Н.К. Верещагин, В.А. Любецкий

Алгоритм определения вторичной структуры РНК

An algorithm that searches for a secondary structures in a given RNA sequences is suggested. The complexity of the algorithm is close to quadratic The algorithm was tested on simulated and natural sequences

Приводится алгоритм решения одной из фундаментальных задач алгоритмической (или, как иногда говорят, вычислительной) генетики – алгоритм определения вторичной структуры РНК Сообщение состоит из трех пунктов: математической постановки задачи, описания алгоритма (включая оценку времени его работы), нескольких слов, относящихся к биологической мотивировке. Алгоритм был компьютерно реализован, был проведен счет на модельных и реальных исходных данных (который показал его удовлетворительность); результаты счета, как и их интерпретации, представлены в журнал «Молекулярная биология» Строго математическое исследование этой задачи важно, но весьма трудно

1. Постановка задачи

Имеется последовательность нуклеотидов длины ДЛИНА (где ДЛИНА порядка 1000). Под отрезком [a,b], где a <= b, понимается множество натуральных чисел {a,a+1,... ,b} Два отрезка [a,b] и [c,d] комплементарны, если в исходной последовательности a-ый нуклеотид комплементарен d-ому, (a+1)-ый нуклеотид комплементарен (d-1)-ому и т.д. (в частности длины отрезков одинаковы) *Шпилькой* назовём последовательность комплементарных пар отрезков [a1,b1], [c1,d1], ... , [an,bn], [cn,dn], все отрезки которой попарно не перекрываются, расположены на числовой оси в порядке возрастания следующим образом: [a1,b1], [a2,b2], ... , [an,bn], [cn,dn], ... , [c1,d1], длина каждого её отрезка не меньше МИНПОДРЯД (где МИНПОДРЯД – целое число порядка 3), все разности a2-b1, a3-b2, ... , d3-c2, d2-c1 меньше МАКСВЫПЯЧИВАНИЕ (где МАКСВЫПЯЧИВАНИЕ – целое

число порядка 20), а разность (cn - bn) не больше МАКСПЕТЛЯ и не меньше МИНПЕТЛЯ (где МАКСПЕТЛЯ – целое число порядка 20, МИНПЕТЛЯ – целое число порядка 3). Кроме того требуется, чтобы *длина шпильки* (определяемая как d1-a1+1) не превосходила ДЛИНА_ШПИЛЬКИ (где ДЛИНА_ШПИЛЬКИ – целое число порядка 40).

Мы говорим, что отрезок [a1,b1] склеен с отрезком [c1,d1], (a1 с d1, a1+1 с d1-1 и т.д.), отрезок [a2,b2] – с отрезком [c2,d2] и т.д. Отрезок [a1,bn] называется *левым основанием*, отрезок [cn, d1] – *правым основанием*. Число a1 называется *началом шпильки*, d1 – *концом шпильки*, bn – *началом петли*, cn – *концом петли*. Также [a1,b1] называется *внешним краем*, а [cn,bn] – *внутренним краем* шпильки.

Перекрытием называется пара шпилек, у которых, во-первых, правое основание первой шпильки имеет не менее ДЛИНА_ПЕРЕСЕЧЕНИЯ общих элементов с левым основанием второй шпильки (где ДЛИНА_ПЕРЕСЕЧЕНИЯ – целое число порядка 5), во-вторых, конец первой шпильки находится левее конца петли второй шпильки и, в-третьих, начало второй шпильки находится правее начала петли первой шпильки. Левое и правое основания первой шпильки перекрытия будут обозначаться соответственно [A',B'] и [C',D'], а второй шпильки – [A,B] и [C,D]. Таким образом, D' < C и A > B'. Пересечение отрезков [C',D'] и [A,B] называется *базой перекрытия* (как говорилось его длина не менее ДЛИНА_ПЕРЕСЕЧЕНИЯ).

Мощностью шпильки называется количество склеенных пар нуклеотидов. *Мощностью перекрытия* называется сумма мощностей его шпилек. Два перекрытия назовем *подобными*, если у них одинаковые мощности, а также одинаковые левые и правые основания обеих шпилек.

Итак, постановка задачи. Приблизительно говоря, алгоритм получив на вход исходную последовательность и число КОЛИЧЕСТВО_ПЕРЕКРЫТИЙ (где КОЛИЧЕСТВО_ПЕРЕКРЫТИЙ – целое число порядка 5) должен напечатать КОЛИЧЕСТВО_ПЕРЕКРЫТИЙ перекрытий наибольшей мощности, при этом из подобных перекрытий печатается не более одного. Точнее говоря, алгоритм будет находить следующие перекрытия. Рассмотрим на множестве перекрытий линейный порядок, чтобы сравнить два перекрытия, сначала сравниваются их мощности, при равных мощностях сравниваются A, затем D', затем C,D,B,B',A',C' в указанном порядке. При этом *чем больше мощность, больше D' (C' A' C') и меньше A D,B,B'*, тем перекрытие «*больше*». Если

отождествлять подобные перекрытия, то указанное отношение действительно является линейным порядком. Для каждого X (1<=X<=ДЛИНА − ДЛИНА_ПЕРЕСЕЧЕНИЯ+1) определим Y=Y(X) = X+ДЛИНА_ПЕРЕСЕЧЕНИЯ-1. Определим K(X) как *наибольшее перекрытие*, база которого включает отрезок [X,Y]. Рассмотрим последовательность перекрытий П1, П2, ..., определяемую по индукции так: Пусть П1, П2, ..., Пi уже определены. Пусть M наибольшая из мощностей перекрытий, неподобных ни одному из перекрытий П1, П2, ..., Пi. Пусть X наименьшее такое, что K(X) имеет мощность M и не подобно ни одному из перекрытий П1, П2, ..., Пi. Тогда П(i+1)=K(X). Тогда алгоритм находит КОЛИЧЕСТВО_ПЕРЕКРЫТИЙ первых элементов этой последовательности, если ее длина не менее КОЛИЧЕСТВО_ПЕРЕКРЫТИЙ, а иначе - находит всю эту последовательность.

Верхняя оценка времени работы алгоритма: первый этап O(ДЛИНА * ДЛИНА_ШПИЛЬКИ^3 * МАКСВЫПЯЧИВАНИЕ) арифметических операций, второй этап O(КОЛИЧЕСТВО ПЕРЕКРЫТИЙ * ДЛИНА_ШПИЛЬКИ * МАКСВЫПЯЧИВАНИЕ * (ДЛИНА_ШПИЛЬКИ + МАКСВЫПЯЧИВАНИЕ)) арифметических операций.

2. Алгоритм решения задачи

Этап 1. На этом этапе мы для всех X находим две четверки (m,A,C,D) и (m',A',B',D') такие, что первая шпилька перекрытия K(X) имеет мощность m' основания [A',B'] и [C',D'], а вторая шпилька перекрытия K(X) – соответственно m и [A,B] и [C,D] (для некоторых B,C'). К сожалению, значения B,C' мы не можем вычислять быстро. Мощность K(X) храним в массиве «мощность» (мощность[X]=мощность K(X)). Любое перекрытие, база которого включает отрезок [X,Y], целиком лежит на отрезке [НАЧАЛО,КОНЕЦ], где НАЧАЛО = max{1,Y-ДЛИНА ШПИЛЬКИ}, КОНЕЦ= min{ДЛИНА,X+ДЛИНА ШПИЛЬКИ}.

Для нахождения значения мощность[X] мы сначала рассматриваем шпильки, начало и конец которых принадлежат отрезку [НАЧАЛО,КОНЕЦ], начало петли больше или равно Y (B>=Y). Назовем такие шпильки *допустимыми*. Определяем качество такой шпильки как тройку (m,C,D), где m - ее мощность, C - конец петли, а D - конец. Качество тем лучше, чем больше m и C и меньше D. Точнее, на множестве таких троек мы рассматри-

ваем отношение частичного порядка (m,C) <= (m',C'), если m <= m' и C <= C', (m,C,D) <= (m',C',D'), если (m,C) < (m',C') или (m,C) = (m',C') и D<=D' То есть на парах (m,C) порядок покомпонентный, а на тройках лексикографический, если тройку (m,C,D) понимать как пару ((m,C),D). При каждом A из отрезка [НАЧАЛО,КОНЕЦ] мы находим множество, называемое максшпилька2[X,A], состоящее из максимальных возможных качеств допустимых шпилек с началом A Все тройки из множества максшпилька2[X] несравнимы, поэтому при каждом m в этом множестве не более одной тройки вида (m,C,D) Множество максшпилька2[X] хранится в виде отсортированного (по возрастанию m) массива троек

Затем, симметрично, рассматриваем шпильки, начало и конец которых принадлежат отрезку [НАЧАЛО,КОНЕЦ], конец петли меньше или равен X (C'<=X) Определяем качество такой шпильки как пару (m,B',A'), где m - ее мощность, а B' - начало петли Качество тем лучше, чем больше m,A' и меньше B' Точнее, на множестве таких троек мы рассматриваем отношение частичного порядка (m,B') <= (m',B''), если m <= m' и B' >= B'', (m,B',A') <= (m',B'',A''), если (m,B') < (m',B'') или (m,B') = (m',B'') и A'<=A'' При каждом D' из отрезка [НАЧАЛО,КОНЕЦ] мы находим множество максшпилька1[X,D'] максимальных возможных качеств таких шпилек

Теперь мы можем найти мощность[X], сравнивая шпильки обоих типов А именно, мощность[X] равно наибольшей сумме m+m', для которой *существует отрезок* [A,D'], содержащий отрезок [X,Y], со следующим свойством (m',B',A') – тройка из списка максшпилька1[D'] с наибольшим m', для которой B' < A, а (m,C,D) – тройка из списка максшпилька2[A] с наибольшим m, для которой C > D' Находим мощность[X] перебором отрезков [A,D'] Затем еще раз перебираем [A,D'], чтобы найти (m',A',B',D') и (m,A,C,D)

Как при фиксированном X заполнить массив максшпилька2? Мы перебираем K из отрезка [Y,КОНЕЦ] в порядке возрастания При каждом K рассматриваем шпильки, начало и конец которых принадлежат отрезку [НАЧАЛО,K], а начало петли больше или равно Y Будем называть такие шпильки K-*допустимыми* При K=КОНЕЦ понятия K-допустимости и допустимости совпадают При каждом A из отрезка [НАЧАЛО,K] мы находим множество максимальных возможных качеств K-допустимых шпилек с началом A и полагаем максшпилька2[A,K,X]

равным этому множеству. При К=КОНЕЦ множество максшпилька2[А,К,Х] будет хранить искомые тройки.

Для того, чтобы прошел индуктивный переход К --> К+1, мы храним дополнительную информацию. А именно, мы отдельно рассматриваем К-допустимые шпильки, у которых конец принадлежит отрезку [К-МАКСВЫПЯЧИВАНИЕ,К]. Назовем такие шпильки *К-продолжаемыми*. (К-Продолжаемые шпильки можно продолжать, склеивая символы с номерами, большими К.) *Качество К-продолжаемой шпильки* определяется как пара (m,C), где m – ее мощность, а С – конец петли. Качества сравниваются покомпонентно: (m,C) <= (m',C'), если m <= m' и C <= C'. Информацию о К-продолжаемых шпильках мы храним в виде *очереди продолж*[А]. А именно, продолж[А] есть очередь длины МАКСВЫПЯЧИВАНИЕ+1, n-ый элемент которой есть множество максимальных качеств К-продолжаемых шпилек с началом А и концом К-МАКСВЫПЯЧИВАНИЕ+n-1. (Элементы в очередь добавляются с конца, а удаляются с начала, нумеруются элементы очереди с начала: 1,2, ...)

Более того, мы рассматриваем и так называемые *внутренние предшпильки* (или, кратко, предшпильки). Грубо говоря, внутренняя предшпилька - это то, что останется от шпильки, если ее разделить на две части и удалить внешнюю часть. Формально (внутренняя) предшпилька отличается от шпильки тем, что в предшпильке длины внешних отрезков ([a1,b1] и [c1,d1]) могут быть меньше МИНПОДРЯД, длина внешних отрезков называется *кратностью предшпильки*. *Предшпилька К-допустима*, если ее конец равен К, а начало петли больше или равно Y. К-допустимые предшпильки нужны, поскольку их можно продолжать до шпилек, склеивая символы с номерами, большими К. Качество предшпилек определяется так же, как и качество К-продолжаемых шпилек. В массиве «предшпилька» мы храним информацию о них. А именно *предшпилька*[А] есть антиочередь длины МИНПОДРЯД-1, r-ый элемент которой есть множество максимальных качеств предшпилек кратности r (где r=1, ..,МИНПОДРЯД-1). (В антиочередь элементы добавляются с начала, а удаляются с конца, нумеруются элементы антиочереди по-прежнему с начала: 1,2, ...)

Множества из (анти)очередей предшпилька[А], продолж[А] хранятся также в виде упорядоченных по возрастанию первого компонента массивов чисел.

Как же совершить переход от К к К+1? Мы имеем правильно заполненные продолж[А], предшпилька[А], максш-

пилька2[A] для данного K Ясно, что если K+1 не комплементарно A, то K+1-допустимых предшпилек нет вовсе, множество K+1-допустимых шпилек совпадает с множеством K-допустимых шпилек, а множество K+1-продолжаемых шпилек равно множеству K-продолжаемых шпилек минус шпильки с концом K-МАКСВЫПЯЧИВАНИЕ То есть, в этом случае для K+1 антиочередь предшпилька[A] состоит из пустых множеств, множество макешпилька2[A] не изменяется, а из очереди продолж[A] удаляется первое множество, и в ее конец вставляется пустое множество

Более сложным образом изменяются эти массивы, если K+1комплементарно A В этом случае к K+1-допустимым шпилькам с началом A добавляются шпильки, полученные склеиванием A с K+1 у K-допустимых предшпилек кратности МИНПОДРЯД-1 с началом A+1, а также и шпильки, полученные склеиванием A с K+1 у шпилек с началом A+1 и концом K Эти шпильки являются K+1-продолжаемыми, поэтому они добавляются к K+1- продолжаемым шпилькам Из множества K-продолжаемых шпилек удаляются все шпильки с концом K-МАКСВЫПЯЧИВАНИЕ и оставшиеся шпильки вместе с вышеуказанными K+1-продолжаемыми шпильками образуют все множество K+1-продолжаемых шпилек Множество K+1-допустимых предшпилек с началом A формируется так берутся все K-допустимые предшпильки с началом A+1 кратности меньше МИНПОДРЯД-1, и в каждой из них склеиваются A и (K+1) Кроме этого, могут добавиться предшпильки кратности 1

Они берутся, во-первых, если у некоторой K-продолжаемой шпильки с началом, принадлежащим отрезку [A+1,A+МАКСВЫПЯЧИВАНИЕ+1] склеить еще A с K+1, и, во-вторых, если просто склеить A с K+1, получив предшпильку мощности 1 Последнее возможно, если K-A находится в пределах от МИНПЕТЛЯ до МАКСПЕТЛЯ

Итак, для получения новых значений макешпилька2[A], продолж[A], предшпилька[A] делаем следующее Удаляем из очереди продолж[A] первое множество Сливаем последнее множество из очереди продолж[A+1] с последним множеством из антиочереди предшпилька[A+1] (именно в них хранится информация о K-допустимых шпильках с началом A+1 и концом K и о предшпильках кратности МИНПОДРЯД-1 с началом A+1) Об операции слияния двух множеств будет сказано позже Увеличиваем первый компонент всех пар в полученном множестве на 1 (получим множество всех максимальных качеств продолжаемых шпи-

лек с началом A и концом K+1) и ставим полученное множество в конец очереди продолж[A]. Теперь добавляем ко всем парам этого множества K+1 в качестве третьего компонента и результат вливаем в множество максшпилька2[A].

Удаляем из антиочереди предшпилька[A+1] последнее множество и увеличиваем первые компоненты пар всех множеств на 1. Это будет концом антиочереди предшпилька[A]. Осталось найти множество, которое надо поставить в начало этой антиочереди, то есть множество максимальных качеств предшпилек кратности 1 с началом A и концом K+1. Для этого сливаем вместе все множества, хранимые во всех очередях продолж[A+1], продолж[A+МАКСВЫПЯЧИВАНИЕ+1]. Получим множество максимальных качеств всех продолжаемых шпилек с началом на отрезке [A+1, A+МАКСВЫПЯЧИВАНИЕ+1]. Увеличим первый компонент всех пар этого множества на 1 и поставим в начало антиочереди предшпилька[A]. Наконец, если МИНПЕТЛЯ <= K-A <= МАКСПЕТЛЯ, то добавляем в это множество еще пару (1,K+1). Для того, чтобы новые значения (анти)очередей не испортили старые, *нужно перебирать A в порядке возрастания*.

Слияние всех множеств, хранимых во всех очередях продолж[A+1], ..., продолж[A+МАКСВЫПЯЧИВАНИЕ+1] значительно сложнее всех остальных операций, поскольку сливается вместе (МАКСВЫПЯЧИВАНИЕ+1)^2 множеств. Назовем множество, полученное в результате этого, через *соседи*[A]. Его можно вычислять индукцией по A более простым способом. Для этого перебираем A в порядке убывания и при каждом A вычисляем антиочередь queue длины МАКСВЫПЯЧИВАНИЕ+1, n-ый элемент которой есть слияние всех множеств очереди продолж[A+n]. Множество соседи[A] получается в результате слияния всех элементов антиочереди queue (что требует слияния МАКСВЫПЯЧИВАНИЕ+1 множеств). При уменьшении A на единицу из антиочереди queue исключается последнее множество, а в начало ставится слияние всех множеств из очереди продолж[A] (что опять требует слияния МАКСВЫПЯЧИВАНИЕ+1 множеств). Таким образом для получения множества соседи[A] мы делаем два слияния МАКСВЫПЯЧИВАНИЕ+1 множеств вместо одного слияния (МАКСВЫПЯЧИВАНИЕ+1)^2 множеств.

Массив максшпилька1 заполняется симметричным образом (K перебираем в порядке убывания).

Теперь обсудим операцию слияния множеств пар вида (m,C). Напомним, что сливаемые множества хранят попарно несравнимые пары, то есть не более одной пары с данным пер-

вым компонентом, и хранятся в виде упорядоченного массива пар. Операция слияния состоит просто в соединении двух упорядоченных массивов в новый упорядоченный массив (как в алгоритме сортировки слиянием). При этом, если окажутся рядом две пары с одним и тем же первым компонентом, то меньшая пара удаляется. Операция слияния нескольких множеств производится естественным образом: первое сливается со вторым, результат сливается с третьим и т.д. Следует заранее исключить из сливаемых множеств пустые, поскольку в типичном случае их должно быть от половины до трех четвертей (шпилька с началом A и концом K существует только если A комплементарно K).

Множества троек вида (m,C,D) сливаются точно так же. Влить множество M1 в множество M2 означает: слить M1 и M2 и результат поместить в M2.

На хранение одного множества пар вида (m,C) следует зарезервировать 2 × ДЛИНА_ШПИЛЬКИ байт, а на хранение одного множества пар вида (m,C,D) – 3 × ДЛИНА_ШПИЛЬКИ байт.

Этап 2. Сначала объясним, как, имея M (A,C,D) и Y, найти наименьшее такое B>=Y, что есть шпилька мощности M с основаниями [A B], [C,D] и саму шпильку, при условии, что M – наибольшая возможная мощность такой шпильки. Применив этот алгоритм и вместе с симметричным ему, мы *можем для каждого* X *найти* K(X) (но делать этого пока не надо).

Алгоритм похож на алгоритм первого этапа. Дадим определение *внешней предшпильки*. Внешняя предшпилька отличается от шпильки тем, что мы разрешаем внутренним (а не внешним отрезкам, как у внутренних предшпилек) иметь длину меньше МИНПОДРЯД (эту длину называем по-прежнему кратностью), а расстояние между началом и концом петли не ограничено сверху. Мы рассматриваем только предшпильки мощности не более M (более мощные предшпильки неперспективны).

Мы перебираем B в порядке возрастания, начиная с B=A. При фиксированном B перебираем c из отрезка [C,D] в порядке убывания и рассматриваем внешние предшпильки с основаниями [A,B] и [c D] (назовем их B,c-допустимыми). Для каждого c из отрезка [C D] мы вычисляем продолж[c,B] = максимальная мощность B,c-допустимой предшпильки кратности не менее МИНПОДРЯД (назовем их полными), и неполн[c,B] = антиочередь длины МИНПОДРЯД-1, r-ый элемент которой есть максимальная мощность B,c-допустимой предшпильки кратности r (допустимой предшпильки кратности менее МИНПОДРЯД будем называть

неполными) Если указанных предшпилек нет, то максимальная мощность считается равной минус бесконечность

Как осуществлять переход B->B+1? Пусть c-1 комплементарно B+1 Какие бывают B+1,c-1-допустимые предшпильки? Полные B+1 c-1-допустимые предшпильки получаются склеиванием c-1 и B+1 из полных B c-допустимых предшпилек и из неполных B c-допустимых предшпилек кратности МИНПОДРЯД-1 Неполные B+1,c-1-допустимые предшпильки кратности r получаются из неполных B,c-допустимых предшпилек кратности r-1 склеиванием c-1 и B+1 Кроме того, любую b,c'-допустимую полную предшпильку, где b лежит на отрезке [B-МАКСВЫПЯЧИВАНИЕ,B] и c' лежит на на отрезке [c, c+МАКСВЫПЯЧИВАНИЕ], можно продолжить, склеив c-1 и B+1

Иными словами, если c-1 комплементарно B+1, то полн[c-1,B+1] есть 1+ max{полн[c,B], последний элемент антиочереди неполн[c,B]} Антиочередь неполн[c-1,B+1] получается из антиочереди неполн[c,B] удалением последнего элемента, увеличением всех элементов на 1 и добавлением в начало элемента 1+ neighbor(c-1), где neighbor(c-1) = наибольшая мощность полной b,c'-допустимой полной предшпильки, где b лежит на отрезке [B-МАКСВЫПЯЧИВАНИЕ,B] и c' лежит на отрезке [c,c+МАКСВЫПЯЧИВАНИЕ]

В начальный момент (при B=A) мы полагаем неполн[C,A] = (1 минус бесконечность, . . минус бесконечность), поскольку имеется единственная A,D-допустимая предшпилька, она имеет мощность 1 и кратность 1, и полагаем неполн[c,A]=(минус бесконечность, . . минус бесконечность) для всех остальных c

Число neighbor(c) вычисляется так Пусть queue(c) – антиочередь, n-ый элемент которой равен максимуму мощностей b,c+n-допустимых полных предшпилек по всем b на отрезке [B-МАКСВЫПЯЧИВАНИЕ, B] Тогда neighbor(c) есть максимальный из элементов антиочереди queue(c) А queue(c) легко найти по индукции при переходе от c к c-1 конец антиочереди удаляется, а в начало ставится наибольший из элементов полн[c, B-МАКСВЫПЯЧИВАНИЕ], . . полн[c,B-1], полн[c,B]

Перебор B останавливаем, когда полн[C,B]=M и при этом Y<=B МИНПЕТЛЯ<=C-B+1<=МАКСПЕТЛЯ (поскольку в этом случае имеется шпилька мощности M с основаниями [A,B], [C,D], где Y<=B) Такое B обязательно найдется Действительно в качестве B годится любое B>=Y, для которого есть шпилька мощности M с основаниями [A,B], [C,D]

Найдя B ищем самy шпилькy просматривая созданные нами массивы и наращивая ее «изнутри» То есть мы будем хранить некоторую внутреннюю предшпильку - часть будущей шпильки Как хранить – неважно например ее можно задавать как список склеенных пар нуклеотидов с названием «шпилька» Кроме списка шпилька мы отдельно храним четверку m'.b c s где m' - мощность хранимой внутренней предшпильки s - ее кратность a b c – ее начало и конец При этом мы поддерживаем следующий инвариант существует b c-допустимая внешняя предшпилька кратности не менее МИНПОДРЯД-s+1 и мощности M-m'+1 (ее соединение с хранимой внешней предшпилькой даст искомую шпильку)

Начинаем с того что кладем в список шпилька парy (B C) и полагаем b=B c=C m'=1 s=1 Затем пока m'<M выполняем следующее Если полн[c+1 b-1]=M-m' или в антиочереди неполн[c+1 b-1] для некоторого r>=МИНПОДРЯД-s r-ый элемент равен M-m' то yменьшаем b и yвеличиваем c на единицy yвеличивая s на единицy (то есть склеиваем y хранимой предшпильки соседние с началом и концом нуклеотиды) Иначе (это может быть только если s>=МИНПОДРЯД) для некоторого c' из отрезка [c+1 c+1+МАКСВЫПЯЧИВАНИЕ] и некоторого b' из отрезка [b-1 b-1-МАКСВЫПЯЧИВАНИЕ] выполнено полн[c' b']=M-m'

Полагаем b=b' c=c' s=1 Наконец заносим парy (b c) в список шпилька и полагаем m'=m'+1

Итак после первого этапа для всех X известна максимальная мощность (мощность[X]) перекрытия база которого включает отрезок [X Y] (где Y= X+ДЛИНА_ПЕРЕСЕЧЕНИЯ-1) Кроме того по любомy X можно найти K(X)

Находим M - наибольшее значение в массиве мощность[X] Находим первое в порядке возрастания X1 для которого мощность[X1]=M Находим K(X1) Ясно что K(X1) есть первый элемент искомой последовательности Пусть [R1 S1] - его база Ясно что если отрезок [X Y] включен в [R1 S1] то K(X)=K(X1) Поэтомy находим X2 – наименьшее X для которого мощность[X]=M и Y(X) > S1 Находим K(X2) Ясно что K(X2)=П2 Пусть [R2 S2] - его база

Так действyем до тех пор пока очередное Xi существует После этого находим M' - наибольшyю мощность[X] для таких X что мощность[X] < M Дальше делаем то же самое заменив M на M' В этом дyхе действyем до тех пор пока не наберется КОЛИЧЕСТВО_ПЕРЕКРЫТИИ перекрытии или очередное M" ' не станет равным нyлю **Конец описания алгоритма.**

Приведем один **пример** исходной последовательности (для аттенюатора гистидинового оперона кишечной палочки EcjhisL)

tagctaattgtacgcatgtcaatctcctcttttgtacagttcattgtacaatgatgagcgttaatta
actatttattaattagtttgtagatcaaggtattgtcagtgagacgaaaatccaggcttcgctatttttggt
gccatcagctaagaggacagtcctcttagccccctcctttcccgctcattcattaaacaaatccattg
ccataaaatatataaaaaagccctgctttctaacgtgaaagtggtt

taggttaaaaagacatcagttgaataaacatteacagagacttttatgacacgcgttcaatttaa
acaccaccatcatcaccatcatcctgactagtctttcaggcgatgtgtgctggaagacatteagatctt
ccagtggtgcatgaacgcatgagaaagccccgggaagatcaccttccgggggctttttattgcgcg
gttgataacggttcagacaggtttaaagaggaataacaaaatgacagacaacactcgtttacgcata
gctatgcagaaatccggccgtttaagtgatgactcacgcgaattgctggcgcgctgtggcattaaaat
taatcttcacaccagcgcctgatcgcgatggcagaaaacatgccgattgatattctgcgcgtgcgtg
acgacgacattcccggtctggtaatggatggcgtggtagaccttgggattatcggcgaaaacgtgct
ggaagaagagctgcttaaccgccgcgcccagggtgaagatccacgctactttaccctgcgtcgtct
ggatttcggcggctgtcgtctttcgctggcaacgccggttgatgaagcctgggacggtc

3 Биологическая мотивировка

Речь идет о задаче поиска участков генома потенциально годных для образования альтернативных вторичных структур в РНК

Регуляция многих бактериальных генов осуществляется на уровне трансляции или взаимодействия процессов трансляции и транскрипции При этом основным регуляторным сигналом является образование вторичной структуры РНК Часто (хотя и не всегда) в рассматриваемой области могут образоваться две альтернативные структуры РНК что и служит регуляторным переключателем Примерами такой регуляции являются аттенюаторы опероно синтеза аминокислот [1] и регуляторные структуры некоторых оперонов рибосомальных белков кишечной палочки [2]

Литература

1 *Витрещак Н Гельфанд М С* Сравнительный подход к анализу регуляции в полных геномах аттенюаторы ароматических аминокислотных оперонов гамма-протеобактерии Молекулярная биология 34(4) в печати

2 *Витрещак Н Банса Н К Гитов И И Гельфанд М С* (1999) Компьютерный анализ регуляторных сигналов в полных бактериальных геномах Инициация трансляции оперонов рибосомальных белков Биофизика 44 601-610