Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКИ И СИСТЕМ УПРАВЛЕНИЯ

КАФЕДРА ТЕОРЕТИЧЕСКОЙ ИНФОРМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Пояснительная записка
к дипломному проекту на тему:

Исследование методов множественного выравнивания нуклеотидных последовательностей в соответствии с рамкой считывания и стоп-кодонами

Студент–дипломник _____ Батусов П. В.

Научный руководитель _____ Страшнов П. В.

Москва 2015

Аннотация

Содержание

ВВЕДЕНИЕ

Современная биоинформатика — это молодая, бурно развивающаяся наука, возникшая в 1976-1978 годах и окончательно оформившаяся в 1980 году со специальным выпуском журнала «Nucleic Acid Research» (NAR) [1]. По сути, это собрание различных математических моделей и методов в помощь биологам для решения биологических задач, таких как: предсказание пространственной структуры белков, расшифровка структуры ДНК, хранение, поиск и аннотация биологической информации.

Основу биоинформатики составляют сравнения. Одна из ключевых задач — поиск сходства последовательностей. Ее решение позволяет понять функциональное назначение частей геномов, оценить эволюционное расстояние между ними. Кроме этого, различия в генотипах могут объяснить различия в фенотипах.

Для того, чтобы определить, насколько две последовательности «похожи», используют алгоритмы выравнивания. Они основаны на размещении исходных последовательностей мономеров ДНК, РНК или белков друг под другом таким образом, чтобы легко увидеть их сходные участки [2]. Качество выравнивания оценивают, назначая штрафы за несовпадение букв и за наличие пробелов (когда приходится раздвигать одну последовательность для того, чтобы получить наибольшее число совпадающих позиций), например через расстояние Левенштейна — это минимальное число элементарных операций (вставка, удаление или замена символа в строке), чтобы превратить одну строку в другую [3]. При сравнении ищется такой вариант выравнивания, чтобы итоговый счет был максимален. В такой постановке задача называется поиском «глобального выравнивания». Необходимо отметить, что для полных геномов глобальное выравнивание не работает, так как при мутации помимо вставок, удалений и замен бывают нелинейные перестройки, которые могут менять порядок и ориентацию целых геномных блоков. Для решения, аналогично задаче поиска глобального выравнивания, формулируют задачу поиска «локального выравнивания»: для двух произвольных строк A и B найти две самые похожие подстроки и их выравнивание.

Алгоритмы множественного выравнивания, аналогично алгоритмам парного выравнивания, представляют собой инструмент для установления функциональных, структурных или эволюционных взаимосвязей между биологическими последовательностями. Несмотря на то, что задача множественного выравнивания была сформулирована более 20 лет назад [4], она до сих пор не теряет своей актуальности. Если говорить о множественном глобальном выравнивании то, по сравнению с парным выравниванием, практически ничего не меняется: необходимо расставить разрывы в выравниваемых строках таким образом, чтобы «счет по столбцам» был максимален. Счет по столбцу можно считать перебирая все пары символов. Множественное локальное выравнивание обобщить на многомерный случай не так просто. Во-первых, какие-то подстроки могут быть не во всех последовательностях. Во-вторых, последовательности могут содержать дуплицированные участки. Поэтому для решения такой задачи необходимо более точно сформулировать условия выравнивания.

Таким образом, две главные составляющие автоматических методов выравнивания — это непосредственно алгоритм и функция оценки качества полученного результата. На сегодняшний день можно выделить два основных алгоритма выравнивания биологических последовательностей: алгоритм Нидлмана-Вунша и алгоритм Смита-Ватермана. Они представляют собой классический пример задачи динамического программирования. Существуют различные их модификации, использующие эвристики для уменьшения количества шагов алгоритма или требуемого объема памяти, однако,эти методы строят выравнивание без сохранения открытых рамок считывания. В погоне за лучшим счетом происходит потеря биологического смысла результата.

Задача множественного выравнивания с учетом открытых рамок считывания требует других, более сложных подходов. Один из существующих методов решения: построить выравнивание исходной нуклеотидной последовательности на аминокислотном уровне [5]. У такого подхода есть несколько проблем. Во-первых, появление преждевременного стоп-кодона. Во-вторых, так как каждая последовательность переводится с одной и той же рамкой считывания от начала и до конца, то присутствие единственного дополнительного нуклеотида приведет к аномальному переводу и выравниванию.

1 Обзор предметной области

Выравнивание аминокислотных или нуклеотидных последовательностей — это процесс сопоставления сравниваемых последовательностей для такого их взаиморасположения, при котором наблюдается максимальное количество совпадений аминокислотных остатков или нуклеотидов [6]. Различают два вида выравнивания: парное (выравнивание двух последовательностей ДНК, РНК или белков) и множественное (выравнивание трех и более последовательностей).

1.1 Существующие методы поиска гомологий в биологических последовательностях

В генетике под гомологиями понимаются участки белков или ДНК, имеющие сходную последовательность аминокислот или нуклеотидов. Обычно существа, у которых есть гомологичные участки белков или ДНК, имеют общего предка, от которого они и получили такой участок. Поскольку в процессе эволюции ДНК подвергается мутациям, эти участки не обязательно идентичны. В них могут быть случайно заменены, добавлены или удалены нуклеотиды или аминокислоты (рисунок 1). Некоторые мутации, такие, как транслокации и инверсии, приводят к изменениям, затрагивающим большие участки генома. Такие мутации сложно учитывать, поскольку локальное сходство проверять легче, чем глобальное, а в результате глобальных мутаций участки ДНК могут быть соединены в непредсказуемом порядке.

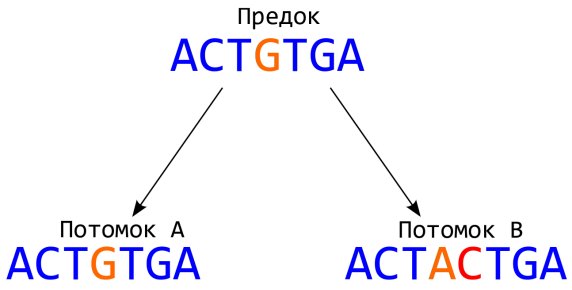

Рис. 1 – Пример мутации

1.1.1 Алгоритм Нидлмана-Вунша

Одним из наиболее распространенных алгоритмов выравнивания является алгоритм Нидлмана-Вунша [7], основанный на двумерном динамическом программировании. Для своей работы алгоритм использует матрицу сходства, которая указывает, насколько схожими можно считать разные нуклеотиды. Использование матрицы позволяет придавать разный вес разным заменам нуклеотидов. Например, поскольку транзиции более вероятны, чем трансверсии, логично считать последовательности, отличающиеся заменой пурина на пурин или пиримидина на пиримидин, более схожими, чем те, которые отличаются заменой пурина на пиримидин или наоборот. Обычно используется симметричная матрица, однако, применение несимметричной матрицы позволяет различать замены в одну и в другую стороны. На рисунке 2 представлен пример матрицы сходства. Здесь А, Г, Т и Ц обозначают, соответственно, аденин, гуанин, тимин и цитозин, а числа в матрице указывают степень сходства между двумя нуклеотидами.

	А	**Г**	**Т**	**Ц**
А	10	−1	−4	−3
Г	−1	7	−3	−5
Т	−4	−3	8	0
Ц	−3	−5	0	9

Рис. 2 – Пример матрицы сходства

Еще один параметр алгоритма — штраф за разрыв последовательности. Он может выражаться произвольной функцией от длины и/или направления разрыва. Для определенности будем рассматривать линейный штраф за разрыв, определяющийся параметром d (за разрыв длинны n будет начислен штраф $d \cdot n$).

На вход алгоритм получает матрицу сходства S, параметр штрафа d и две последовательности (строки), которые необходимо выровнять. Для получения результата выполняется построение матрицы $F_{i,j}$, где i и j изменяются от нуля до длины, соответственно, первой и второй строк. Вначале алгоритм инициализирует $F_{i,0}$ и $F_{0,j}$ равными, соответственно, $d \cdot i$ и $d \cdot j$ для всех i и j. Затем происходит вычисление оставшихся элементов матрицы по формуле 1.

$$F_{i,j} = max \begin{cases} F_{i-1,j-1} + S_{A_i,B_j} \\ F_{i-1,j} + d \\ F_{i,j-1} + d \end{cases} \tag{1}$$

После того, как матрица посчитана, необходимо определить, каким путем появилось значение в правом нижнем углу. Например, если $F_{i,j} = F_{i-1,j-1} + S_{A_{i-1},B_{j-1}}$, то элемент (i, j) появился из элемента $(i-1, j-1)$, и т. д. Элементы в верхней строке произошли из элементов левее себя, элементы из левого столбца — из элементов выше себя. Переход вида $(i, j) \to (i-1, j-1)$ означает, что i-му символу в первой строке соответствует j-й символ во второй строке. Переход вида $(i, j) \to (i-1, j)$ означает, что i-му символу первой строки ничего не соответствует, а переход $(i, j) \to (i, j-1)$ — что j-му символу второй строки ничего не соответствует. Путь в матрице от левого верхнего угла к правому нижнему даст искомое выравнивание последовательностей.

Очевидно, что алгоритм всегда ищет выравнивание с максимальным счетом, так как строя матрицу F, он рассматривает всевозможные варианты размещения одной строки относительно другой. Время работы и количество используемой памяти пропорционально произведению длин последовательностей.

1.1.2 Алгоритм Смита-Ватермана

Алгоритм Смита-Ватермана [8] аналогичен алгоритму Нидлмана-Вунша, но решает задачу локального выравнивания: находит подстроки первой и второй строк, обладающие максимальным сходством.

На вход алгоритм получает матрицу сходства S, две последовательности и два вектора I и D, вектор стоимостей добавления и вектор стоимостей удаления, соответственно. Элементы матрицы $F_{i,0}$ и $F_{0,j}$ инициализируются нулями. Вычисление оставшихся элементов происходит по формуле 2.

$$F_{i,j} = max \begin{cases} F_{i-1,j-1} + S_{A_i,B_j} \\ F_{i-1,j} + D_{A_i} \\ F_{i,j-1} + I_{B_j} \\ 0 \end{cases} \quad (2)$$

Для получения выравнивания необходимо найти максимальный элемент в матрице. Если переходить от этого элемента по цепочке предыдущих, то путь закончится в каком-то нулевом элементе. Индексы этих двух элементов равны индексам начал и концов подстрок: первые индексы — в первой строке, вторые — во второй. Путь интерпретируется так же, как и в алгоритме Нидлмана-Вунша.

Видно, что оба алгоритма похожи друг на друга. Они имеют одинаковую сложность и затраты по памяти, что делает такие алгоритмы неприемлемыми для работы с большим количеством генетического материала.

1.1.3 Алгоритм Хиршберга

Оба предыдущих алгоритма требуют объем памяти, пропорциональный произведению длин выравниваемых последовательностей, что затрудняет обработку больших строк, поэтому очень важно иметь методы, уменьшающие затраты памяти без критического увеличения времени счета. В 1975 году был предложен алгоритм Хиршберга, значительно сокращающий затраты памяти [9]. Он позволяет вычислять оптимальное выравнивание строк длины n и m, используя $O(n+m)$ количество памяти, но примерно вдвое большее времени счета по сравнению с алгоритмом Нидлмана-Вунша.

Идея алгоритма состоит в том, что одна из двух входных последовательностей разбивается на две части, и исходная задача сводится к двум, меньшим, задачам выравнивания второй входной последовательности с каждой из частей. Решение подзадач осуществляется путем аналогичного сведения к подзадачам. На рисунке 3 показана схема разбивки задачи на две подзадачи: верхнюю, которая решается в прямоугольнике A исходной таблицы, и нижнюю — в прямоугольнике B. Последовательности имеют длины n и m, соответственно. Для разбиения каждой задачи на подзадачи необходимо вычислить значение k^*. При этом используется объем памяти, линейно зависящий от m. Верхняя задача заключается в выравнивании строки с длинами не больше $n/2$ и k^*, а нижняя – с длинами не больше $n/2$ и $m - k^*$.

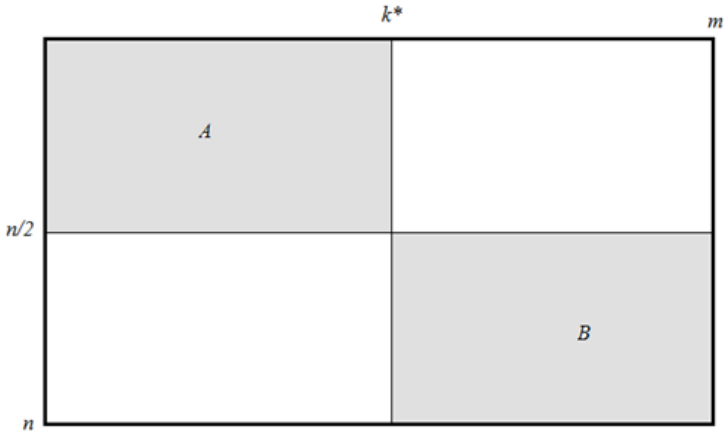

Рис. 3 – Разделение задачи выравнивания на две подзадачи

Для представления задач в алгоритме Хиршберга можно использовать бинарные деревья [10]. Узлам дерева соответствуют подзадачи, которые заключаются в выравнивании меньших подпоследовательностей. Каждый узел дерева хранит в памяти границу прямоугольной области, в которой решается соответствующая задача динамического программирования. Дерево в процессе работы алгоритма строится по уровням. Сначала оно состоит только из корневого узла, который соответствует прямоугольнику $[0, 0] \times [n, m]$. Создание двух узлов эквивалентно разбиению задачи на две подзадачи и разделению области решения на две, меньшего размера.

Алгоритм Хиршберга заключается в обходе полного дерева всех подзадач. Результат выравнивания можно будет получить, если пройтись по листьям построенного дерева (рисунок 4). Для оптимизации вычислений можно выполнять обход (решение подзадач) только части вершин дерева: тех, которые удалены от корня на величину, не превосходящую заранее заданную константу h — максимальную глубину обхода дерева. При достижении глубины дерева h или минимального размера прямоугольника применяется алгоритм Нидлмана-Вунша, который работает вдвое быстрее алгоритма Хиршберга.

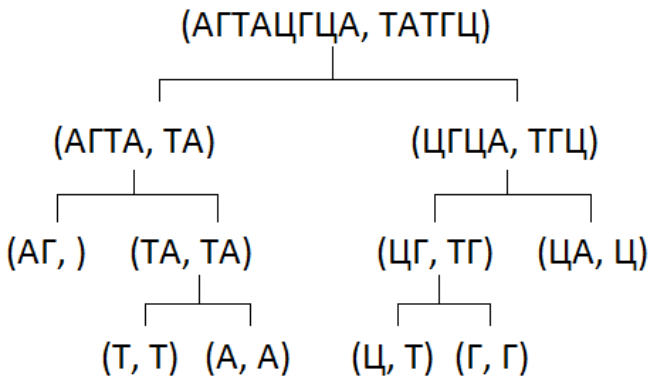

Рис. 4 – Дерево подзадач для алгоритма Хиршберга

Дополнительное ускорение можно получить за счет распараллеливания. Заметим, что на каждом шаге алгоритма полученные подзадачи никак не связаны между собой, и, следовательно, их решения могут вычисляться в отдельных потоках.

1.2 Алгоритмы множественного выравнивания

В пункте 1.1 были рассмотрены основные подходы для получения парного выравнивания. Для некоторых областей биоинформатики задачу поиска выравнивания необходимо переложить на многомерный случай, например при реконструкции эволюционной последовательности (получение филогенетических деревьев) или при выявлении шаблона функциональных семейств и сигналов ДНК.

1.2.1 Выравнивание в «кубе»

Рассмотрим задачу выравнивания трех последовательностей A_1, A_2 и A_3. Построим трехмерную матрицу F (рисунок 5) с длинами сторон $len(A_i)$, $i = 1, 2, 3$, где $len(A_i)$ — длина i-ой строки. Аналогично алгоритму Нидлмана-Вунша (пункт 1.1.1) определим значение в ячейке $F_{i,j,k}$ $i = 1 \dots len(A_1)$, $j = 1 \dots len(A_2)$, $k = 1 \dots len(A_3)$ по формуле 3.

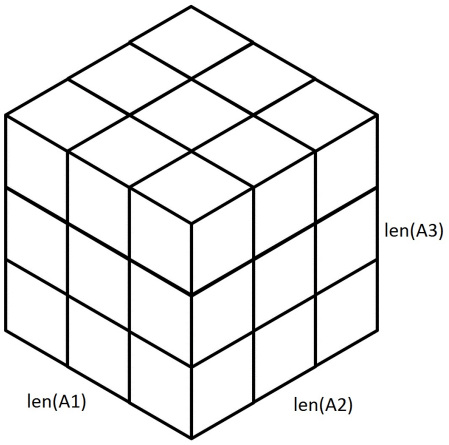

Рис. 5 – Матрица F для выравнивания трех последовательностей

$$F_{i,j} = max \begin{cases} F_{i-1,j-1,k-1} + S(A_{1_i}, A_{2_j}) + S(A_{1_i}, A_{3_k}) + S(A_{2_j}, A_{3_k}) \\ F_{i-1,j-1,k} + S(A_{1_i}, A_{2_j}) + 2d \\ F_{i-1,j,k-1} + S(A_{1_i}, A_{3_k}) + 2d \\ F_{i,j-1,k-1} + S(A_{2_j}, A_{3_k}) + 2d \\ F_{i-1,j,k} + 3d \\ F_{i,j-1,k} + 3d \\ F_{i,j,k-1} + 3d \end{cases} \tag{3}$$

Можно заметить, что каждая грань куба — это парное выравнивание двух последовательностей с учетом некоторой части третей, что и дает в итоге полный перебор всех возможных вариантов. Нулевые грани куба $F_{0,j,k}$, $F_{i,0,k}$ и $F_{i,j,0}$ заполняются аналогично алгоритму Нидлмана-Вунша.

Что бы получить ответ необходимо найти путь от ячейки $F_{len(A_1),len(A_2),len(A_3)}$, где записан итоговый счет за выравнивание, до $F_{0,0,0}$. Так как имеется всего семь возможных перемещений в кубе и $len(A_1) \cdot len(A_2) \cdot len(A_3)$ ячеек, то сложность алгоритма можно оценить как $O(7 \prod_{i=1}^{3} len(A_i))$.

Не составляет большого труда «продлить» аналогичным образом это решение на n-мерный случай и получить «честное» многомерное выравнивание. Под словом «честное» подразумевается, что рассмотрены все возможные варианты выравнивания последовательностей и полученный результат всегда имеет максимальный счет. Единственный недостаток — слишком большая вычислительная сложность алгоритма: $O((2^n - 1) \prod_{i=1}^{n} len(A_i))$, что делает такой подход совершенно неприменимым для выравнивания большого числа и/или длинных последовательностей.

1.2.2 Выравнивание выравниваний. Алгоритм Clustal

Другой подход заключается в в получении парного выравнивания между первыми двумя последовательностями, после чего полученный результат выравнивается с третей и так далее. То есть, если f — функция вычисления парного выравнивания, а $A_1, \ldots, A_n$ — выравниваемые последовательности, то алгоритм можно условно записать формулой 4.

$$f(f(f(\dots f(f(A_1, A_2), A_3)\dots), A_{n-1}), A_n) \tag{4}$$

Очевидно, что результат алгоритма будет зависеть от порядка исходных последовательностей. Существуют различные соображения по поводу наиболее правильного выбора этого порядка. Можно не ограничиваться выравниваниями типа «последовательность против выравнивания», но так же производить выравнивание «выравнивание против выравнивания». Например, если есть четыре последовательности, из которых первая очень похожа на четвертую, вторая — на третью, а гомология между остальными парами (1-2, 1-3, 2-4, 3-4) более слабая, то разумно сначала сделать два парных выравнивания: первой последовательности с четвертой и второй с третей, а затем уже выровнять эти два выравнивания друг с другом.

Похожим образом работает Clustal — один из самых популярных алгоритмов множественного выравнивания. По сути это жадный алгоритм с «умным» способом выбора пар. Сначала происходит построение всех парных выравниваний, после чего по полученным результатам строится «дерево-подсказка». На рисунке 6 представлен пример возможного дерева. Для четырех последовательностей A_1, A_2, A_3 и A_4 строится таблица (на рисунке слева) числа в которой обозначают их схожесть друг с другом. Видно, что самые близкие последовательности — A_1 и A_3, их выравнивание будет первым, затем оно выравнивается с A_4 и в конце с A_2.

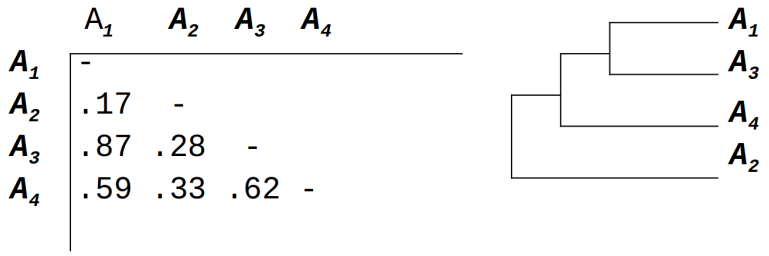

Рис. 6 – Построение дерева-подсказки для алгоритма Clustal

1.3 Выравнивание с учетом открытых рамок считывания

ЗАГОТОВКА

1.4 Представление генетической информации в электронном виде

Поскольку различных нуклеотидов и стандартных аминокислот немного, для их кодирования используют один символ — первую букву из названия. С другой стороны, названия многих аминокислот начинаются с одинаковых букв, поэтому для кодирования приходится использовать те, которые остаются незанятыми.

1.4.1 Формат FASTA

В формате FASTA [11] строчка, начинающаяся с символа '>', называется строкой описания. Она содержит имя последовательности и некоторую дополнительную информацию, предназначенную для идентификации. Другие строки, начинающиеся с символа

';', являются комментариями и игнорируются. За строкой описания следует код последовательности. При кодировании нуклеотидов буквами A, C, G, T и U кодируют, соответственно, аденин, цитозин, гуанин, тимин и урацил. Обычно, длинные последовательности разбивают на несколько строк длиной не более 80 символов — это не правило формата, но представление данных таким образом выглядит более наглядно для человека.

1.4.2 Формат FASTQ

FASTQ — формат представления биологической последовательности совместно с данными о качестве. Он используется для представления данных секвенирования. При кодировании уровней качества используются символы из таблицы ASCII от '!' до '~'.

Существует два различных способа выражать уровень качества через вероятность ошибки: формулы 5 и 6, где Q — уровень качества, а p — вероятность, что элемент последовательности ошибочный. При малых значениях p эти способы дают практически идентичные результаты, но с ростом p уровни качества начинают заметно различаются (рисунок 7).

$$Q = -10 \cdot \log_{10} p \tag{5}$$

$$Q = -10 \cdot \log_{10} \frac{p}{(1-p)} \tag{6}$$

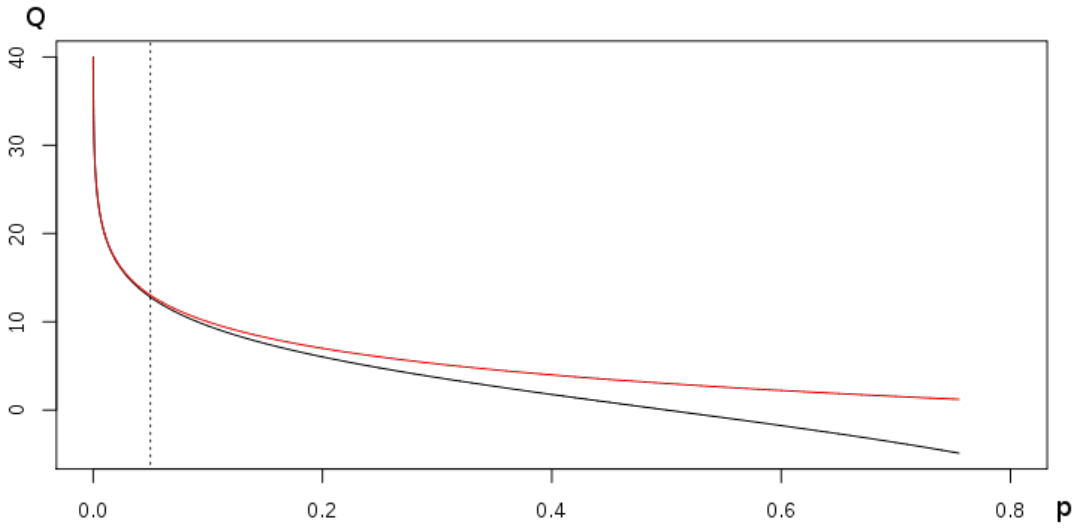

Рис. 7 – График уровней качества для формул 5 (красная) и 6 (черная)

Файл в формате FASTQ содержит четыре строки для каждой последовательности. Первая строка начинается с символа '@', после которого идет описание последовательности (строка описания). Следующая строка содержит набор символов, кодирующих саму последовательность аналогично формату FASTA. За ней идёт строка, начинающаяся с символа '+', содержащая дополнительное описание последовательности. Последняя строка содержит уровни качества.

1.4.3 Формат GenBank

Запись в формате GenBank состоит из двух секций: секции аннотации и секции данных [12]. В первой хранится всевозможная информация о последовательности: из какого организма получена, ссылки на другие работы, различные примечания, а во второй — сама последовательность, аналогично формату FASTA. Начало секции аннотации отмечается кодовым словом «LOCUS», а секция данных начинается со слова «ORIGIN». В конце описания последовательности ставится специальный маркер «//». Формат GenBank, по сравнению с форматами FAST и FASTQ, позволяет представить больше дополнительной информации о последовательности.

ЗАКЛЮЧЕНИЕ

Список литературы

[1] Миронов Андрей Александрович. Лекция "Введение в биоинформатику". Режим доступа — URL: http://mipt.ru/dbmp/student/files/bioinformatics/public_lection/.

[2] Выравнивание последовательностей [электронная публикация]. — URL: https://ru.wikipedia.org/wiki/Выравнивание_последовательностей.

[3] Двоичные коды с исправлением выпадений, вставок и замещений символов. Доклады Академий Наук СССР. 163.4:845-848. 1965.

[4] Humberto Carrillo, David Lipman "The Multiple Sequence Alignment Problem in Biology" on Applied Mathematics Vol. 48, No. 5. (Oct., 1988).

[5] MACSE: Multiple Alignment of Coding SEquences Accounting for Frameshifts and Stop Codons [электронная публикация]. — URL: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0022594.

[6] А.В. Бутвиловский Е.В. Барковский В.Э. Бутвиловский. Выравнивание аминокислотных и нуклеотидных последовательностей.

[7] Needleman S. B., Wunsch C. D. "A general method applicable to the search for similarities in the amino acid sequence of two proteins" on Journal of Molecular Biology Vol. 48, no. 3. 1970.

[8] Smith T. F., Waterman M. S. "Identification of common molecular subsequences" on Journal of Molecular Biology Vol. 147, no. 1. 1981.

[9] Hirschberg D. S. A linear space algorithm for computing maximal common subsequences.

[10] Параллельный алгоритм глобального выравнивания с оптимальным использованием памяти [электронная публикация]. — URL: http://www.science-education.ru/107-8139.

[11] What is FASTA format? [электронная публикация]. — URL: http://zhanglab.ccmb.med.umich.edu/FASTA/.

[12] GenBank format [электронная публикация]. — URL: http://quma.cdb.riken.jp/help/gbHelp.html.