АНАЛИЗ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ
(Вычислительные системы)
1985 год Выпуск 113

УДК 519.766.48

АЛГОРИТМЫ ПОИСКА НЕСОВЕРШЕННЫХ ПОВТОРОВ В ГЕНЕТИЧЕСКИХ ТЕКСТАХ

В.Д.Гусев, В.А.Куличков, А.Е.Никулин

Введение

Под генетическим текстом будем понимать представление молекул нерегулярных полимеров (ДНК, РНК, белков) в виде упорядоченной последовательности мономеров (нуклеотидов или аминокислот). Алфавит нуклеотидов состоит из 4 символов, алфавит аминокислот - из 20. Генетические тексты, описывающие молекулы ДНК, РНК и белков, называют часто первичными структурами соответствующих молекул, подразумевая при этом их линейное представление, а не пространственную конфигурацию, с которой связаны термины "вторичная структура", "третичная" и т.д.

Повторы являются важными структурными элементами генетических текстов. Они играют большую роль в формировании вторичных структур молекул нерегулярных полимеров, в эволюционных перестройках этих молекул и т.д. Наряду с идеальными (совершенными) повторами, образуемыми точно совпадающими участками текста, часто встречаются участки текста, близкие в том или ином смысле (например, в смысле хэммингова расстояния или редакционного [I]). Такие участки естественно называть несовершенными повторами.

Как правило, несовершенные повторы образуются из совершенных в результате искажающих воздействий, приводящих к замене отдельных символов другими (единичными), а также вставке новых символов или устранению уже имеющихся. В данной работе рассматриваются несовершенные повторы, образуемые парами участков, переводимых друг в друга лишь одними заменами символов. Интерес к данному классу повторов вызван тем, что частота замен в ходе эволюции генетиче-

ских макромолекул значительно выше, чем частота вставок и устранений символов, т.е. несовершенные повторы такого вида превалируют над остальными. В то же время трудоемкость отыскания таких повторов может быть существенно уменьшена по сравнению с общим случаем за счет нескольких факторов, одним из которых является относительная простота вычисления хэммингова расстояния по сравнению с редакционным.

Целью данной работы является построение алгоритма отыскания в достаточно длинных текстах ($N \sim 10^3$-10^5 символов) <u>всех</u> несовершенных (в указанном выше смысле) повторов фиксированной длины с заданной мерой различия. В идейном отношении он близок к матричным алгоритмам отыскания гомологичных участков с "диагональной фильтрацией". В наиболее четкой (хотя и не доведенной до рекуррентного соотношения) форме идея диагональной фильтрации высказана в работе [2]. В настоящей работе эта идея получает свое дальнейшее развитие в плане введения элементов пошаговой адаптации в поисковую процедуру, что позволяет во многих случаях добиться значительного сокращения трудоемкости по сравнению с алгоритмами матричного типа.

Алгоритм представлен в двух формах. Одна из них ориентирована на генетические приложения и в ней существенно используется фактор малости алфавита. Другая имеет очень простую логическую структуру и в ней отсутствует ограничение на мощность алфавита, но она проигрывает первой по трудоемкости. Отдельно рассмотрен случай поиска мало различающихся несовершенных повторов.

Все другие (отличные от матричных) алгоритмы отыскания несовершенных повторов, близких в смысле хэммингова расстояния, можно условно отнести к алгоритмам "ядерного" типа (см., например, [3-5]). Ядром несовершенного повтора, образуемого двумя словами одинаковой длины, естественно назвать пару максимальных по длине и одинаково расположенных участков этих слов, не искаженных заменами, т.е. образующих идеальный повтор. В алгоритмах ядерного типа вначале отыскивают все потенциально возможные ядра, т.е. идеальные повторы заданной длины, а затем анализируют окрестности одинаковых ядер с целью обнаружения несовершенных повторов. Выделение ядер сильно уменьшает круг потенциальных претендентов на образование несовершенного повтора, если размер ядра достаточно велик. К тому же процедура отыскания ядер достаточно проста. Соответствующая техника, основанная на идеях хеширования, описана, например, в [6].

Основным недостатком известных ядерных алгоритмов является тот (подчас завуалированный) факт, что снижение их трудоемкости достигается за счет отказа от выявления всех несовершенных повторов заданного типа. Это происходит при необоснованном завышении размера ядра и фиксации его положения внутри несовершенного повтора. Снятие этих ограничений увеличивает объем перебора и может сделать эти алгоритмы более трудоемкими, чем матричные. Сохранение же ограничений приводит к потере значительного количества несовершенных повторов. К примеру, авторы работ [4,5] по итогам обработки первичной структуры σ-субъединицы РНК-полимеразы E·coli (длина 1839 символов) обнаружили в ней всего 8 несовершенных повторов длины 20 с 5 несовпадениями, включая и те, что входят в состав более длинных несовершенных повторов (см.табл.2 из [4]). Фактическое же число (20,5)-повторов в этом тексте равно 53; кроме них имеются 15 повторов длины 20 с 4 несовпадениями и 1 повтор длины 20 с тремя несовпадениями.

1. Обозначения и определения

Введем следующие обозначения:

S - алфавит (конечное множество символов);

$n = |S|$ - мощность алфавита;

$p(\alpha)$ - вероятность появления в тексте элемента α алфавита;

T - текст над алфавитом S (конечная последовательность символов из S);

N - длина текста;

T[i] - i-й по порядку элемент текста (или элемент, занимающий i-ю позицию, $1 \le i \le N$);

$a_l(i)$ - слово длины l, начинающееся с i-й позиции текста;

$d_l(i,j)$ - обобщенное хэммингово расстояние между словами $a_l(i)$ и $a_l(j)$ (число позиций с несовпадающими символами); несовершенный (с точностью до замен) повтор или (l,k)-повтор - пара слов $a_l(i)$ и $a_l(j)$, такая, что $d_l(i,j)=k$ $(1 \le k \le l)$. При k=0 имеем совершенный или (l,0)-повтор.

Рассматривается задача отыскания в произвольном тексте T при фиксированных значениях l и k всевозможных пар слов, удовлетворяющих определению (l,k)-повтора. Параметры l и k целесообразно выбирать такими, чтобы ожидаемое число (l,k)-повторов для случайного текста с теми же характеристиками N и $\mathcal{P} = \{p(\alpha) \mid \alpha \in S\}$, что и у исследуемого, было невелико (порядка 1). Формально, предполагая, что $l << N$, это условие можно записать в виде:

$$E\xi_N(l,k) = C_N^2 \cdot \pi(l,k) = C_N^2 \cdot C_l^k \, P_2^{l-k}(1-P_2)^k \le 1, \qquad (1)$$

где $\xi_N(l,k)$ - случайная величина, численно равная количеству (l,k)-повторов в последовательности из N независимых испытаний, E - знак математического ожидания, $\pi(l,k)$ - вероятность того, что два случайно выбранных l-слова различаются по k позициям, $P_2 = \sum_{\alpha \in S} p^2(\alpha)$ - вероятность совпадения любой пары символов из сравниваемых l-слов. Левая часть (I) есть частный случай приведенного в [7] более общего выражения для $E\xi_N(m,l,k)$ при $m=2$, где $\xi_N(m,l,k)$ - случайная величина, равная числу всех m-кратных (l, k)-повторов в последовательности из N испытаний. Соблюдение условия (I) позволяет с большой вероятностью трактовать выделяемые (l, k)-повторы как неслучайные (функционально значимые).

2. Взаимосвязь с задачей отыскания идеальных повторов (метод перекодировок)

Рассмотрим возможные схемы выделения (l,k)-повторов при наложении ограничений на некоторые параметры задачи. Эти ограничения естественным образом возникают в ряде приложений. Техника, традиционно используемая для отыскания совершенных повторов (хеширование, суффиксные и префиксные деревья, конечные автоматы), в силу ряда причин не подходит для отыскания несовершенных повторов. Тем не менее, иногда удается свести вторую задачу к первой и воспользоваться известными методами. Некоторые из рассматриваемых ниже случаев показывают, что трудоемкость процедуры сведения существенным образом зависит от таких параметров задачи, как k, n и l. Резкой границы (по трудоемкости) между задачами отыскания совершенных и несовершенных повторов не существует. Трудоемкость монотонно возрастает с увеличением параметра k.

Случай I. Длина текста N мала. Наиболее естественно в этой ситуации воспользоваться прямым алгоритмом отыскания (l,k)-повторов, не прибегая к оптимизации. Слово $a_1(1)$ текста сравнивается посимвольно со словами $a_1(2), a_1(3), \ldots, a_1(N-l+1)$. Сравнение элементов каждой пары слов продолжается до тех пор, пока текущее расстояние $d_s(1,j) < k+1$, $j = 2, \ldots, N-l+1$, $s \le l$. Если $s=l$ и $d_s(1,j) = k$, фиксируем наличие (l,k)-повтора. Аналогичным образом слово $a_1(2)$ текста сравнивается со всеми оставшими -

ся, затем $a_1(3)$ и т.д. до $a_1(N-1)$. Трудоемкость такой процедуры в наихудшем случае составляет $O(N^2 l)$.

При отыскании (l,k)-повторов в одном тексте по описанной схеме построчно определяются (точнее ранжируются по числу несов - падений) лишь наддиагональные элементы матрицы расстояний. Можно рассматривать и такие (l,k)-повторы, элементы которых разнесены по двум разным текстам, т.е. $a_1(i) \in T_1$, $1 \le i \le N_1-l+1$, $a_1(j) \in T_2$, $1 \le j \le N_2-l+1$ и $d_1(i,j) = k$. В этом случае определяются все элементы матрицы расстояний ($d_1(i,j)$ уже может отличаться от $d_1(j,i)$).

Случай 2. Параметры n и k малы. Задача отыскания несовершенных повторов может быть сведена к задаче отыскания совершенных повторов соответствующими перекодировками элементов алфавита. Рассмотрим, к примеру, случай нуклеотидных последовательностей ($S = \{A,Г,Т,Ц\}$, $n = 4$).

При k=1 два l-слова, образующие (l,1)-повтор, будут иметь несовпадающие символы в одной из l позиций. Существует C_n^2 возможных типов несовпадений (в нашем случае их 6: А и Г, А и Т, А и Ц, Г и Т, Г и Ц, Т и Ц). Рассмотрим C_n^2 вариантов перекодировки элементов алфавита, каждый из которых "склеивает" (делает неразличимыми) пару различных элементов алфавита, не затрагивая остальных. Применительно к нуклеотидным последовательностям это эквивалентно переходу к следующим 6 алфавитам мощности 3: А=Г,Т,Ц; А=Т, Г,Ц; А=Ц,Г,Т; Г=Т,А,Ц; Г=Ц,А,Т; Т=Ц,А,Г. Нетрудно видеть, что задача отыскания всех (l,1)-повторов в тексте Т, составленном из элементов алфавита мощности n, сводится с использованием указанных перекодировок к решению C_n^2 задач отыскания (l,0)-повторов в текстах той же длины, но с алфавитом мощности n-1 и к выбору среди полученных идеальных повторов таких, которые содержали лишь одну пару несовпадающих символов в исходном алфавите.

При k=2 у слов, образующих (l,2)-повтор, имеются несовпадающие символы в двух произвольных позициях. Из четырех символов, занимающих указанные позиции, могут быть два различных, три или все четыре. Несовпадения первого типа требуют "склеивания" лишь двух элементов алфавита. Этот случай был рассмотрен выше (k=1), но там соответствующие (l,2)-повторы отсеивались. Несовпадения второго типа требуют "склеивания" сразу трех элементов алфавита, т.е. необходимо просмотреть C_n^3 вариантов перекодировок. Несовпадения третьего типа требуют "склеивания" двух разных пар элемен -

тов алфавита, что дает дополнительно $\sum_{k=1}^{n-3}(n-k)C^{2}_{n-k-1}$ вариантов перекодировок.

Применительно к первичным структурам НК-молекул суммарное количество перекодировок, необходимых для выявления всех (1,2)-повторов, равно 7. Это перекодировки вида А=Г=Т,Ц; А=Г=Ц,Т; А=Т=Ц,Г; Г=Т=Ц,А (второй тип несовпадений) и А=Г, Т=Ц; А=Т, Г=Ц; А=Ц, Г=Т (третий тип). Несовпадения первого типа также выявляются с помощью указанных перекодировок.

При k=3, n=4 уже невозможно выявить все (1,3)-повторы, используя метод перекодировок. Действительно, два 1-слова с тремя несовпадениями типа А,Т; Т,Г; Г,Ц могут быть преобразованы в идеальный повтор лишь перекодировкой вида А=Т, Т=Г, Г=Ц, что эквивалентно переходу к алфавиту мощности I, где тождественными окажутся все 1-слова текста.

Выход из положения в данной ситуации может заключаться в представлении (1,k)-повтора в виде конкатенации (l_1,k_1)- и (l_2, k_2)-повторов, где $l = l_1+l_2$, $k = k_1+k_2$. К примеру, (20,3)-повтор может быть представлен в форме (I0,0) ⊕ (I0,3)-повтора, где " ⊕ " - знак операции конкатенации, либо в форме (I0,I) ⊕ (I0,2)-повтора. В первом случае для выявления (20,3)-повторов нужно найти все (I0,0)-ядра (идеальные повторы длины I0) и отобрать среди них те пары, которые при расширении вправо или влево на I0 символов дадут по три несовпадения. Во втором случае нужно найти все (I0,I)-повторы и отобрать среди них те, которые при расширении увеличат число несовпадений не более чем на 2.

При k = 4 и 5 пришлось бы уже рассмотреть 3 варианта распределения несовпадений по обеим половинкам 1-слова: 4=4+0=3+I=2+2, 5=5+0=4+I=3+2. При k = 6 уже возник бы вариант типа 3+3, и мы вынуждены были бы применить рассмотренный прием к каждой из половинок в отдельности, что эквивалентно членению 1 на большее чем два количество слагаемых. Уменьшение длины отыскиваемых повторов приводит к увеличению их числа и, следовательно, к увеличению объема последующего перебора.

Описанная схема обнаружения (1,k)-повторов в общем случае имеет скорее методологическое, чем практическое значение. Она хорошо иллюстрирует связь между задачами отыскания совершенных и несовершенных повторов, рост вычислительных трудностей с увеличением значений n и k и универсальный характер метода перекодировок как средства обнаружения несовершенных повторов (дополнительные примеры на эту тему приведены в [8]).

Случай 3. Величина l мала. Несовпадения могут быть размещены по длине (l,k)-повтора C_l^k способами. Каждому способу соответствует свое множество (l,k)-повторов, образуемое l-словами с идентичным заполнением (l-k) фиксированных позиций. Все такие слова могут быть найдены за один просмотр исходного текста с помощью процедуры хеширования каждого из его l-слов, причем аргументом функции расстановки является не все l-слово, а лишь (l-k) составляющих его символов, расположенных в одинаковых (для каждого из l-слов) позициях. Любая пара из отобранного множества l-слов будет образовывать $(l, \le k)$-повтор, т.е. повтор с числом несовпадений, не превышающим k. Среди них легко отобрать искомые (l,k)-повторы. Подобную процедуру следует повторить C_l^k раз, меняя каждый раз набор позиций, в которых допустимы несовпадения. Метод может быть использован в комбинации с методом дробления (l,k)-повторов (см. случай 2) в ситуации, когда дробление приводит к слишком коротким словам.

3. Рекуррентный алгоритм отыскания (l,k)-повторов

Можно усовершенствовать прямой алгоритм отыскания (l,k)-повторов (см. случай I в п.2), используя следующее рекуррентное соотношение, связывающее расстояния $d_l(i+1,j+1)$ и $d_l(i,j)$:

$$d_l(i+1,j+1) = \begin{cases} d_l(i,j)+1, & \text{если } T[i] = T[j] \;\&\; T[i+l] \neq T[j+l]; \\ d_l(i,j)-1, & \text{если } T[i] \neq T[j] \;\&\; T[i+l] = T[j+l]; \\ d_l(i,j) & \text{в остальных случаях.} \end{cases} \quad (2)$$

Указанное соотношение вытекает из двух других, непосредственно трактуемых исходя из определения хэммингова расстояния:

$$d_l(i,j) = d_{l-1}(i+1,j+1) + \begin{cases} 1, & \text{если } T[i] \neq T[j] \\ \text{иначе } 0; \end{cases}$$

$$d_l(i+1,j+1) = d_{l-1}(i+1,j+1) + \begin{cases} 1, & \text{если } T[i+l] \neq T[j+l], \\ \text{иначе } 0. \end{cases}$$

Соотношение (2) в неявном виде используется в упоминавшихся выше алгоритмах матричного типа с "диагональной фильтрацией". Термин "диагональный" мы используем для того, чтобы подчеркнуть, что в алгоритме, реализующем соотношение (2), таблица расстояний заполняется по диагоналям. Трудоемкость рекуррентного алгоритма состав-

ляет $O(N^2)$, т.е. снижается в 1 раз по сравнению с прямым алгоритмом. Мультипликативная константа очень мала и определяется временем реализации двух операций сравнения символов. Число сравнений, в принципе, может быть уменьшено вдвое, если заметить, что по мере продвижения вдоль текста рамки ширины 1 одни и те же символы сравниваются дважды: один раз, когда они находятся в конце 1-слова, и другой раз, когда в результате сдвига рамки они оказываются в начале 1-слова.

4. Алгоритм со "скачком"

Из соотношения (2) вытекает, что при сдвиге на один символ, т.е. при переходе от сопоставления слов $a_1(i)$ и $a_1(j)$ к сопоставлению слов $a_1(i+1)$ и $a_1(j+1)$, расстояние не может измениться более чем на 1. Отсюда следует, что при сдвиге на r символов справедливо следующее неравенство

$$d_1(i,j) - r \le d_1(i+r, j+r) \le d_1(i,j) + r. \qquad (3)$$

Неравенство (3) может быть использовано для ускорения описанного выше рекуррентного алгоритма. Действительно, если $d_1(i,j) = k+r$, т.е. расстояние между словами $a_1(i)$ и $a_1(j)$ на величину r больше критического, то для всех $1 \le s \le r-1$ имеем $d_1(i+s, j+s) > k$, а $d_1(i+r, j+r) \ge k$. Это означает, что нет необходимости сравнивать между собой слова $a_1(i+1)$ и $a_1(j+1)$, $a_1(i+2)$ и $a_1(j+2)$, ..., $a_1(i+r-1)$ и $a_1(j+r-1)$, а можно сразу перейти к сопоставлению слов $a_1(i+r)$ и $a_1(j+r)$, т.е. совершить скачок по диагонали (отсюда и название алгоритма).

Сама возможность совершения скачка не влечет автоматически снижения трудоемкости, поскольку при сдвиге более чем на один символ мы уже не в состоянии воспользоваться соотношением (2). Требуются специальные схемы реализации скачка. Две из них изложены ниже.

I. Машинно-независимая схема реализации скачка. Будем сравнивать 1-слова не слева направо, а справо налево, начиная с последнего символа. Сравнение слов $a_1(i)$ и $a_1(j)$ продолжается до тех пор, пока текущее расстояние $d_s(i_1, j_1) \le k+1$ ($s = 1, \ldots, l$; $i_1 = i+l-s$; $j_1 = j+l-s$). Если $s=l$ и $d_1(i,j) \le k$, то слова $a_1(i)$ и $a_1(j)$ есть искомые (в смысле $(l, \le k)$-повтора), и сравниваются следующие два слова: $a_1(i+1)$ и $a_1(j+1)$. Если же $s \le l$

и $d_s(i_1, j_1) = k+1$, то осуществляется скачок через (l-s+1) символ и начинается сравнение слов $a_1(i+1-s+1)$ и $a_1(j+1-s+1)$ (рис.I, заштрихованы те участки l-слов, символы которых сопоставляются друг с другом).

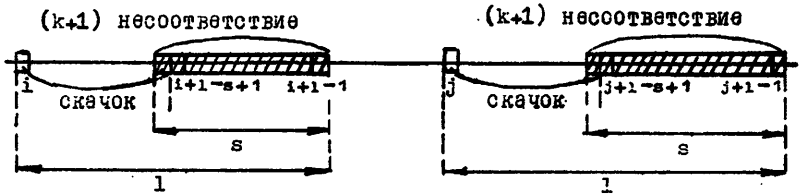

Рис.I

Трудоемкость описанной схемы уже будет существенным образом зависеть от l и k. При не слишком больших по сравнению с l значениях k каждая диагональ заполняется с сублинейной трудоемкостью, т.е. число сравнений оказывается меньшим, чем длина диагонали. Трудоемкость алгоритма в целом составляет $O(N^2/c(l,k))$, где $c(l,k)$ - выигрыш по сравнению с рекуррентным алгоритмом.

Оценим среднее значение $c(l,k)$ для последовательности независимых испытаний с заданным распределением вероятностей $\mathcal{P}=\{p(\alpha)\}$ исходов α ($\alpha \in S$). Такая модель может рассматриваться в качестве весьма грубого (но достаточного для наших целей) приближения генетических текстов.

Пусть $a_1(i)$ и $a_1(j)$ - сравниваемые (с конца) слова длины l, а s - случайная величина, равная длине минимального участка этих слов, на котором набирается точно (k+1) несовпадений (см. рис.I). Вычислим E(s) - математическое ожидание величины $s = s(k+1)$. Требование минимальности s означает, что (k+1)-е несовпадение должно быть зафиксировано в s-й по счету паре сравниваемых элементов. Остальные k несовпадений могут быть распределены по длине (s-1) C^k_{s-1} различными способами. Вероятность того, что два случайных слова длины (s-1) отличаются по k позициям, равна

$$C^k_{s-1}\, P_2^{s-1-k}(1-P_2)^k,$$

где $P_2 = \sum_{\alpha \in S} p^2(\alpha)$ - вероятность совпадения двух сравниваемых символов. Вероятность того, что два случайных слова длины s, различающиеся по последней позиции, в целом различаются по (k+1) позиции, есть $P(s,k+1) = C^k_{s-1} P_2^{s-1-k}(1-P_2)^{k+1}$. Таким образом, $s = k+1$ с вероятностью $P(k+1,k+1)$, $s=k+2$ с вероятностью $P(k+2\ k+1)$ и т.д., откуда

$$E(s(k+1)) = \sum_{s=k+1}^{\infty} s \cdot P(s,k+1) = \left(\frac{1-P_2}{P_2}\right)^{k+1} \cdot \sum_{s=k+1}^{\infty} s \cdot C_{s-1}^{k} P_2^{s}. \qquad (4)$$

Основываясь на (4), можно реальную схему движения вдоль последовательности со скачками переменной величины заменить условной (усредненной) с одинаковыми скачками (рис.2, стрелками указаны места перескоков; заштрихованы участки текста, подвергшиеся обработке).

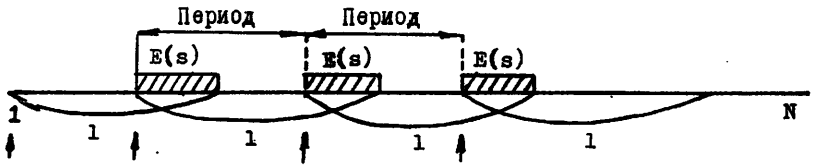

Рис.2

В соответствии с рис.2 текст представлен последовательностью периодов, каждый из которых состоит из обрабатываемого участка длины E(s) и необрабатываемого длины 1 - 2E(s). Обработка одного периода в рекуррентном алгоритме требует $2\cdot(E(s) + 1 - 2E(s)) = 2\cdot(1-E(s))$ сравнений, а в алгоритме со скачком - E(s) сравнений. Отсюда средний выигрыш алгоритма со скачком по сравнению с рекуррентным составляет величину

$$c(1,k) = \frac{2(1 - E(s(k+1)))}{E(s(k+1))}. \qquad (5)$$

ПРИМЕР I. Пусть $n=4$, $p(\alpha) = \frac{1}{4}$ для всех $\alpha \in S$, $1 = 16$, $k = 4$. Тогда $E(s(5)) \approx 7$, $C(16,4) \approx 2{,}6$, т.е. число сравнений уменьшается по сравнению с рекуррентным алгоритмом примерно в 2,6 раза. С увеличением 1 при фиксированном k выигрыш растет линейно.

Достоинством описанной схемы реализации скачка является ее простота. Недостаток же схемы в том, что с увеличением k необрабатываемые участки текста (на рис.2 они расположены между заштрихованными участками) уменьшаются. При достаточно больших по сравнению с 1 значениях k длины этих участков становятся равными нулю, т.е. алгоритм перестает давать выигрыш по сравнению с рекуррентным. Приравнивая (5) единице, можно для фиксированного 1 определить критическое значение k и наоборот. Заметим, что повторы

со значениями k порядка критического и выше, как правило, не представляют практического интереса, поскольку для них не выполняется условие (I).

Эффективность алгоритма, оцениваемую числом сравнений пар символов, можно повысить, усложнив схему сравнений путем фиксации номеров позиций, по которым обнаружены несовпадения символов. Хотя усложнение логики сравнений требует дополнительных затрат памяти и времени, это, как правило, компенсируется уменьшением числа сравнений, которое для данной схемы при любом k меньше, чем в рекуррентном алгоритме. Схема допускает разные модификации, но идея остается прежней: как только текущее расстояние между двумя сравниваемыми словами становится равным (k+1), осуществляется переход (скачок) к ближайшей паре слов, для которой текущее расстояние равно k. Под текущим расстоянием $d_s(i, j)$, где $s \le l$, здесь понимается число несовпадений, зафиксированных при сопоставлении s из l пар символов, но сопоставляемые символы расположены не обязательно подряд.

Элементарный шаг алгоритма выглядит следующим образом. Пусть при сравнении слов $a_l(i)$ и $a_l(j)$ выполнилось условие $d_s(i,j) = k+1$ $(s \le l)$. Обозначим через $i_1,\dots,i_{k+1}$; $j_1,\dots,j_{k+1}$ упорядоченные по возрастанию номера позиций, по которым обнаружены несовпадения в i-м и j-м словах соответственно ($i_1-i = j_1-j$, $i_2-i_1 = j_2-j_1$ и т.д.). Поскольку число несовпадений превысило критическое, переходим к анализу слов $a_l(i_1+1)$ и $a_l(j_1+1)$, учитывая, что k несовпадений между ними уже зафиксированы. Сравниваем символы $T[i_1+1]$ и $T[j_1+1]$. При этом возможны два случая:

а) $T[i_1+1] \neq T[j_1+1]$. В этой ситуации переходим к сопоставлению слов $a_l(i_2+1)$ и $a_l(j_2+1)$ по вышеописанной схеме;

б) $T[i_1+1] = T[j_1+1]$. Если не все символы слов $a_l(i_1+1)$ и $a_l(j_1+1)$ проверены, возвращаемся назад и начинаем их проверять. Как только обнаруживается несовпадение, переходим к анализу слов с номерами, на единицу большими, чем номера позиций ближайших к i_1 и j_1 несовпадений. Если все символы между i_1 и i_2 (соответственно j_1 и j_2) были проверены ранее, это будут слова $a_l(i_2+1)$ и $a_l(j_2+1)$, в противном случае несовпадение может быть обнаружено и между ними.

Если при возврате назад несовпадений не обнаружено, то пара слов $a_l(i_1+1)$ и $a_l(j_1+1)$ является искомой. Сдвигаемся на один символ, уменьшаем на единицу текущее расстояние, если $T[i_1+1] \neq$

$\neq T[j_1+1]$, и продолжаем процесс, сравнивая уже слова $a_1(i_1+2)$ и $a_1(j_1+2)$.

Различные модификации описанной схемы отличаются, главным образом, порядком просмотра непроверенных символов при возврате назад (см. п. "б"). Одна из таких модификаций реализована в дипломной работе Семенихиной И.В. (НГУ, 1985 г.). Там же приведены сравнительные оценки трудоемкости.

2. Машинно-ориентированная схема реализации скачка. Алгоритмы, описанные в предыдущем разделе, носят универсальный характер. У них нет ограничений на величину l, мощность алфавита n и они могут быть реализованы на любой ЭВМ.

Однако использование специфики генетических текстов (малость алфавита) и особенностей конкретной ЭВМ может способствовать значительному ускорению алгоритма. Идея ускорения заключается в вычислении хэммингова расстояния между парой l-слов не посимвольно, а сразу для достаточно большой группы символов, упакованных в одно машинное слово. Чем больше разрядность машинного слова и чем меньше мощность алфавита, тем больший достигается выигрыш. Определенное значение имеет специфика команд конкретной ЭВМ. Очень удобной, в частности, оказывается команда подсчета единиц в машинном слове.

Элементарный шаг алгоритма выглядит следующим образом. Пусть сравниваются слова $a_1(i)$ и $a_1(j)$. Вычисляем расстояние $d_1(i,j)$ (см. ниже). Если $d_1(i,j) \le k$, пара слов $a_1(i)$ и $a_1(j)$ – искомая. Сдвигаемся на один символ и повторяем процедуру применительно к словам $a_1(i+1)$ и $a_1(j+1)$ (можно просто скорректировать расстояние). Если же $d_1(i,j) = k + r$, где $r > 0$, совершаем скачок на r символов и переходим к сравнению слов $a_1(i+r)$ и $a_1(j+r)$, т.е. вычисляем $d_1(i+r, j+r)$.

Таким образом, продвижение по каждой диагонали идет скачками переменной длины r. Среднее значение r для нашего класса последовательностей $\bar{r}(l,k) = (1-P_2)\cdot l - k$, где $P_2 = \sum_{\alpha \in S} p^2(\alpha)$, откуда средняя трудоемкость

$$T = O(N^2/r(l,k)). \qquad (6)$$

ПРИМЕР 2. Пусть $n = 4$, $p(\alpha) = \frac{1}{4}$ для всех $\alpha \in S$, $l = 16, k = 4$. Тогда $\bar{r}(l,k) = 8$, т.е. в среднем оценивается лишь восьмая часть элементов в матрице расстояний.

Мультипликативная константа, подразумеваемая в (6), определяется временем вычисления хэммингова расстояния. Возможные схемы его вычисления в терминах команд БЭСМ-6 приведены на рис.3. Ориентация на БЭСМ-6 обусловлена большой длиной разрядной сетки в данной машине (48 разрядов) и удобным для наших целей набором команд.

Корректность всех четырех схем просматривается достаточно легко. В схемах "а"-"в" используются по два разряда для представления каждого символа, причем в первых двух схемах код символа записывается по горизонтали, а в третьей схеме - по вертикали (это соответствует тому, что старшие разряды кодов собираются в одно слово, а младшие - в другое). Базовой (и наиболее прозрачной) является схема "в". С ее помощью можно вычислять расстояния для слов с $1 \leq 48$. При $48 < 1 \leq 96$ схема применяется двукратно и т.д. Реальное время работы алгоритма, основанного на этой схеме, приведено в таблице.

Таблица

Время работы алгоритма отыскания всех (20,7)-повторов в первичных структурах различной длины (ЭВМ БЭСМ-6)

Длина последовательности	1027	2191	4010	8322
Время поиска (сек)	3,23	13,47	36,06	139,3

Схема "а" - аналог схемы "в" для "горизонтальной кодировки". Старшие и младшие разряды кодов собираются с помощью операции разборки. В схеме "б" операция разборки (довольно редкая в практике конструирования ЭВМ) заменена более типовыми операциями.

В схеме "г" использована трехразрядная кодировка со следующим свойством: хэммингово расстояние между любой парой различных кодов равно двум. Благодаря этому удвоенное хэммингово расстояние между сравниваемыми словами получается за две операции. В принципе, его сразу можно сравнивать с удвоенным пороговым значением.

Отметим, что во всех четырех схемах присутствует операция (команда) подсчета числа единиц в коде, которая является наиболее трудоемкой, к тому же имеется не во всех ЭВМ. В принципе, она может быть заменена простым табличным алгоритмом. Пусть, к примеру, код содержит m байтов. Если отсутствует команда определения чис -

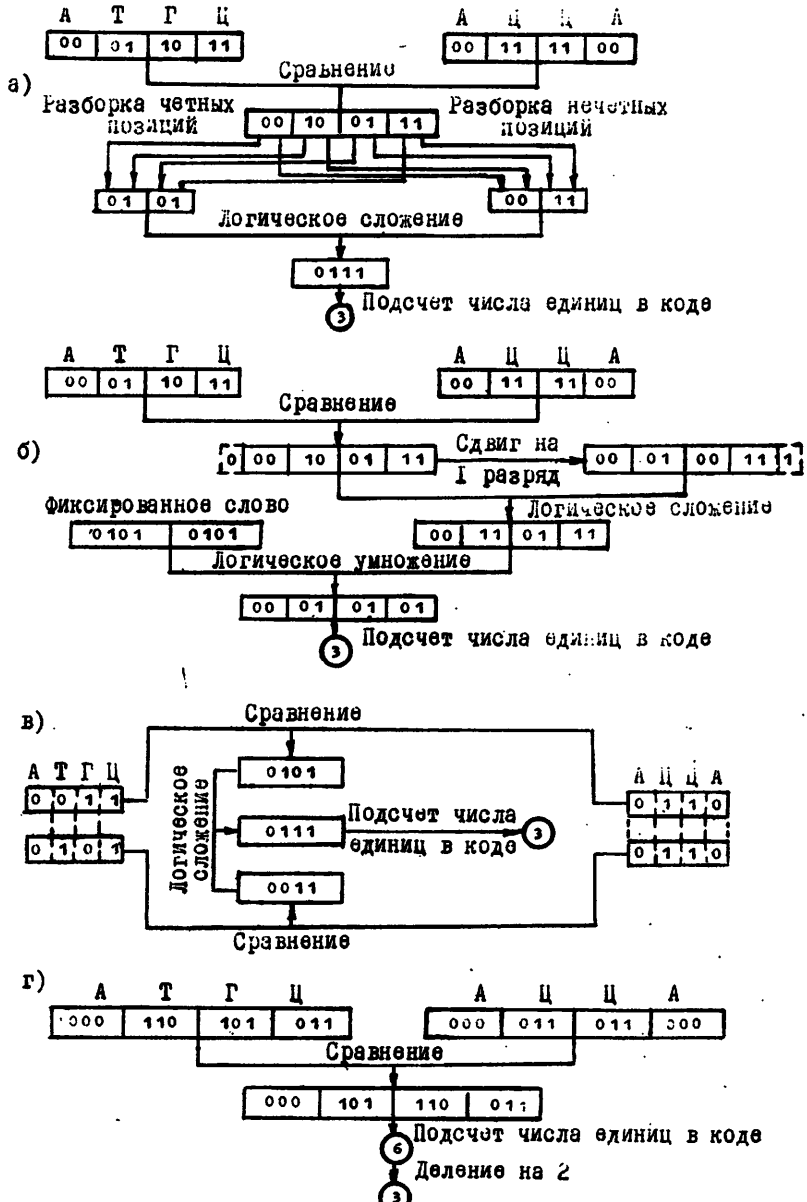

Рис.3. Четыре схемы вычисления хэммингова расстояния между словами в четырехбуквенном алфавите (на примере слов АТГЦ и АЦЦА, $d = 3$).

ла единиц в байте, заводим массив из 256 чисел, каждое из которых есть число единиц в восьмиразрядной кодовой комбинации, определяющей порядковый номер данного числа в массиве. Просматриваем m - байтовый код байт за байтом, интерпретируем каждый байт как целое число, определяющее вход в таблицу, и суммируем значения соответствующих элементов таблицы. После m обращений к таблице получаем результат, осуществив (m-1) целочисленных сложений.

Заключение

Рассмотрена задача отыскания в длинных текстах всех несовершенных повторов заданной длины, т.е. участков текста, близких в смысле хэммингова расстояния. Задача актуальна в генетических приложениях и в приложениях, связанных с автоматическим обнаружением и исправлением ошибок в тексте.

Показана связь задач отыскания несовершенных и совершенных (идеальных) повторов. Использованный при этом метод перекодировок, в принципе, может рассматриваться как самый общий метод сведения первой задачи ко второй, более простой.

Предложены два алгоритма решения задачи с квадратичной трудоемкостью в наихудшем случае, когда допускается сильное различие между словами, образующими несовершенный повтор. Один алгоритм имеет универсальный характер и не накладывает никаких ограничений на длины отыскиваемых повторов, мощность алфавита и тип используемой ЭВМ. Второй алгоритм (более быстрый) является машинно-ориентированным и существенно использует фактор малости алфавита, что характерно для генетических приложений. Первый алгоритм реализован на ЭВМ единой серии в рамках ППП СИМВОЛ и апробирован на генетических и музыкальных текстах. Второй алгоритм реализован на БЭСМ-6 и апробирован на генетических текстах.

Литература

1. WAGNER R.A., FISCHER M.I. The string-to-string correction problem.- JACM,1974,v.21, N 1,p.168-173.

2. WHITE C.T., HARDIES S.C., HUTCHISON C.A., EDGELL M.H. The diagonal-traverse homology search algorithm for locating similarities between two sequences. - Nucl.Acids Res., 1984, v.12, N 1, p.751-766.

3. KORN L.J., QUEEN C.L., WEGMAN M.N. Computer analysis of nucleic acid regulatory sequences.- Proc.Natl.Acad.Sci. USA,1977, v.74,N 10,p.4401-4405.

4. ЖАРКИХ А.А., СОЛОВЬЕВ В.В., КОЛЧАНОВ Н.А.Контекстный анализ полинуклеотидных последовательностей. Новосибирск, 1983. - 44 с. (Препринт/ИЦиГ СО АН СССР).

5. КОЛЧАНОВ Н.А., СОЛОВЬЕВ В.В., ЖАРКИХ А.А. Высокая насыщенность прямыми повторами в генах РНК-полимераз по данным контекстного анализа. -ДАН, 1983, т.273, №3, с.741-744.

6. ГУСЕВ В.Д., КОСАРЕВ Ю.Г., ТИТКОВА Т.Н. О задаче поиска повторяющихся отрезков текста. -В кн.: Вычислительные системы, вып.62. Ассоциативное кодирование. Новосибирск, 1975, с.49-71.

7. МИХАЙЛОВ В.Г. Центральная предельная теорема для числа неполных длинных повторений. -Теория вероятностей и ее применения, 1975, т.XX, вып.4, с.880-884.

8. ГУСЕВ В.Д. Механизмы обнаружения структурных закономерностей в символьных последовательностях. -В кн.: Проблемы обработки информации (Вычислительные системы, вып.100). Новосибирск, 1983, с.47-66.

Поступила в ред.-изд.отд.
11 ноября 1985 года