**БИОАЛГОРИТМИКА**

**14.08 Занятие 5**

1. **Повторение: поиск вхождений слова в строку**
2. **Биологическая мотивация: поиск гомологов белков и НК**

Реальная задача: определить степень сходства двух данных геномов или белков. Или, шире, найти в большой базе данных (genbank) последовательности, наиболее сходные (в некотором смысле) с данной.

На практике для этой и смежных задач используются программы семейства BLAST. Используются сложные эвристические алгоритмы. Мы рассмотрим основные идеи на примере упрощенных задач.

1. **Поиск точных сходств**
   1. Постановка задачи

Дано две длинные строки S и T (геномы) длиной |S|=m и |T|=n и число k << m,n.

Требуется найти все совпадающие подстроки длины ≥ k.

Определение: *подстрокой* строки T = t1…tn называется строка T’ = t1+i…tm+i, где 0 ≤ i и m + i ≤ n. При i = 0 подстрока называется *префиксом* строки, при m + i = n – *суффиксом*.

* 1. Наивный алгоритм поиска

Идея: каждая общая подстрока длины ≥ k входит в строки S и T, начиная с позиций is и it соответственно, где 1 ≤ is ≤ m – k + 1 и 1 ≤ it ≤ n – k + 1. Переберем все возможные пары (is, it), всего их m∙n. Если для каждой пары стартовых позиций мы будем выполнять до k сравнений букв (проверка совпадений), то время работы алгоритма составит O(mnk). Но мы уже не настолько наивны. Ниже на примере показано, как искать общие подстроки, сравнивая каждую пару букв по одному разу, и, таким образом, затрачивая время O(mn).

Представим себе таблицу m×n, в которой на i-й строке и в j-м столбце стоит 1, если si = tj (буквы в исходных строках на этих позициях совпадают) и 0 иначе.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | G | A | T | A | G | T | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| C | 1 | 0 | 0 | 0 | 0 | 0 | 0 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| T | 0 | 0 | 0 | 1 | 0 | 0 | 1 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| A | 0 | 0 | 1 | 0 | 1 | 0 | 0 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| G | 0 | 1 | 0 | 0 | 0 | 1 | 0 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| A | 0 | 0 | 1 | 0 | 1 | 0 | 0 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| A | 0 | 0 | 1 | 0 | 1 | 0 | 0 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |

Проверим все пары букв на совпадение, «проходя» при этом «вдоль диагоналей». Количество совпадений, встреченных на некоторой диагонали *подряд*, заносим в переменную-счетчик. Достижение счетчиком значения k означает нахождение искомой подстроки.

Замечание 1. В данной задаче с биологическим контекстом, как правило, искомые общие участки могут идти как слева направо, так и справа налево (направления у строки нет). Это означает, что после одного прохода вдоль диагоналей нам надо пройти также вдоль всего семейства «перпендикулярных» им диагоналей. Это удваивает число операций на поиск, но асимптотика остается O(mn).

Замечание 2. Нужно ли учитывать при оценке эффективности алгоритма время на вывод ответов, или память, необходимую для их хранения? Мы договоримся, что не учитываем.

Замечание 3. Предложена идея оптимизации: при поиске несовпадения переходить не на следующую пару символов, а через k-1. Это дает время O(mn/k).

* 1. Оптимизация: таблица возможных подстрок

Покажем, как с помощью построения дополнительной структуры данных уменьшить время поиска сходств до O(n+m). Допустим, в нашей задаче k=5. Сколько всего существует различных строк длины 5? Понятно, что 45=1024. Если у строк S и T есть общая подстрока длины 5, то она принимает одно из этих 1024 значений. Идея: проверим, какие из этих 1024 значений принимают одновременно некоторая подстрока S и некоторая подстрока T.

1) Создадим массив длиной 1024. Каждому 5-буквенному слову можно однозначно сопоставить индекс массива. (Напишите программу, которая по строке выдает соответствующее число: 0 для AAAAA, 1 для AAAAC, …, 1023 для TTTTT)

2) Будем просматривать все подстроки длины 5 в строке S. Для каждой подстроки (их m-4 – почему?) занесем информацию об индексе вхождения этой подстроки в элемент массива, соответствующий этой подстроке. Но что, если некоторое значение встретилось в строке S несколько раз и хранить надо оба вхождения? Для этого договоримся, что элементами массива будут указатели на связные списки (изначально пустые) и новое вхождение некоторой подстроки будет означать добавление элемента в соответствующий список.

3) Пройдем по всем подстрокам длины 5 в строке T. Для каждой очередной подстроки обратимся к соответствующему элементу массива и узнаем все вхождения этой подстроки в строку S.

Анализ времени работы и затраченной памяти. Шаг 1 занимает O(4k) операций и O(4k) памяти. Шаг 2 занимает O(m) операций. Шаг 3 занимает O(n) операций. Итого: O(4k + m + n) операций и O(4k) памяти. Оценим при различных реалистичных m,n,k. Возьмем m и n ~106 (размер участка генома). Наивный алгоритм требует ~1012 времени и O(1) памяти.

При k = 5: операций ~103 + 106 +106 , т.е. ~ 106. Памяти ~103. Гораздо лучше наивного поиска.

При k = 10: операций ~106 + 106 +106 , т.е. ~ 106. Памяти ~106. Все еще лучше.

При k = 20: операций ~1012 + 106 +106 , т.е. ~ 1012. Памяти ~1012. Уже при таком k получаем паритет по порядку времени выполнения и существенный проигрыш по памяти. Как это обойти? Об этом подробно в следующем занятии.