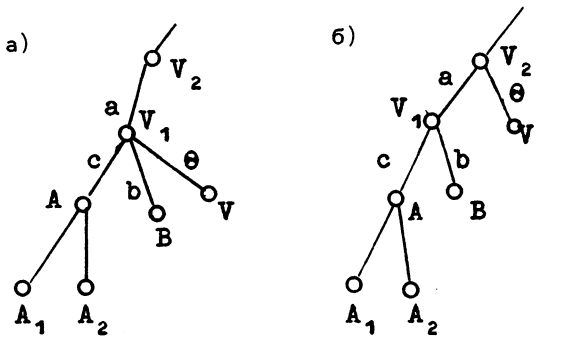


Рис. 2

Рис. 3

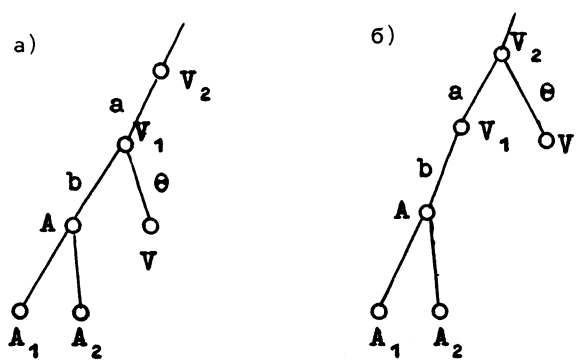


Рис. 4

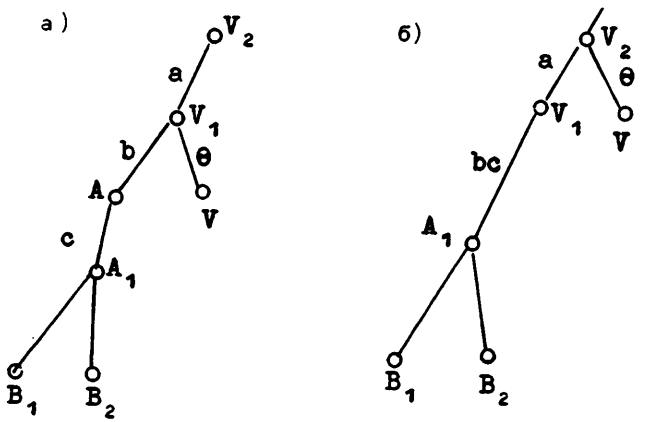


Рис. 5

так и строка, соответствующая вершине V, встречаются в тексте один раз. А так как строка, соответствующая вершине V_1, короче, то возникает противоречие с тем, что V соответствует префикс-идентификатору.

Во всех четырех случаях после удаления символа "a" справа префикс-идентификатор, соответствующий вершине V, становится короче на один символ и имеет вид $R\theta$. Следовательно, необходимо сделать вершину V θ-сыном вершины V_2. В случаях 1 и 3 корректировка на этом заканчивается (рис.2б, 4б).

В случае 4 после удаления θ-сына у вершины V_1 как вершина A, так и ее отец V_1 имеют единственного сына. Чтобы не нарушить условие компактности, вершину A объединяем с вершиной V_1, а полученное ребро V_1A_1 помечаем конкатенацией символов, помечавших ребра V_1A и AA_1 (рис.5б).

В случае 2 (рис.3б) после корректировки у вершины V_1 остается единственный лист A, соответствующий некоторому префикс-идентификатору j. Поскольку это противоречит определению префикс-идентификатора, то удаляем вершину A, делаем вершину V_1 листом и помечаем его позиционным номером j.

Мы рассмотрели случай, когда ребро V_2V_1 помечено единственным символом "a". Но поскольку мы имеем дело с компактным префиксным деревом, возможна ситуация, когда ребро V_2V_1 помечено цепочкой символов. Это означает, что в префиксном дереве, построенном для строки $Sa\theta$, отсутствует вершина, соответствующая подстроке R. Этот случай сводится к предыдущему путем создания вспомогательной вершины M, соответствующей цепочке R.

Изменения, вносимые в дерево в связи с коррекцией списка $L2$, незначительны. Поскольку в каждом префикс-идентификаторе из $L2$ последний символ ("a") заменяется на θ, в дереве не появляются новые вершины и не исчезают старые, происходит лишь смена меток у ребер, соответствующих концевым вершинам цепочек из $L2$.

Поскольку префикс-идентификатор для $(N+1)$-й позиции не входит в список $L1$, то в алгоритме $(A2)_2$ он корректируется отдельно. Изложенные выше соображения позволяют без дальнейших пояснений представить структурную схему алгоритма коррекции префиксного дерева.

4.2. Алгоритм $(A2)_2$.

I. Пусть V_0 – корень дерева T. У вершины V_0 удаляем θ-сына.

II. Для каждого префикс-идентификатора P_j из списка $L1$ (в порядке убывания позиций) выполняем следующие действия.

1. Обозначаем через V лист с меткой j, через V_1 – отца вершины V, а через V_2 – отца вершины V_1.

2. Определяем вершину M следующим образом:

а) если ребро V_2V_1 помечено только одним символом, то полагаем $M = V_2$;

б) если ребро V_2V_1 помечено цепочкой символов $\omega = xa$, то порождаем вершину M, делаем ее отцом вершины

V_1 и сыном вершины V_2. Ребро (V_2, M) помечаем цепочкой x , а ребро (M, V_1) - символом a.

3. Делаем вершину V θ-сыном вершины M с тем же самым позиционным номером.

4. Если у вершины V_1 только два сына, из которых первый является θ-сыном, обозначаем второго сына через A. Тогда:

а) если A - лист, то вершину A удаляем, а ее позиционный номер присваиваем вершине V_1;

б) если A - не лист и имеет единственного сына A_1, то вершину A удаляем, делаем A_1 сыном V_1, а ребро V_1A_1 помечаем конкатенацией цепочек, соответствующих ребрам V_1A и AA_1.

Ш. Для каждого префикс-идентификатора P_j из списка L2 выполняем следующие действия.

1. Обозначаем лист с меткой j через V, а его отца - через V_1.

2. Делаем a-сына вершины V_1 (т.е. вершину V) θ-сыном с тем же самым позиционным номером.

ПРИМЕР 1. Сконструируем префиксное дерево для строки Sθ (рис.6б) из префиксного дерева Saθ, изображенного на рис. 6а.

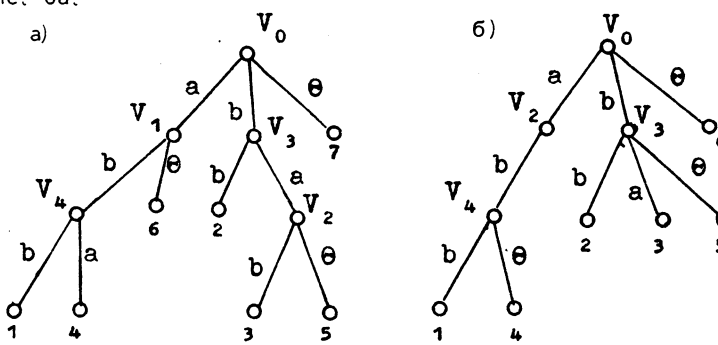


Рис. 6

Список L1 содержит префикс-идентификаторы P_6, P_5, список L2 = $\{P_4\}$. Коррекцию осуществляем в следующем порядке:

- лист 7 удаляется (см. п.I);
- лист 6 становится θ-сыном вершины V_0 (см.п.П.3);
- лист 5 становится θ-сыном вершины V_3 (см.п.П.4.а);
- лист 4 становится θ-сыном вершины V_4 (см.п.Ш).

Расположение сыновей у вершины V_1 соответствует случаю 3, у вершины V_2 - случаю 2.

Чтобы продемонстрировать корректировку вершин, имеющих θ-сына и соответствующих случаям 1 и 4, а также шаг 2 алгоритма $(A2)_2$, приведем еще один пример.

ПРИМЕР 2. Рассмотрим строку $Sb\theta = bbabbabb\theta$. Из префиксного дерева для $Sb\theta$ (рис.7а) построим префиксное дерево для $S\theta$ (рис.7б).

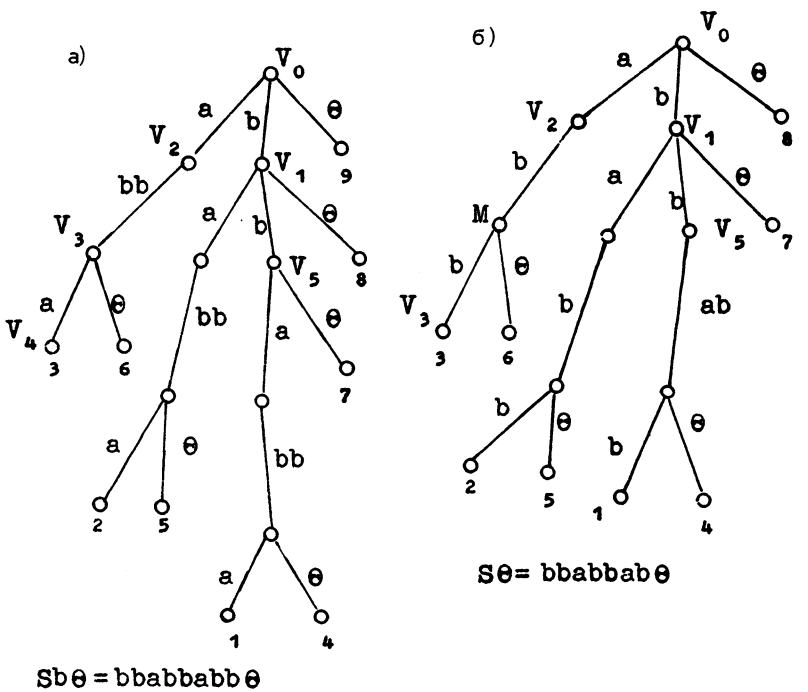


Рис. 7

Имеем список $L1=\{P_8,P_7,P_6,P_5,P_4\}$, список $L2$ пуст. Корректировку осуществляем в следующем порядке :

- лист 9 удаляется (см. п.I);
- лист 8 становится Θ-сыном вершины V_0 (см. п.П.3);
- лист 7 становится Θ-сыном вершины V_1 (см.п.П.4.б);
- лист 6 корректируем следующим образом: создается вершина M, которая является отцом вершины V_3 и сыном вершины V_2; ребра V_2M и MV_3 помечаются символом "b"; лист 6 становится Θ-сыном вершины M, вершина V_4 удаляется, а ее позиционный номер присваивается вершине V_3 (см.п.П.2.б и п.П.4.а).

Аналогично листу 6 корректируются листья 5 и 4, Расположение сыновей у вершины V_1 соответствует варианту 1, а у вершины V_5 - варианту 4.

4.3. Оценка трудоемкости алгоритма (А2). Число корректируемых концевых префикс-идентификаторов в списках $L1$ и $L2$ определяется длиной максимального префикс-идентификатора из списка $L2$ (если он не пуст), либо аналогичной характеристикой для $L1$ (когда список $L2$ пуст). Число внутренних префикс-идентификаторов, которые могут быть затронуты корректировкой, не превышает длины списка $L1$, т.е. опять же определяется длиной максимальной цепочки из $L1$. Таким образом, суммарное число производимых "актов" коррекции совпадает по порядку величины с длиной максимальной цепочки из $L2$ (или $L1$).

Каждый акт коррекции имеет константную трудоемкость (добавляется либо удаляется вершина, переобозначается ребро). Поэтому трудоемкость алгоритма $(A2)_2$ составляет $O(l(P_{max}))$, где $l(P_{max})$ - длина максимальной цепочки из списка $L2$ (или $L1$). В наихудшем случае (он реализуется, когда $S = a^N\Theta$) величина $l(P_{max})$ ограничена сверху значением N и, следо-

вательно, трудоемкость коррекции составляет $O(N)$. Однако этот случай маловероятен с практической точки зрения.

Реальный интерес представляют оценки в среднем для последовательности независимых одинаково распределенных случайных величин (эта схема в первом приближении применима к генетическим текстам). Длина максимального префикс-идентификатора лишь на единицу может превышать длину максимального повтора в последовательности S . Для случайных последовательностей с равновероятной встречаемостью элементов алфавита средняя длина максимального повтора составляет $O(\log N/\log|\Sigma|)$, где $|\Sigma|$ - число элементов алфавита [10]. Отсюда трудоемкость алгоритма коррекции префиксного дерева при устранении одного символа справа составляет для указанного класса последовательностей $O(\log N/\log|\Sigma|)$.

5. Алгоритм корректировки меры сложности (алгоритм А3)

5.1. Обоснование. Пусть мера сложности C_{i+1} для подстроки S_{i+1} , выделяемой окном D , уже определена. Необходимо скорректировать C_{i+1} таким образом, чтобы получить меру сложности для подстроки S_i . Заметим, что при сдвиге окна могут измениться наряду с первым и последним компонентами также некоторые внутренние компоненты истории (см. далее пример 3).

С помощью алгоритма А3 формируется список позиций $L3 = \{j_1, j_2, \ldots, j_m\}$, соответствующих началам корректируемых участков в разложении $H(S_{i+1})$. Элементы списка $L3$ упорядочены по убыванию. Каждый из корректируемых участков содержит целое число компонентов разложения $H(S_{i+1})$ (не меньше двух). После коррекции число компонентов в рассматриваемом участке (это будут уже компоненты истории $H(S_i)$) может остаться без изменения либо уменьшиться на единицу.

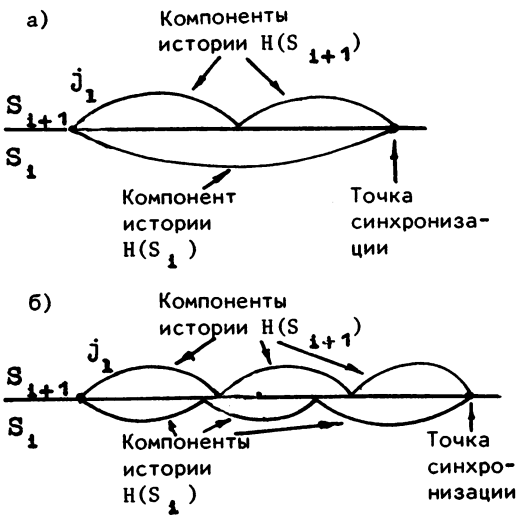


Рис. 8

Собственно, коррекция для произвольного участка с номером $1, 1 \le 1 \le m$, сводится к определению по префиксному дереву T_i длины очередного компонента истории $H(S_i)$, начинающегося с позиции j_1. Если конец этого компонента в разложении $H(S_i)$ совпадает с концом соответствующего компонента в $H(S_{i+1})$ (см. рис. 8а), то происходит "синхронизация" разложений $H(S_{i+1})$ и $H(S_i)$ и дальше они вновь будут совпадать вплоть до позиции с номером j_{i+1}. Точка синхронизации определяет правую границу корректируемого участка.

Если после первого шага не происходит синхронизации двух разложений (см. рис.8б), по дереву T_i определяется очередной компонент истории $H(S_i)$ и т.д. до тех пор, пока не выполнится условие синхронизации. Заметим, что в подавляющем большинстве случаев на практике точка синхронизации достигается через один-два шага.

Опишем теперь принцип формирования списка L3. Обозначим через "A" символ, добавляемый слева к строке S_{i+1} при сдвиге окна вдоль последовательности S на одну позицию. Это будет первый символ строки S. Нетрудно видеть, что в связи с добавлением символа "A" измениться могут (кроме граничных)

лишь те компоненты истории $H(S_{i+1})$ (причем не обязательно все), которые являются префиксами префикс-идентификатора позиции i в строке S_i, а также следующие за ними компоненты, дополняющие корректируемый участок до ближайшей точки синхронизации. Поэтому основой для формирования списка $L3$ служит фрагмент дерева T_i, включающий в себя префикс-идентификатор позиции i.

Будем предполагать, что само дерево T_i уже получено путем коррекции дерева T_{i+1}, но значения величин $J(V)$ (см. алгоритм A1) еще не откорректированы, за исключением вершины, соответствующей листу с номером i (это новая вершина, которой в T_{i+1} не было; для нее $J(V_i) = i$). Пусть $(V_i = V_{i_1}), V_{i_2}, \ldots, (V_{i_l} = V_0)$ - цепочка вершин в дереве T_i, через которые проходит путь из листа V_i с номером i в корень дерева V_0. С учетом сделанного выше замечания значения $J(V_{i_2}), \ldots, J(V_{i_{l-1}})$ характеризуют первые вхождения подстрок, соответствующих вершинам $V_{i_2}, \ldots, V_{i_{l-1}}$, в строку S_{i+1}. Именно подстроки с указанными адресами могут служить потенциально возможными началами участков коррекции.

Действительно, после добавления символа A слева эти подстроки, рассматриваемые уже как составляющие строки S_i, утрачивают свой статус "подстрок с первым вхождением". Все они, в принципе, могут быть скопированы с соответствующих префиксов первого префикс-идентификатора строки S_i. Реально же такое копирование имеет место лишь тогда, когда начало подстроки совпадает с началом какого-либо компонента истории $H(S_{i+1})$. Если при этом подстрока состоит не менее чем из двух символов, то указанный компонент при копировании подстроки удлиняется, поскольку правая его граница лежит внутри подстроки (иначе это не была бы подстрока с первым вхождением в S_{i+1}).

ПРИМЕР 3. Пусть S_{i+1} и S_i - две подстроки, выделяемые рамкой размера $D = 23$ из текста S.

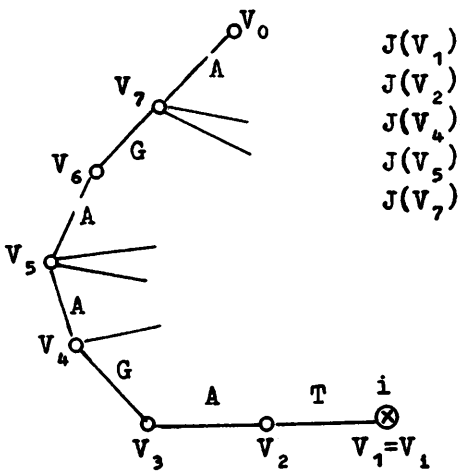
Позиция i

$$H(S_i) = A\cdot G\cdot A\cdot \begin{bmatrix} AGA \\ A\cdot GA \end{bmatrix}\cdot T\cdot T\cdot GAAGA\cdot A\cdot C\cdot C\cdot \begin{bmatrix} AGAAGA \\ AGA\cdot AGA \end{bmatrix}\cdot C;$$

$$H(S_{i+1}) = G\cdot A\cdot \begin{bmatrix} A \\ A\cdot GA \end{bmatrix}\cdot T\cdot T\cdot GAAGA\cdot A\cdot C\cdot C\cdot \begin{bmatrix} AGAAGA \\ AGA\cdot AGA \end{bmatrix}\cdot CC.$$

$J(V_7)$; $J(V_5)=J(V_6)=i+3$; $J(V_4)=i+10$; $J(V_2)=J(V_3)=i+16$.

Префикс-идентификатором позиции i в строке S_i является подстрока P_i = AGAAGAT. Фрагмент дерева префикс-идентификаторов T_i, включающий в себя P_i, изображен на рис. 9.



$J(V_1)=i$ – новое значение

$J(V_2)=J(V_3)=i+16$;
$J(V_4)=i+10$;
$J(V_5)=J(V_6)=i+3$;
$J(V_7)=i+2$

} значения $J(V)$ для S_{i+1}

Рис. 9

Потенциально возможные начала участков коррекции, соответствующие первым вхождениям в S_{i+1} префиксов префикс-идентификатора P_i, помечены стрелками снизу. Реально в список I3 попадают лишь участки с адресами $J(V_5)=i+3$ и $J(V_2)=i+16$ (они выделены квадратными скобками). Участок с адресом $J(V_4)=i+10$ не корректируется, так как не является на-

чалом компонента. Участок с адресом $J(V_7)=i+2$ не корректируется, так как содержит всего один символ. Подстроки, соответствующие вершинам V_2 и V_3 (а также V_5 и V_6), имеют одинаковое первое вхождение, поэтому корректируются вместе.

С учетом сделанных пояснений схема коррекции выглядит следующим образом. Пусть i_0 - позиция, указывающая первое вхождение в строку S_{i+1} добавляемого слева символа. Просматриваем в дереве T_i цепочку вершин $V_{i_2},\ldots,V_{i_{l-1}}$ с еще неоткорректированными значениями $J(V)$. Для каждой вершины $V_{i_k}, 2 \le k \le l-1$, сравниваем значение $J(V_{i_k})$ с двумя величинами: i_0 и $J(V_{i_{k-1}})$. Если $J(V_{i_k})$ является началом компонента в разложении $H(S_{i+1})$ и $J(V_{i_k}) \neq i_0$, и $J(V_{i_k}) \neq J(V_{i_{k-1}})$, то заносим значение $J(V_{i_k})$ в список $L3$. Формирование списка $L3$ заканчивается, как только впервые встретится вершина со значением $J(V_{i_k})$, равным i_0. Если этой вершине соответствует строка из более чем одного символа, то значение i_0 также заносится в список $L3$ и завершает его формирование. Затем осуществляется коррекция величин $J(V_{i_k})$: все они полагаются равными i.

Если путь из листа в корень проходит всего по двум вершинам V_i и V_0, это означает, что символ "A" не встречается в строке S_{i+1} и, следовательно, в $H(S_{i+1})$ отсутствуют компоненты, подлежащие коррекции.

Остается добавить, что при сдвиге окна влево на один символ добавляется новый (первый) компонент сложности. Кроме того, если длина последнего компонента в разложении $H(S_{i+1})$ равнялась единице, то при сдвиге окна он исчезает и соответственно C_i уменьшается на 1.

5.2. Приведем структурную схему алгоритма корректировки меры сложности.

Алгоритм А3.

I. Пусть V_0 – корень дерева T_i, а W и W_1 – пара вершин дерева таких, что W_1 является отцом W. Полагаем $W = V_i$.

II. Формирование списка $L3$.

До тех пор пока $W_1 \neq V_0$ или $J(W) \neq i_0$, выполняем следующие действия:

1. Если $J(W_1) \neq J(W)$ и $J(W_1)$ соответствует началу компонента в разложении $H(S_{i+1})$, то заносим $J(W_1)$ в список $L3$;

2. $W = W_1$; W_1 есть отец вершины W.

III. Коррекция значений $J(W)$.

1. Полагаем $W = V_i$; W_1 является отцом вершины W.

2. До тех пор пока $W_1 \neq V_0$, выполняем следующие действия:

а) $J(W_1) = i$;

б) $W = W_1$; W_1 есть отец вершины W.

IV. Корректировка меры сложности.

1. Обозначим через C_i и C_{i+1} значения меры сложности для строк S_i и S_{i+1} соответственно.

2. Если последний компонент в разложении $H(S_{i+1})$ больше 1, то $C_i = C_{i+1} + 1$, иначе $C_i = C_{i+1}$.

3. Для каждого значения $J_k(W)$ из списка $L3$ ($k=1,\ldots,l$) (в порядке возрастания позиций) выполняем с помощью алгоритма А1 разложение на компоненты сложности фрагмента, начинающегося с указанной позиции и заканчивающегося точкой синхронизации. Если число компонентов в новом и старом разложении различно, уменьшаем значение сложности C_i на единицу. Если начало $J_k(W)$ следующего участка коррекции находится внутри

предыдущего участка коррекции, то разложение на компоненты сложности с позиции $J_k(W)$ не выполняется.

5.3. Оценка трудоемкости алгоритма коррекции меры сложности. Количество потенциально возможных участков коррекции ограничено сверху длиной префикс-идентификатора i-й позиции в строке S_i. Реально корректируемых участков будет значительно меньше из-за дополнительных ограничений на отбираемые значения $J(V)$, что сильно затрудняет получение оценок в среднем. Заведомо завышенной оценкой средней длины префикс-идентификатора может служить оценка средней длины максимального повтора в последовательности длины D, выделяемой сколь-зящим окном. Она имеет порядок $\log D/\log|\Sigma|$.

Каждая коррекция сводится к определению одного-двух (гораздо реже - большего числа) компонентов сложности по дереву T_i. Многочисленные эксперименты, проводившиеся при рамках разного размера, не выявили зависимости среднего числа корректируемых компонентов от величины D. Поэтому в первом приближении будем оценивать этот параметр константой.

Поиск компонента сложности по дереву T_i связан с просмотром одной из его ветвей (см. алгоритм А1). Трудоемкость поиска в среднем составляет $O(\log D)$.

Суммарная трудоемкость коррекции в среднем будет иметь порядок $\log^2 D/\log|\Sigma|$. В наихудшем случае (специально сконструированная последовательность) трудоемкость коррекции может линейным образом зависеть от размера окна D. Сопоставление с алгоритмом А2 показывает, что процедура коррекции меры сложности дает превалирующий вклад в суммарную оценку трудоемкости вычисления сложностного профиля. Последняя, с учетом вышесказанного, имеет порядок $N\cdot\log^2 D/|\Sigma|$.

З а к л ю ч е н и е

Предложен алгоритм вычисления сложностного профиля символьных последовательностей большой длины. Трудоемкость алгоритма в среднем составляет $O(N \cdot \log^2 D/|\Sigma|)$, где N - длина текста, D - размер скользящего окна анализа, $|\Sigma|$ - мощность алфавита.

В основу алгоритма положена конструкция, называемая "деревом префикс-идентификаторов". Самостоятельный интерес (помимо алгоритма в целом) представляют и отдельные его блоки, такие, в частности, как: а) блок построения префиксного дерева и вычисления с его помощью меры сложности конечной последовательности; б) блок корректировки префиксного дерева, соответствующего фрагменту, выделяемому из текста скользящим окном анализа; в) блок корректировки меры сложности.

Л и т е р а т у р а

1. ГУСЕВ В.Д. Сложностные профили символьных последовательностей //Настоящий сборник. - С. 35-63.

2. LEMPEL A., ZIV J. On the Complexity of Finite Sequences //IEEE Trans. on Inf. Th. - 1976. -Vol. IT-22, N1. -P.75-81.

3. ГУСЕВ В.Д., ЧУПАХИНА О.М., КУЛИЧКОВ В.А. Сложностные профили - новый метод обнаружения структурных закономерностей в первичных структурах НК-молекул и белков //3 Всесоюзн. совещание "Теоретические исследования и банки данных по молекулярной биологии и генетике". Новосибирск, июль 1988 г.: Тез.докл. - Новосибирск, 1988. -С. 38-39.

4. ГУСЕВ В.Д., КУЛИЧКОВ В.А., ЧУПАХИНА О.М. Сложностной анализ генетических текстов (на примере фага λ). - Новосибирск, 1989. - 49 с. - (Препринт/АН СССР, Сиб. отд-ние.Институт математики; № 20).

5. ГУСЕВ В.Д. Механизмы обнаружения структурных закономерностей в символьных последовательностях //Проблемы обработки информации. - Новосибирск. - 1983.- Вып. 100: Вычислительные системы. -С. 47-66.

6. MAJSTER M.E., REISER A. Efficient On-line Construction and Correction of Position Trees //SIAM J. Comput. - 1980. - Vol. 9, N 4. -P. 785-807.

7. BLUMER J. How Much is That DAWD in the Window? A Moving Window Algorithm for the Directed Acyclic Word Graph // Journal of Algorithms. - 1987. -Vol. 8. - P. 451-469.

8. WEINER P. Linear Pattern Matching Algorithms //Conf.Record, IEEE 14th Annual Symposium on Switching and Automata Theory, 1973. - P. 1-11.

9. АХО А., ХОПКРОФТ Дж., УЛЬМАН Дж. Построение и анализ вычислительных алгоритмов. - М.: Мир, 1979.

10. ЗУБКОВ А.М., МИХАЙЛОВ В.Г. Предельные распределения случайных величин, связанных с длинными повторениями в последовательности независимых испытаний //Теория вероятностей и ее применения. - 1974. -Т. XIX, №1. -С. 173-181.

Поступила в ред.-изд.отд.
23 ноября 1989 года