Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

**УТВЕРЖДАЮ**
заведующий кафедрой
_____ В. В. Шайдуров
"22" июня 2020 г.

# БАКАЛАВРСКАЯ РАБОТА

**Направление** 02.03.01 Математика и компьютерные науки

# ИСПОЛЬЗОВАНИЕ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ И СВЁРТОЧНЫХ ФУНКЦИЙ ДЛЯ СРАВНЕНИЯ НУКЛЕОТИДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Научный руководитель
к. ф.-м. н., доцент _____ 22 июня 2020 Е.Д. Карепова

Выпускник _____ 22 июня 2020 А.А. Молявко

Красноярск 2020

# РЕФЕРАТ

Выпускная квалификационная работа на тему «Использование быстрого преобразования Фурье и свёрточных функций для сравнения нуклеотидных последовательностей» содержит 38 страниц текстового документа, состоит из *Введения*, трёх глав, заключения, выводов, 7 рисунков и списка литературы; список литературы содержит 30 использованных источников.

Ключевые слова: СИМВОЛЬНЫЕ ПОСЛЕДОВАТЕЛЬНОСТИ, ПОИСК, СРАВНЕНИЕ, СВЁРТКА, ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ, БЫСТРОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ.

Цель данной работы — программная реализация метода сравнения символьных последовательностей на основе быстрого преобразования Фурье.

Сравнение символьных последовательностей — важная задача как в прикладных, так и в фундаментальных областях математики. Проблема сравнения символьных последовательностей важна также в биоинформатике, где она является ключевым инструментом для проведения самых разных исследований. Требуется уметь находить в исследуемых последовательностях точно совпадающие участки, а также участки с теми или иными несоответствиями, такими как замены, вставки и/или выпадения отдельных символов.

В работе предложен и протестирован алгоритм сравнения символьных последовательностей, основанный на следующем. Каждая из сравниваемых последовательностей преобразуется в набор бинарных, количество бинарных последовательностей зависит от количества различных символов в исходной последовательности. Все бинарные последовательности интерпретируются как коэффициенты полиномов. Произведение данных полиномов, вычисленное с помощью быстрого преобразования Фурье, однозначно определяет свёртку исходных бинарных последовательностей, по величине которой можно судить о наличие совпадений.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Задача поиска повторов или общих подпоследовательностей в символьных последовательностях является одной из классических задач математики и информатики. Она имеет большое количество различных приложений во многих областях современной науки, в том числе и в области биоинформатики. Наиболее значимой здесь является задача организации быстрого и эффективного сравнения больших (длиной до $10^{12}$ символов) последовательностей, представляющих собой цепочки ДНК. Кроме этого, также очень важно уметь осуществлять быстрый поиск по образцу в таких обширных массивах данных. Несмотря на заметный прогресс в этом направлении, достигнутый в основном благодаря развитию вычислительной техники, обе эти задачи далеки от окончательного решения.

В настоящее время большинство алгоритмов, так или иначе решающих эту задачу, базируются на идее выравнивания (*alignment*). Суть выравнивания в том, что мы каким-то образом находим в сравниваемых последовательностях общее «ядро», а затем расширяем его до тех пор, пока не накопится слишком много ошибок. Выравнивание является исторически первым методом, получившим широкое распространение, однако при этом обладающее рядом существенных недостатков, к которым относятся

1) расходимость метода и следовательно его неспособность сравнивать последовательности длиннее $l \approx 10^4$, и
2) произвол в выборе штрафных функций, который не может быть устранён формальными методами.

**Целью** настоящей работы является разработка и программная реализация нового метода сравнения символьных последовательностей, основанного на преобразовании символьных последовательностей в цифровые и работе с ними, на примере генетических текстов, а также его тестирование и расширение.

Основная идея данного метода принадлежит доктору физико-математических наук, профессору Владимиру Викторовичу Шайдурову.

Для достижения указанной цели были поставлены следующие **задачи**.

1. Анализ научной литературы по теме выпускной работы.

2. Разработка метода сравнения символьных последовательностей, основывающегося на вычислении функции свёртки двух полиномов и использовании быстрого преобразования Фурье.
3. Реализация разработанного алгоритма на языке C#, отладка и тестирование программы.
4. Проведение серии вычислительных экспериментов с целью определения пределов применимости разработанного метода и его алгоритмической реализации.
5. Проведение серии вычислительных экспериментов на реальных генетических текстах с целью сравнения с иными распространёнными приложениями, использующимися для сравнения и поиска в последовательностях.

Работа состоит из введения, трех глав, заключения и списка цитируемой литературы. В первой главе содержится постановка задачи данной работы, вводятся основные понятия и описываются подходы к решению поставленной проблемы. Вторая глава содержит подробное описание разработанного алгоритма. В третьей главе приведены результаты вычислительных экспериментов на различных типах входных данных, а также ведётся обсуждение полученных результатов и возможных путей развития метода. Список литературы состоит из 30 источников.

Особая благодарность доктору физико-математических наук, профессору Михаилу Георгиевичу Садовскому за биологическую постановку задачи, полезные обсуждения и помощь в освоении предметной области, а также Игорю Анатольевичу Боровикову из EADP Data and AI, Electronic Arts in Redwood Shores, Калифорния, США за стимулирующий интерес к работе.

# 1 Основные понятия и подходы

В настоящей главе изложены точная постановка задачи данного исследования, строгие определения и утверждения, которые будут использоваться в дальнейшем, а также приведён анализ литературы по обсуждаемой проблеме.

## 1.1 Постановка задачи исследования

Основной задачей настоящей работы является программная реализация и анализ эффективности работы нового алгоритма сравнения символьных последовательностей и поиска в них заданных тем или иным образом подпоследовательностей. Эта задача весьма актуальна в настоящее время в связи с развитие биоинформатики и молекулярной биологии, для которых сравнение и поиск по образцу в нуклеотидных (либо аминокислотных) последовательностях являются основным инструментом анализа [1]. Следует подчеркнуть, что задача сравнения двух и более последовательностей, а также поиска в них подпоследовательностей, не ограничивается только биоинформатикой и молекулярной биологией: эта задача актуальна для многих разделов как фундаментальной, так и прикладной математики, например, актуарной математики [2].

Дадим строгие определения и введём точные понятия. Начнём с того, что укажем на две, хотя и разные, но тесно связанные постановки вопроса о поиске по образцу:

– поиск в последовательности всех точных совпадений с заданным образцом, и

– поиск неточных совпадений, допускающих те или иные (малые) ошибки в точном совпадении.

Вторая задача является весьма трудоёмкой, а её теоретическая и программная сложность на порядки превосходят аналогичные показатели для задачи точного поиска. Тем не менее, решение первой задачи — поиска точных совпадений — всегда является неотъемлемой частью решения второй — поиска совпадений, допускающих неточности и ошибки.

### 1.1.1 Задача поиска точных соответствий

Сформулируем строго задачу поиска точных совпадений. Пусть имеются две символьные последовательности $\mathfrak{T}$ и $\mathfrak{L}$ из конечного алфавита $\aleph$. Всюду далее мы будем рассматривать четырёхбуквенный алфавит $\aleph = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$, соответствующий алфавиту нуклеотидных последовательностей. Длину последовательности будем обозначать $|\cdot|$. Будем полагать, что $|\mathfrak{T}| \ll |\mathfrak{L}|$. Требуется найти все **точные** вхождения подпоследовательности $\mathfrak{T}$ в последовательность $\mathfrak{L}$.

Простейшим — и, очевидно, весьма неэффективным — алгоритмом решения этой задачи является полный перебор. К настоящему времени предложено большое число различных алгоритмов поиска точных совпадений, имеющих сложность $O(n \cdot \ln n)$, где $n$ — длина последовательности, по которой ведётся поиск. Не проводя подробного анализа всех существующих алгоритмов поиска точных совпадений, отметим, что в целом прогресс именно в этой области теоретической информатики весьма велик.

### 1.1.2 Задача поиска неточных соответствий

Задача поиска неточных совпадений существенно труднее и в теоретическом, и в прикладном плане. Дело в том, что в постановке этой задачи есть две большие содержательные, а не технические сложности.

I. Проблема определения расстояния на множестве символьных последовательностей состоит в том, что содержательно такие метрики существенно менее продуктивны, чем метрики в непрерывных пространствах.

II. Весьма затруднительна задача формального описания всех классов неточностей, допустимых при сравнении, и способов учёта их влияния на метрику близости двух последовательностей.

В классических работах В. И. Левенштейна [3–5] предложен подход, ставший основой для всей группы методов, получивших название *выравнивание* (alignment); основу подхода составляет система штрафных функций. Идея, стоящая за таким подходом, прозрачна и интуитивно понятна: разные ошибки дают разный вклад в наблюдаемое несоответствие, и эту разницу следует учитывать. Не обсуждая здесь всех методов и программных реализаций вы-

равнивания (см. по этому поводу, например, работы [1, 6–10]), подчеркнём только, что указанный метод обладает двумя значительными недостатками, упомянутыми выше: расходимостью и произволом в выборе системы штрафных функций (весов) при вычислении редакционного расстояния [3–5] при осуществлении выбора того или иного варианта выравнивания. Расходимость означает, что при выборе системы штрафных (весовых) функций приходится увеличивать штрафы для мутаций, располагающихся ближе к концам сравниваемых участков, иначе процесс расширения этих участков — при формально определяемом в силу этого метода сходстве — остановится только тогда, когда будет достигнут конец последовательности.

Произвол в выборе системы штрафных функций при определении редакционного расстояния означает, что можно выбрать различные варианты этих штрафов, при этом результат может различаться весьма существенно и формального способа выбора единственного варианта системы штрафов не существует.

Ещё одна проблема, до сих пор не имеющая эффективного решения, — это проблема вставок и выпадений. Пусть в двух последовательностях $\mathfrak{T}_1$ и $\mathfrak{T}_2$ есть общая подпоследовательность $\mathfrak{L}$, то есть $\mathfrak{L}$ входит как в $\mathfrak{T}_1$, так и в $\mathfrak{T}_2$. Представим теперь, что в одной из последовательностей (например, в $\mathfrak{T}_1$ для определённости) фрагмент $\mathfrak{L}$ «слегка испорчен»: он оказался разделён на две части $\mathfrak{L}_1$ и $\mathfrak{L}_2$, так, что между этими частями находится сравнительно малая вставка; обозначим её $\mathfrak{l}$. В одной последовательности $\mathfrak{L} = \mathfrak{L}_1 \cup \mathfrak{L}_2$, в другой — $\mathfrak{L}' = \mathfrak{L}_1 \cup \mathfrak{l} \cup \mathfrak{L}_2$ (здесь символом $\cup$ обозначена конкатенация). Такая (пусть даже малая) неточность в сравниваемых последовательностях представляет собою очень большую трудность для существующих к настоящему моменту времени методов поиска и сравнения.

Стоит заметить, что несоответствие типа «выпадение» является симметричной ситуацией — вставка «лишнего» символа в одну из последовательностей есть то же самое, что удаление символа из второй последовательности.

### 1.1.3 Задача сравнения символьных последовательностей

Пусть имеются две символьные последовательности $\mathfrak{T}_1$ и $\mathfrak{T}_2$ из конечного алфавита $\aleph$. Тогда под решением задачи сравнения двух символьных последовательностей понимается следующий результат: указать список всех подпоследовательностей длины не менее $l_{\min}$, встречающихся в $\mathfrak{T}_1$ и $\mathfrak{T}_2$ одновременно, быть может, с указанием числа их копий и локализацией (координат вхождения в каждую из последовательностей).

Выбор величины $l_{\min}$ определяется характером тех задач, которые решаются с помощью сравнения двух (либо нескольких) последовательностей; очевидно, что случай $l_{\min} = 1$ не интересует никого и никакого содержательного — при том, что формально он может быть получен — смысла не имеет.

## 1.2 Обзор литературы

Задача поиска по образцу в символьных последовательностях тесно связана с задачей сравнения символьных последовательностей. Обе задачи важны как сами по себе в теоретической математике и информатике, так и в прикладных областях — в первую очередь в биоинформатике и геномике. В этих направлениях достигнуты значительные результаты, однако обе задачи едва ли могут считаться окончательно решёнными. К настоящему времени появляются всё новые и новые алгоритмы и подходы, которые обогащают арсенал средств, используемый исследователями в различных областях.

Подход к сравнению символьных (генетических) последовательностей, основанный на выравнивании, был предложен в работе [11]. Кратко, идея метода заключается в следующем: имеются две последовательности и одну надо «подогнать» под другую. Для такой подгонки (выравнивания) разрешается с одной из последовательностей производить т. н. элементарные преобразования из некоторого фиксированного набора: замену символа, находящегося в $j$-ой позиции, на другой, а также удаление и/или вставку символа. Каждой такой операции приписывается некоторый вес (число); для определения наилучшей подгонки находят её суммарный вес. Наилучшим считается выравнивание, имеющее минимальный суммарный вес.

Идея выравнивания опирается на классические работы В. Левенштейна

по кодам, исправляющим ошибки [3–5,12,13], в которых было введено понятие *редакционного расстояния* (*edit distance*). Подход к сравнению генетических последовательностей на основе редакционного расстояния [11] получил широкое распространение в связи с существовавшими в то время ограничениями в вычислительных возможностях. При этом данный метод обладает рядом недостатков.

Первый недостаток — расходимость. Это означает, что система штрафных функций должна быть искусственно построена так, чтобы по мере роста выровненной части последовательностей суммарный штраф рос достаточно быстро. Если этого не делать, то метод даёт бесконечно длинное совпадение по типу *всё равно всему*.

Второй важный недостаток — вырожденность получаемых сравнений. С ростом длины сравниваемых участков (т. е. по мере увеличения длины участков, считающихся совпадающими, пусть и неточно) происходит вырождение: несколько существенно различных укладок[1] дают одно и то же значение суммарной штрафной функции и нет никаких формальных и тем более строгих способов выбрать одну из них. Выбор здесь происходит экспертным образом, фактически — по желанию исследователя.

В результате, в задачах геномики, генетики и биоинформатики наиболее распространённым методом сравнения стало выравнивание. К настоящему времени предложен ряд модификаций метода выравнивания, направленных на преодоление указанных недостатков [1, 6, 14–20]. Основным способом решения этих проблем является предварительная обработка сравниваемых последовательностей — использование различных алгоритмов сжатия, в том числе учитывающих биологическую природу последовательностей, а также вариации способов определения редакционного расстояния.

Ещё одним примером развития идеи выравнивания является MSA (*multiple sequence alignment*) — множественное выравнивание, которое отличается тем, что сравниваются не две, а три и более последовательностей [21, 22]. В таком случае вес ошибки назначаются с учётом того, сколько последовательностей имеют совпадение в данной позиции, что может улучшить сходимость

[1]Под укладкой будем понимать ту часть одной из сравниваемых последовательностей, которая подвергается преобразованиям.

метода.

Несмотря на широкое распространение метода выравнивания, ведутся исследования в области алгоритмов, принципиально не использующих эту идею. Примеры таких работ приведены в [9]. Суть одного из методов заключается в том, что для каждой последовательности строятся частотные словари всех подстрок фиксированной длины, затем вычисляется их общий «предок», с которым сравнивается каждая последовательность [23, 24]. Еще одним методом, не базирующемся на выравнивании, является сравнение специально построенных прореженных словарей длинных $k$-меров [25, 26].

Идейно близкой работой к нашей является [27]. Здесь реализована та же идея, что и в нашем случае — преобразование символьной последовательности в набор числовых, при работе с которыми можно использовать инструменты математического и комплексного анализа. Отличие состоит в том, что в этой работе последовательности преобразуются не в бинарные. Другой важной особенностью этой работы является применение преобразования Фурье для анализа последовательностей. Однако автор данной работы не использовал предложенные им подходы для всеобъемлющего сравнения генетических последовательностей большой длины. Один из возможных вариантов алгоритмической реализации метода анализа и сравнения символьных последовательностей с помощью преобразования Фурье защищён патентом США [28].

Одной из классических задач биоинформатики является построение филогении (системы родства на основе подобия нуклеотидных последовательностей тех или иных генов). В настоящее время существует большое разнообразие вычислительных инструментов для построения филогении. В работе [29] предложен подход к решению данной проблемы, основанный на преобразовании последовательностей в числовые и применении к ним дискретного преобразования Фурье.

## 1.3 Основные понятия, используемые при построении нового алгоритма сравнения символьных последовательностей

Начнем издалека: для начала расскажем о том, как определяется операция свёртки для непрерывных функций, а затем для дискретных функций и для последовательностей.

### 1.3.1 Понятие свёртки двух функций

Пусть даны две функции $f,\ g\colon \mathbb{R} \to \mathbb{R}$, которые измеримы на $\mathbb{R}$ в смысле меры Лебега. Тогда свёрткой этих функций называют функцию $h\colon \mathbb{R} \to \mathbb{R}$, определяемую следующим образом:

$$h(t) = \int\limits_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau\,. \tag{1}$$

Функцию $h$ можно интерпретировать как показатель «схожести» одной функции с отраженной и сдвинутой копией другой, а в каждой конкретной точке $t_0$ свёртка $h(t_0)$ — это площадь под графиком функции $f(\tau)g(t_0-\tau)$.

Если $f$ и $g$ — дискретные функции, определенные на одном и том же множестве $\{t_0,\ldots,t_k,\ldots\}$, тогда их свёртка определяется как сумма произведений значений $f$ на также сдвинутую и отраженную $g$:

$$h(t) = \sum_{t_i:t_0\le t_i\le t} f(t_i)g(t-t_i), \qquad t\ \in\ \{t_0,\ldots,t_k,\ldots\}\,. \tag{2}$$

К примеру, если функции $f(t)$ и $g(t)$ определены на множестве $\mathbb{Z}^+$ целых неотрицательных чисел, т. е. $t\ \in\ \{0,1,2,\ldots\}$, то их свёртка $h$ определяется следующим образом:

$$h(0) = f(0)g(0),$$
$$h(1) = f(0)g(1-0) + f(1)g(1-1) = f(0)g(1) + f(1)g(0),$$
$$h(2) = f(0)g(2) + f(1)g(1) + f(2)g(0)\,,$$
$$\ldots$$

Частным случаем свёртки дискретных функция является свёртка последовательностей. Более наглядно это можно продемонстрировать следующим образом: пусть $f = \{f_i\}_{i=0}^{n},\ g = \{g_i\}_{j=0}^{m},\ (m \le n)$. Требуется найти

последовательность $h = \{h_k\}_{k=0}^{m+n}$, которая является свёрткой $f$ и $g$. Запишем исходные последовательности одну под другой следующим образом:

$$\begin{array}{cccccc} & & & f_0 & f_1 & \cdots & f_n \\ g_m & \cdots & g_1 & g_0 & & & \end{array} \tag{3}$$

Затем «подвинем» вторую последовательность на один элемент вправо:

$$\begin{array}{cccccc} & & & f_0 & f_1 & \cdots & f_n \\ g_m & \cdots & g_2 & g_1 & g_0 & & \end{array} \tag{4}$$

Вычисляя для каждой такой укладки попарные произведения элементов, находящихся друг под другом и складывая их, мы как раз и получим свёртку исходных последовательностей $f_i$ и $g_j$. Из (3) и (4) имеем $h_0 = f_0 g_0$, $h_1 = f_1 g_0 + f_0 g_1$. Аналогичным образом вычисляем элементы $h$ до тех пор, пока не дойдем до $h_m$. На этом шаге мы получим следующее:

$$\begin{array}{cccccc} f_0 & f_1 & \cdots & f_m & \cdots & f_n \\ g_m & g_{m-1} & \cdots & g_0 & & \end{array}$$

Отсюда $h_m = \sum_{i=0}^{m} f_i g_{m-i}$. Если мы продолжим двигать нижнюю последовательность вправо, то первые элементы последовательности $f_i$ уже не будут ни с кем пересекаться, но вычисление свёртки на этом не заканчивается. Мы можем двигать последовательность и вычислять суммы попарных произведений соответствующих элементов $f_i$ и $g_j$, пока не получим следующее:

$$\begin{array}{cccccc} f_0 & \cdots & f_n & & & \\ & & g_m & g_{m-1} & \cdots & g_0 \end{array}$$

Последним элементом свёртки станет $f_n g_m$, а её общая длина составит $m+n+1$, так как элемент $g_m$ сначала «пробежит» все $f_i$ (коих $n+1$), а затем ещё $m$ шагов до пересечения $f_n$ и $g_0$.

Заметим, что если последовательности $f_i$ и $g_j$ принимают значения только 0 или 1, то для каждой «укладки» этих последовательностей значение свёртки будет в точности отображать количество пар $\{1, 1\}$ в текущем выравнивании. Именно этот факт лежит в основе нашего алгоритма сравнения двух символьных последовательностей.

### 1.3.2 Дискретное преобразование Фурье и его свойства

Пусть у нас есть многочлен степени $n-1$:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} . \tag{5}$$

Каждому многочлену степени $n-1$ можно поставить в соответствие вектор длины $n$, компонентами которого являются коэффициенты данного многочлена. Пусть $\mathbf{A} = \{a_0, a_1, \ldots, a_{n-1}\}$ — вектор коэффициентов многочлена $P(x)$.

Дискретное преобразование Фурье $\mathbb{F}(\mathbf{A})$ (ДПФ, в англоязычной литературе DFT, Discrete Fourier Transform) для многочлена $P(x)$ (или вектора $\mathbf{A} = \{a_0, a_1, \ldots, a_{n-1}\}$, составленного из его коэффициентов) — это вектор значений этого многочлена в точках $x_k = \exp\left(\frac{2\pi i}{n} k\right),\ k = 0, \ldots, n-1$, то есть во всех значениях корня степени $n$ из единицы. Следовательно,

$$\begin{aligned}\mathbb{F}(P(x)) = \mathbb{F}(\mathbf{A}) = \mathbb{F}(a_0, a_1, \ldots, a_{n-1}) = \\ = \left(P\left(e^0\right), P\left(e^{\frac{2\pi i}{n}}\right), P\left(e^{\frac{4\pi i}{n}}\right), P\left(e^{\frac{6\pi i}{n}}\right), \ldots, P\left(e^{\frac{2(n-1)\pi i}{n}}\right)\right) ,\end{aligned} \tag{6}$$

где $P\left(e^{\frac{2\pi i}{n} k}\right)$ — значение многочлена (5) в точке.

Соответственно, обратное ДПФ $\mathbb{F}^{-1}$ для вектора $\{a_0, a_1, \ldots, a_{n-1}\}$ — это вектор коэффициентов многочлена степени $n-1$, проходящего через точки $\left\{\left(1, a_0\right),\ \left(e^{\frac{2\pi i}{n}}, a_1\right),\ \left(e^{\frac{4\pi i}{n}}, a_2\right), \ldots, \left(e^{\frac{2(n-1)\pi i}{n}}, a_{n-1}\right)\right\}$.

Очевидно, что при умножении двух многочленов их значения в соответствующих точках просто перемножаются, поэтому $\mathbb{F}(P_1(x) \cdot P_2(x)) = \mathbb{F}(P_1(x)) \cdot \mathbb{F}(P_2(x))$, где в правой части подразумевается перемножение соответствующих компонент вектора. Применим к обеим частям $\mathbb{F}^{-1}$ и получим:

$$\mathbb{F}^{-1}\Big(\mathbb{F}(P_1(x) \cdot P_2(x))\Big) = \mathbb{F}^{-1}\Big(\mathbb{F}(P_1(x)) \cdot \mathbb{F}(P_2(x))\Big)$$

или

$$P_1(x) \cdot P_2(x) = \mathbb{F}^{-1}\Big(\mathbb{F}(P_1(x)) \cdot \mathbb{F}(P_2(x))\Big) .$$

То есть, для того, чтобы вычислить произведение многочленов $P_1(x)$ и $P_2(x)$, можно сначала найти Фурье-образы $\mathbb{F}(P_1(x))$ и $\mathbb{F}(P_2(x))$ для каждого из них, затем почленно перемножить (это требует только $n$ операций умножения), и к этому произведению применить $\mathbb{F}^{-1}$.

Вычисление $\mathbb{F}^{-1}$ — это, по сути, задача интерполяции: восстановить коэффициенты многочлена по набору его значений. Обозначим $e^{\frac{2\pi i}{n}k} = e_n^k$ и запишем $\mathbb{F}$ в матричной форме:

$$\begin{pmatrix} e_n^0 & e_n^0 & e_n^0 & \dots & e_n^0 \\ e_n^0 & e_n^1 & e_n^2 & \dots & e_n^{n-1} \\ e_n^0 & e_n^2 & e_n^4 & \dots & e_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_n^0 & e_n^{n-2} & e_n^{2(n-2)} & \dots & e_n^{(n-1)(n-2)} \\ e_n^0 & e_n^{n-1} & e_n^{2(n-1)} & \dots & e_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} \tag{7}$$

Здесь $\{a_0, \dots, a_{n-1}\}$ — вектор коэффициентов $P(x)$, а $\{y_0, \dots, y_{n-1}\}$ — вектор значений данного многочлена в точках $x_k = \exp\left(\frac{2\pi i}{n}k\right)$.

Умножим обе части (7) слева на обратную матрицу, получим:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} e_n^0 & e_n^0 & e_n^0 & \dots & e_n^0 \\ e_n^0 & e_n^1 & e_n^2 & \dots & e_n^{n-1} \\ e_n^0 & e_n^2 & e_n^4 & \dots & e_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_n^0 & e_n^{n-2} & e_n^{2(n-2)} & \dots & e_n^{(n-1)(n-2)} \\ e_n^0 & e_n^{n-1} & e_n^{2(n-1)} & \dots & e_n^{(n-1)(n-1)} \end{pmatrix}^{-1} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} \tag{8}$$

Можно убедиться, что данная обратная матрица имеет следующий вид:

$$\frac{1}{n} \begin{pmatrix} e_n^0 & e_n^0 & e_n^0 & \dots & e_n^0 \\ e_n^0 & e_n^{-1} & e_n^{-2} & \dots & e_n^{-(n-1)} \\ e_n^0 & e_n^{-2} & e_n^{-4} & \dots & e_n^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_n^0 & e_n^{-(n-2)} & e_n^{-2(n-2)} & \dots & e_n^{-(n-1)(n-2)} \\ e_n^0 & e_n^{-(n-1)} & e_n^{-2(n-1)} & \dots & e_n^{-(n-1)(n-1)} \end{pmatrix} \tag{9}$$

Подставляем (9) в (8):

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{pmatrix} = \frac{1}{n} \begin{pmatrix} e_n^0 & e_n^0 & e_n^0 & \dots & e_n^0 \\ e_n^0 & e_n^{-1} & e_n^{-2} & \dots & e_n^{-(n-1)} \\ e_n^0 & e_n^{-2} & e_n^{-4} & \dots & e_n^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_n^0 & e_n^{-(n-2)} & e_n^{-2(n-2)} & \dots & e_n^{-(n-1)(n-2)} \\ e_n^0 & e_n^{-(n-1)} & e_n^{-2(n-1)} & \dots & e_n^{-(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} \tag{10}$$

Отсюда получаем

$$a_k = \frac{1}{n}\sum_{j=0}^{n-1} y_j e_n^{-kj}\,. \tag{11}$$

Если выписать выражение для значений $\mathbb{F}$ $y_k = \sum_{j=0}^{n-1} a_j e_n^{kj}$, то понятно, что эти задачи мало чем отличаются. $\mathbb{F}^{-1}$ можно вычислять точно так же, как и $\mathbb{F}$, только вместо $e^{\frac{2\pi i}{n}k}$ следует использовать $e^{-\frac{2\pi i}{n}k}$, а результат делить на $n$.

Если использовать достаточно быструю процедуру вычисления прямого и обратного $\mathbb{F}$, то мы так же быстро сможем вычислять произведение многочленов.

### 1.3.3 Быстрое преобразование Фурье

Быстрое преобразование Фурье (англ. FFT — Fast Fourier Transform) — это алгоритм для ускоренного вычисления ДПФ. Смысл состоит в следующем: исходный многочлен

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1}x^{n-1}$$

мы разбиваем на два:

$$\begin{aligned} A_0(x) &= a_0 + a_2 x + a_4 x^2 + a_6 x^3 + \ldots + a_{n-2}x^{n/2-1} \\ A_1(x) &= a_1 + a_3 x + a_5 x^2 + a_7 x^3 + \ldots + a_{n-1}x^{n/2-1}\,. \end{aligned}$$

Видно, что

$$A(x) = A_0(x^2) + xA_1(x^2)\,. \tag{12}$$

Если мы за линейное время сможем вычислить $\mathbb{F}(A(x))$ по $\mathbb{F}(A_0(x))$ и $\mathbb{F}(A_1(x))$, то сложность алгоритма в общем составит $O(n \cdot \log_2(n))$.

Пусть

$$\begin{aligned} \mathbb{F}(A_0(x)) = (a_0^0, a_1^0, a_2^0, \ldots, a_{\frac{n}{2}-1}^0) = \\ = \left(A_0\left(e^{\frac{2\pi i}{n/2}\cdot 0}\right), A_0\left(e^{\frac{2\pi i}{n/2}\cdot 1}\right), \ldots, A_0\left(e^{\frac{2\pi i}{n/2}(\frac{n}{2}-1)}\right)\right) = \\ = \left(A_0\left(e^{\frac{4\pi i}{n}\cdot 0}\right), A_0\left(e^{\frac{4\pi i}{n}\cdot 1}\right), \ldots, A_0\left(e^{\frac{4\pi i}{n}(\frac{n}{2}-1)}\right)\right) \end{aligned}$$

$$\mathbb{F}(A_1(x)) = (a_0^1, a_1^1, a_2^1, \ldots, a_{\frac{n}{2}-1}^1) = \\ = \left(A_1\left(e^{\frac{2\pi i}{n/2}\cdot 0}\right), A_1\left(e^{\frac{2\pi i}{n/2}\cdot 1}\right), \ldots, A_1\left(e^{\frac{2\pi i}{n/2}(\frac{n}{2}-1)}\right)\right) = \\ = \left(A_1\left(e^{\frac{4\pi i}{n}\cdot 0}\right), A_1\left(e^{\frac{4\pi i}{n}\cdot 1}\right), \ldots, A_1\left(e^{\frac{4\pi i}{n}(\frac{n}{2}-1)}\right)\right).$$

Найдем $\mathbb{F}(A(x)) = (a_0, a_1, \ldots, a_{n-1})$. Используя (12), для первой половины коэффициентов получим:

$$a_k = A\left(e^{\frac{2\pi i}{n}k}\right) = A_0\left(e^{\frac{2\pi i}{n}2k}\right) + e^{\frac{2\pi i}{n}k} A_1\left(e^{\frac{2\pi i}{n}2k}\right) = \\ = a_k^0 + e^{\frac{2\pi i}{n}k} a_k^1, \qquad k = 0, \ldots, \frac{n}{2} - 1\,. \quad (13)$$

Для второй половины:

$$a_{k+\frac{n}{2}} = A\left(e^{\frac{2\pi i}{n}(k+\frac{n}{2})}\right) = A_0\left(e^{\frac{2\pi i}{n}2(k+\frac{n}{2})}\right) + e^{\frac{2\pi i}{n}(k+\frac{n}{2})} A_1\left(e^{\frac{2\pi i}{n}2(k+\frac{n}{2})}\right) = \\ = A_0\left(e^{\frac{2\pi i}{n}(2k+n)}\right) + e^{\frac{2\pi i}{n}(k+\frac{n}{2})} A_1\left(e^{\frac{2\pi i}{n}(2k+n)}\right) = \\ = A_0\left(e^{\frac{2\pi i}{n}2k} e^{\frac{2\pi i}{n}n}\right) + e^{\frac{2\pi i}{n}k} e^{\frac{2\pi i}{n}\frac{n}{2}} A_1\left(e^{\frac{2\pi i}{n}2k} e^{\frac{2\pi i}{n}n}\right) = \\ = A_0\left(e^{\frac{2\pi i}{n}2k}\right) - e^{\frac{2\pi i}{n}k} A_1\left(e^{\frac{2\pi i}{n}2k}\right) = \\ = a_k^0 - e^{\frac{2\pi i}{n}k} a_k^1, \qquad k = 0, \ldots, \frac{n}{2} - 1\,. \quad (14)$$

Объединяя (13) и (14), получаем:

$$\begin{cases} a_k = a_k^0 + e^{\frac{2\pi i}{n}k} a_k^1\,, \\ a_{k+\frac{n}{2}} = a_k^0 - e^{\frac{2\pi i}{n}k} a_k^1\,, \end{cases}$$

$k = 0, \ldots, \frac{n}{2} - 1$.

# 2 Описание алгоритма

В настоящей главе описан алгоритм сравнения двух символьных последовательностей произвольной длины, не использующий идею выравнивания, и допускающий эффективное распараллеливание.

## 2.1 О преобразовании символьных последовательностей

Пусть мы имеем две конечные последовательности символов из одного и того же алфавита. Для определенности возьмем четырёхбуквенный алфавит $\aleph = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$, с которым будем работать. Исходные последовательности обозначим $\{p_k\}$ и $\{q_k\}$. Они могут быть как одинаковой длины, так и разной. Каждой из исходных последовательностей поставим в соответствие по $|\aleph|$ бинарные последовательности (в нашем случае 4), полученные следующим образом:

- Заменим все символы $\mathsf{A}$ в $\{p_k\}$ на 1, остальные символы на 0. Получим первую бинарную последовательность $\{p_k^{\mathsf{A}}\}$.
- Заменим все символы $\mathsf{C}$ в $\{p_k\}$ на 1, остальные символы на 0. Получим вторую бинарную последовательность $\{p_k^{\mathsf{C}}\}$.
- То же самое проделаем с символами $\mathsf{G}$ и $\mathsf{T}$ в $\{p_k\}$, получим $\{p_k^{\mathsf{G}}\}$ и $\{p_k^{\mathsf{T}}\}$. Аналогично поступим с $\{q_k\}$.

Возьмем бинарные последовательности, соответствующие одному и тому же символу (например, $\{p_k^{\mathsf{A}}\}$ и $\{q_k^{\mathsf{A}}\}$), и вычислим свёртку этих последовательностей. Как было отмечено ранее, каждое значение полученной свёртки – это, по сути, количество пар $\{1, 1\}$ в каждом выравнивании, что в нашем случае означает, что в исходной последовательности на этих местах стоит один и тот же символ. Если мы для каждого символа из алфавита вычислим свёртку пары соответствующих ему бинарных последовательностей, а после просуммируем эти свёртки, то в результате получим супер-свёртку, в которой каждое значение — это количество совпадений по всем символам. Так как мы можем для каждого элемента свёртки однозначно определить, какие «части» исходных последовательностей в это время «накладывались» друг на друга,

мы также можем определить, сколько из этих «наложенных» элементов совпало. Если число совпадений близко к общей длине пересечения, это говорит о том, что данные части последовательностей очень похожи, и различаются только в нескольких символах.

Таким образом, мы можем искать точные или почти точные совпадения в символьных последовательностях путем преобразования их в бинарные и работая по описанному выше алгоритму.

## 2.2 Почему вместо вычисления свёртки последовательностей $\{a_0, \ldots, a_n\}$ и $\{b_0, \ldots, b_k\}$ можно перемножать многочлены с коэффициентами $\{a_0, \ldots, a_n\}$ и $\{b_k, \ldots, b_0\}$?

Ещё раз коротко рассмотрим несколько первых шагов вычисления свёртки. Наша цель — вычислить свёртку последовательностей $\{a_i\}_{i=0}^{n}$ и $\{b_j\}_{j=k}^{0}$, так как именно в этом случае последовательности «пройдут» одна над другой в естественном порядке, и значения свёртки будут отражать количество совпадений элементов последовательностей в каждом положении. Сначала исходные последовательности пересекаются по одному элементу следующим образом:

$$\begin{array}{cccccc} & & & a_0 & a_1 & \ldots & a_n \\ b_0 & \ldots & b_{k-1} & b_k & & & \end{array}$$

Получим первый элемент свёртки $c_0 = a_0 b_k$. После мы сдвигаем нижнюю последовательность на один элемент вправо, получим:

$$\begin{array}{cccccc} & & a_0 & a_1 & \ldots & a_n \\ b_0 & \ldots & b_{k-2} & b_{k-1} & b_k & \end{array}$$

Тогда второй элемент свёртки $c_1 = a_0 b_{k-1} + a_1 b_k$. Аналогичным образом вычисляется $c_2 = a_0 b_{k-2} + a_1 b_{k-1} + a_2 b_k$, и так далее. Всего получится $n + k + 1$ элемент, так как $b_k$ сначала «пройдет» все $a_i$, $i = 0, \ldots, n$, а затем ещё $k$ шагов до тех пор, пока $b_0$ не дойдёт до $a_n$.

Теперь рассмотрим, как перемножаются многочлены

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

и

$$B(x) = b_k + b_{k-1}x + b_{k-2}x^2 + \ldots + b_0x^k.$$

В результате получится многочлен

$$C(x) = c_0 + c_1x + \ldots + c_{n+k}x^{n+k}$$

степени $n + k$, то есть имеющий $n + k + 1$ коэффициент. Свободный коэффициент $C(x)$ является произведением свободных членов $A(x)$ и $B(x)$, то есть $c_0 = a_0b_k$.

Коэффициент при первой степени $x$ получается как $c_1 = a_1b_k + a_0b_{k-1}$. И так далее. То есть, вместо вычисления свёртки напрямую путем «накладывания» одной последовательности на другую и вычисления для каждого такого наложения нескольких операций умножения и сложения можно перемножить многочлены с соответствующими коэффициентами, предварительно инвертировав одну последовательность. Если мы сможем достаточно быстро перемножить эти многочлены, то получим существенный прирост в скорости вычисления. Для того, чтобы быстро перемножать многочлены, можно использовать Быстрое Преобразование Фурье, или БПФ.

## 2.3 Способ вычисления свёртки

На вход получаем две последовательности $\mathfrak{N}$ и $\mathfrak{L}$ символов конечного алфавита $\aleph$ длины $N$ и $L$ соответственно. Первую последовательность будем называть «главной», а вторую — «шаблоном». Каждую из исходных последовательностей преобразуем в $|\aleph|$ бинарных по правилу, описанному в п. 2.1. Бинарные последовательности, полученные из $\mathfrak{N}$, обозначим $\mathfrak{N}_\alpha,\ \alpha \in \aleph$, а полученные из $\mathfrak{L}$ — $\mathfrak{L}_\alpha,\ \alpha \in \aleph$. Для дальнейших вычислений $\mathfrak{L}_\alpha$ для всех $\alpha$ необходимо инвертировать, т. е. последовательность $a_0\, a_1\, \ldots\, a_{n-1}\, a_n$ преобразовать в $a_n\, a_{n-1}\, \ldots\, a_1\, a_0$.

После вычисления свёртки с помощью прямого и обратного БПФ получается последовательность длины $N + L - 1$, поэтому все $2 \times |\aleph|$ бинарные последовательности необходимо дополнить справа нулями до этой длины. Кроме этого, поскольку быстрое преобразование Фурье наиболее эффективно работает при длине, равной точной степени двойки, необходимо также

дополнить последовательности нулями до длины $\widetilde{N} = 2^{\lceil \log_2(N+L-1) \rceil}$.

Вычисляем прямое БПФ для каждого $\mathfrak{N}_\alpha$ и $\mathfrak{L}_\alpha$. Затем для каждой пары преобразованных последовательностей, соответствующих одному и тому же символу $\alpha$, вычисляем произведение соответствующих (т. е. имеющих одинаковые номера) элементов. Эта процедура повторяется для всех $\alpha$. В результате этих операций мы получаем $|\aleph|$ векторов, элементами которых являются указанные выше попарные произведения. Затем эти $|\aleph|$ векторов складываются как векторы.

Рассмотрим последний шаг более подробно на примере: пусть $\aleph = \{\alpha_1, \alpha_2\}$, $|\aleph| = 2$. Положим $\mathfrak{N} = \alpha_2\,\alpha_1$, $\mathfrak{L} = \alpha_1\,\alpha_1$. В этом случае бинарные последовательности будут следующие: $\mathfrak{N}_{\alpha_1} = (0,1,0,0)$, $\mathfrak{N}_{\alpha_2} = (0,1,0,0)$, $\mathfrak{L}_{\alpha_1} = (1,1,0,0)$, $\mathfrak{L}_{\alpha_2} = (0,0,0,0)$.

Пусть

$$\widehat{\mathfrak{N}}_{\alpha_1} = \mathbb{F}(\mathfrak{N}_{\alpha_1}) = (a_1^1, a_2^1, a_3^1, a_4^1), \qquad \hat{\mathfrak{L}}_{\alpha_1} = \mathbb{F}(\mathfrak{L}_{\alpha_1}) = (b_1^1, b_2^1, b_3^1, b_4^1),$$
$$\widehat{\mathfrak{N}}_{\alpha_2} = \mathbb{F}(\mathfrak{N}_{\alpha_2}) = (a_1^2, a_2^2, a_3^2, a_4^2), \qquad \hat{\mathfrak{L}}_{\alpha_2} = \mathbb{F}(\mathfrak{L}_{\alpha_2}) = (b_1^2, b_2^2, b_3^2, b_4^2)\,,$$

где $\mathbb{F}$ — быстрое преобразование Фурье. Поэлементно перемножив $\widehat{\mathfrak{N}}_{\alpha_1}$ и $\hat{\mathfrak{L}}_{\alpha_1}$, получим

$$(a_1^1\,b_1^1,\ a_2^1\,b_2^1,\ a_3^1\,b_3^1,\ a_4^1\,b_4^1)\,.$$

Перемножив $\widehat{\mathfrak{N}}_{\alpha_2}$ и $\hat{\mathfrak{L}}_{\alpha_2}$, получим

$$(a_1^2\,b_1^2,\ a_2^2\,b_2^2,\ a_3^2\,b_3^2,\ a_4^2\,b_4^2)\,.$$

В результате сложения данных произведений получится последовательность

$$(a_1^1\,b_1^1 + a_1^2\,b_1^2,\ a_2^1\,b_2^1 + a_2^2\,b_2^2,\ a_3^1\,b_3^1 + a_3^2, b_3^2,\ a_4^1\,b_4^1 + a_4^2\,b_4^2)\,.$$

Применив к данной последовательности обратное БПФ, мы получим требуемую свёртку.

Стоит отметить, что, несмотря на то, что в процессе вычисления прямого быстрого преобразования Фурье мы переходили к комплексным числам, после применения обратного БПФ мнимые части всех элементов станут равными нулю. Более того, занулятся все элементы последовательности, начиная с $N + L$, так как в результате всех преобразований мы получим то же, что получили бы, вычисляя свёртку напрямую — путём умножения и сложения элементов последовательностей.

# 3 Вычислительные эксперименты

## 3.1 Выбор тестовых последовательностей

Проверку работоспособности метода мы проводили на наборе символьных последовательностей, среди которых были:

– случайные нескоррелированные, порожденные с помощью реализации случайного процесса Бернулли; относительные частоты появления каждого символа назначались различным образом, в том числе следую правилам Чаргаффа (частота символа A с высокой точностью равна частоте T, частота G — частоте C);

– случайные нескоррелированные, специальным образом изменённые путем искусственного внедрения одинаковых участков, одиночных и множественных повторов различной длины;

– реальные нуклеотидные последовательности на примере геномов хлоропласта (характерная длина $\approx 1{,}5 \cdot 10^5$ символов), которые, как известно, довольно сильно отличаются от случайных;

– реальные нуклеотидные последовательности, также изменённые путем добавления одинаковых участков.

В некоторых случаях во вставленные участки последовательностей были внесены небольшие изменения — вставки, удаления, замены отдельных символов.

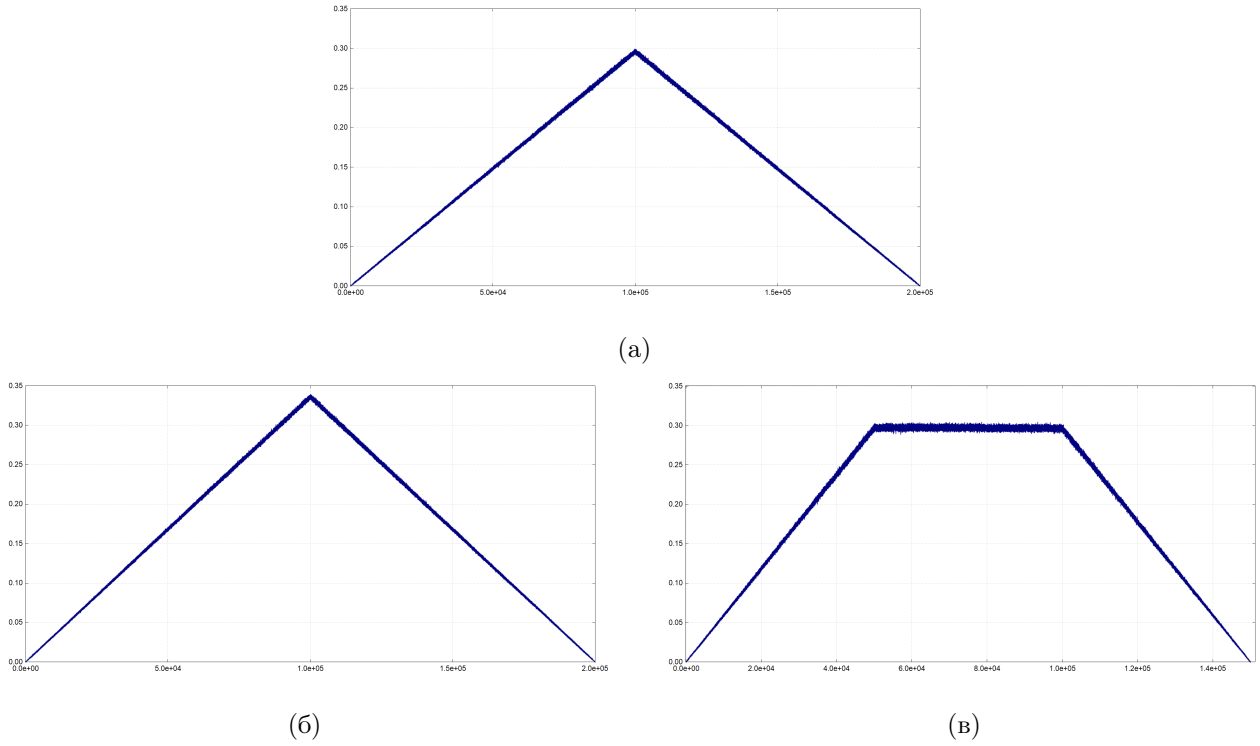
## 3.2 Результаты экспериментов со случайными последовательностями

### 3.2.1 Последовательности, не имеющие совпадающих участков

В данном эксперименте были сгенерированы две случайные нескоррелированные последовательности, обе длины $10^5$ символов, которые затем сравнивались по методу, описанному в п. 2.3. В первом случае частота встречи символа A (обозн. $p_{\mathsf{A}}$) приближённо равна 0,1, аналогично $p_{\mathsf{C}} \approx 0{,}2$, $p_{\mathsf{G}} \approx 0{,}3$, $p_{\mathsf{T}} \approx 0{,}4$. Результат представлен на Рис. 1(а). Здесь и всюду далее на рисунках по оси абсцисс указан порядковый номер элемента свёртки, по оси

ординат — значение свёртки, делённое на наименьшую из длин сравниваемых последовательностей.

Для второго примера снова были сгенерированы две случайные нескоррелированные последовательности длины $10^5$ символов, однако частоты различных символов выбирались в соответствии с правилом Чаргаффа, а именно $p_{\mathsf{A}} \approx p_{\mathsf{T}} \approx 0{,}1$, $p_{\mathsf{G}} \approx p_{\mathsf{C}} \approx 0{,}4$. Результат этого вычислительного эксперимента представлен на Рис. 1(б).



(а)

(б)

(в)

Рисунок 1 – Стандартные графики свёртки в случае, когда последовательности: (а) одинаковой длины; (б) одинаковой длины, частоты распределены в соотв. с правилом Чаргаффа; (в) разной длины.

На рисунках 1(а) и 1(б) представлены типичные результаты, получаемые при сравнении различных последовательностей, которые не имеют общих участков, но при этом имеют одинаковую длину. Если же длины сравниваемых последовательностей разные, то картина подобна той, что представлена на Рис. 1(в). Во всех случаях общая длина свёртки есть сумма длин сравниваемых последовательностей.

В первой половине графиков значение свёрточной функции возрастает потому, что увеличивается длина пересечения (или длина общей части каждой последующей укладки), следовательно возрастает количество слу-

чайных совпадений одиночных символов, которые также вносят вклад в значение свёртки. Это явление (случайное совпадение одного символа, реже двоек, троек и т. д.) мы называем «шум», о нем будет сказано подробнее ниже, см. п. 3.5.

Соответственно, во второй половине графиков значение свёртки убывает по обратной причине — длина пересечения уменьшается, следовательно, уменьшается количество случайных совпадений.

### 3.2.2 Последовательности с искусственно внедрёнными совпадениями

Для следующего эксперимента мы взяли две случайные нескоррелированные последовательности ($p_{\mathsf{A}} \approx 0{,}1,\ p_{\mathsf{C}} \approx 0{,}2,\ p_{\mathsf{G}} \approx 0{,}3,\ p_{\mathsf{T}} \approx 0{,}4$) длины $10^5$ символов. Для проведения эксперимента из первой последовательности был скопирован участок длины $2 \cdot 10^3$ символов и внедрен (вставлен вместо другого такой же длины) во вторую последовательность. Вообще говоря, имеет значение то место, откуда был скопирован этот участок, и куда вставлен. Чем ближе он находится к серединам последовательностей (и ближе к центру свёрточной функции), тем сложнее задача его локализации. Подробнее проблема локализации будет рассматриваться в п. 3.5. В данном примере расположение этого участка в исходных последовательностях значения не имеет.

Результат вычисления свёртки представлен на Рис. 2. Отчетливый «скачок» слева свидетельствует о наличии в исследуемых последовательностях совпадающего участка (того самого, который мы вставили). Высота этого «скачка» напрямую зависит от длины совпадающей части.

Ещё одним вариантом указанного выше эксперимента было сравнение последовательности с самой собой, так как такая задача также важна в прикладных исследованиях. Результат этого эксперимента представлен на Рис.3. Большой «скачок» по центру возникает потому, что в этот момент вся последовательность совпала сама с собой. Именно поэтому значение свёртки в этом месте равно единице (или, что то же самое, совпало 100 % от длины последовательности).

Для следующего примера в качестве первой последовательности мы взяли случайную последовательность длины $10^5$ символов, а в качестве вто-

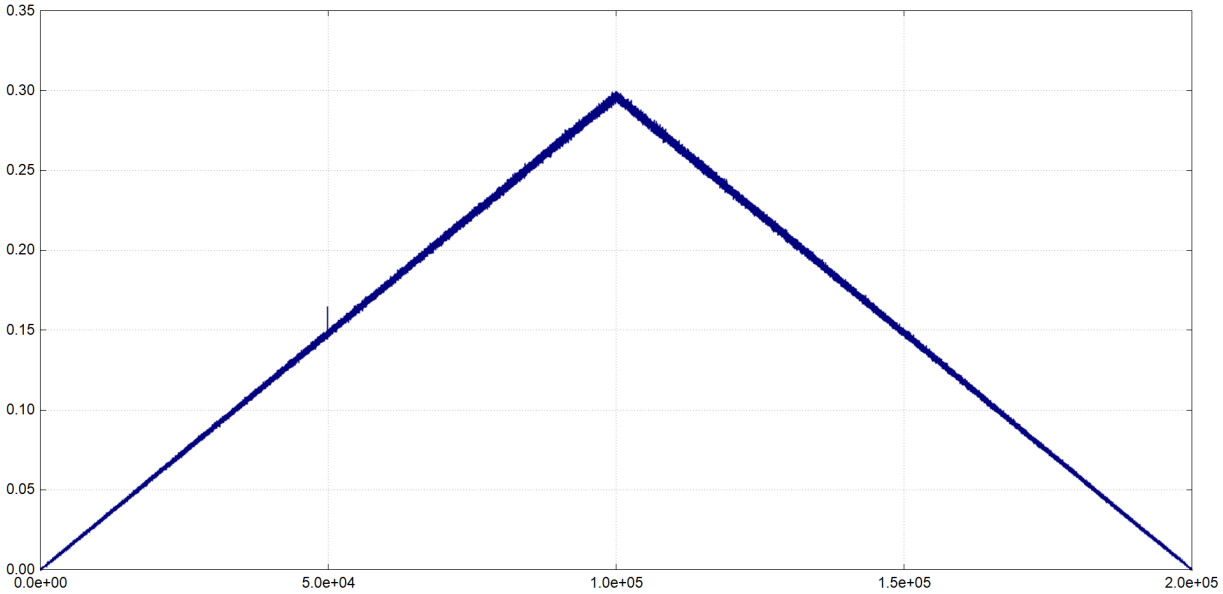



Рисунок 2 – Пример поиска точного совпадения длиной 2000 символов в двух разных последовательностях длиной $10^5$ каждая. Пик значения свёртки четко указывает на это совпадение.

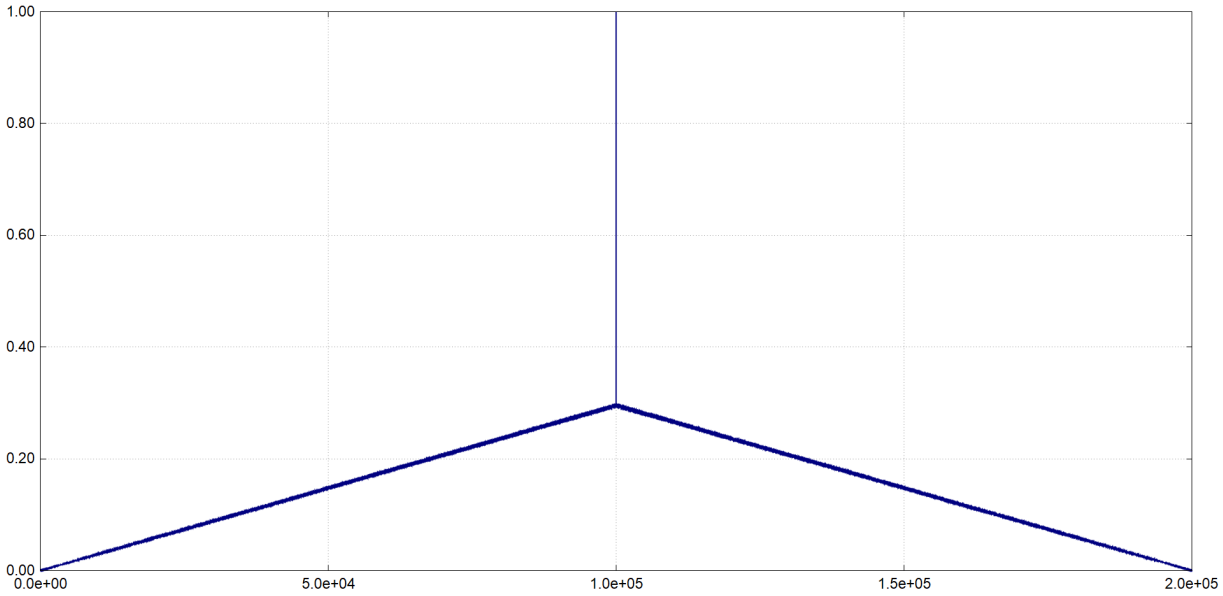



Рисунок 3 – Пример сравнения случайной последовательности с самой собой.

рой — участок первой последовательности длины $10^4$ символов. Результат представлен на Рис.4. Он имеет много общего с тем результатом, который мы получили в случае Рис. 1(в), так как в этом эксперименте исследуемые последовательности также были различной длины.

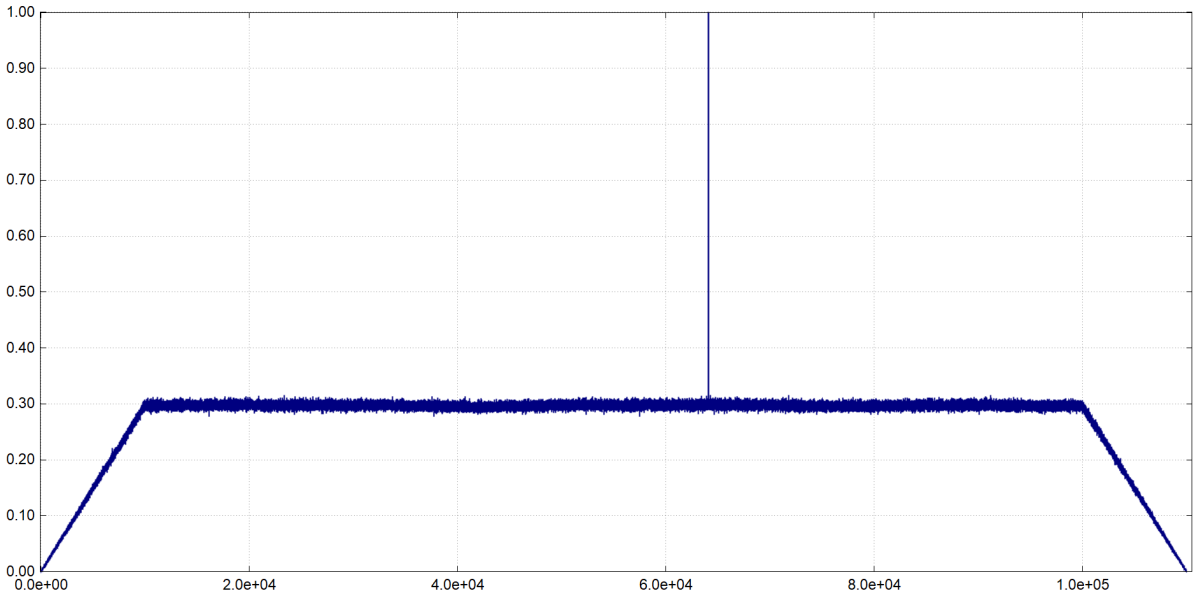



Рисунок 4 – Пример сравнения случайной последовательности длины $10^5$ символов с её участком длины $10^4$ символов.

### 3.3 Результаты экспериментов с реальными последовательностями

Для тестирования метода на реальных генетических последовательностях мы выбрали геном хлоропласта одного из видов орхидей рода Фаленопсис (*Phalaenopsis equestris*). Для первого эксперимента мы взяли целый геном длины $\approx 1{,}5 \cdot 10^5$ символов в качестве первой последовательности, и его участок длины $10^4$ символов в качестве второй последовательности. Результат представлен на Рис.5.

Полученная свёртка похожа на ту, что мы получили в случае Рис. 4, за исключением некоторых отличий. Во-первых, из-за реальной природы данной последовательности в ней присутствует большое количество участков, похожих между собой, из-за чего значения в центральной части свёртки несколько больше. Во-вторых, структура последовательности неоднородна — частоты появления одних и тех же символов в различных её участках и в последовательности в общем отличаются, — поэтому график свёрточной функции в центральной части не монотонен, в отличие от Рис. 4.

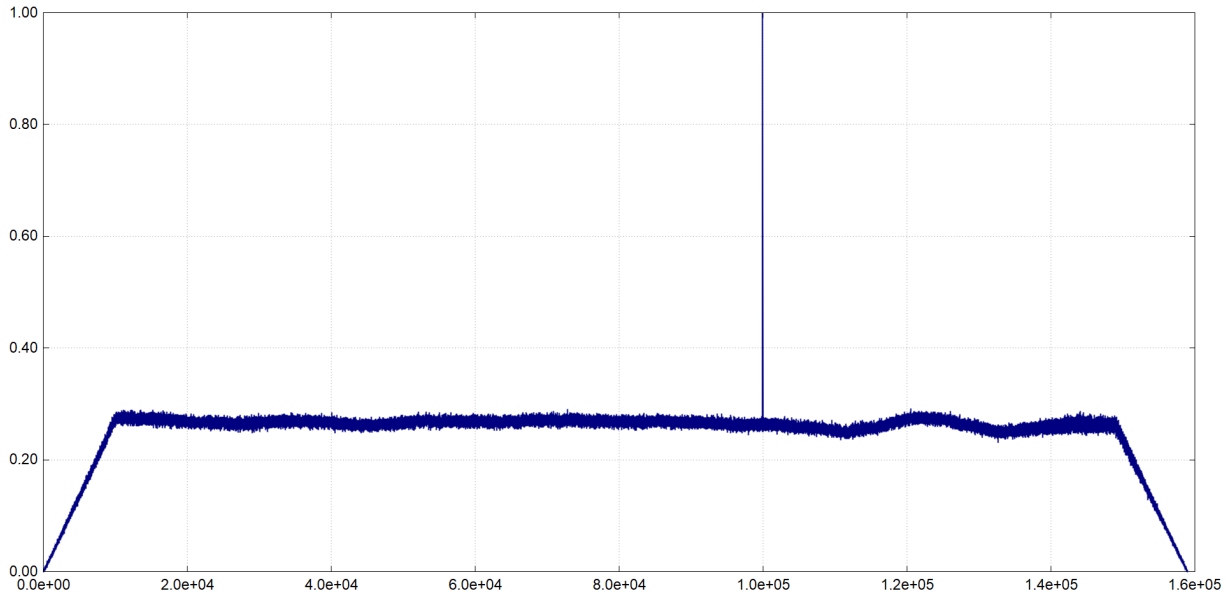


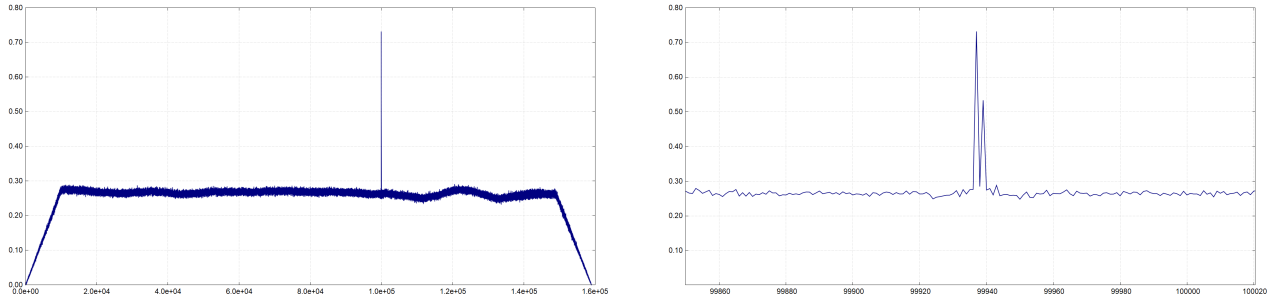

Рисунок 5 – Пример сравнения генома хлоропласта длины $\approx 1{,}5 \cdot 10^5$ символов с его участком длины $10^4$ символов.

## 3.4 Последовательности с неточными совпадениями

До сих пор мы работали только с последовательностями, в которых искомые совпадения были абсолютно точными. Однако, наибольшее прикладное значение имеет задача поиска именно неточных совпадений, другими словами, в которых мы допускаем наличие вставок, выпадений и замен отдельных символов или блоков символов. Стоит отметить, что именно эта задача в настоящее время является труднорешаемой, в особенности для инструментов, работающих на основе «выравнивания», таких как BLAST и др.

Напротив, разработанный нами метод весьма хорошо себя проявляет при идентификации такого рода несоответствий. Это можно заметить на следующем примере: в качестве исходных мы взяли те же самые последовательности, что и в первом примере п. 3.4 — целую последовательность длины $\approx 1{,}5 \cdot 10^5$ символов, и её участок длины $10^4$ символов, а затем во вторую последовательность внедрили вставку длины 2 символа. Результат представлен на Рис. 6.

Можно заметить, что исходный «пик» разделился на два, а расстояние между ними равно в точности длине вставки. Если бы мы добавили ещё одну вставку во вторую последовательность, то получили бы три «пика» вместо двух.



(а) (б)

Рисунок 6 – Результат эксперимента с несовпадением типа вставки/исключения: (а) свёртка целиком; (б) пик в увеличенном масштабе.

## 3.5 Проблема локализации

Как уже было сказано выше, значение свёрточной функции для каждой укладки — это в точности число совпавших в ней символов. Но помимо действительно значащих совпадений идущих подряд 10-20-30 и т. д. элементов, в него также достаточно большой вклад вносят совпадения одиночных символов, которые, с одной стороны, для нас никакого практического значения не имеют, а с другой, только «заглушают» сигнал от значащих совпадений, которые теряются на фоне шума. Это делает невозможным идентификацию наличия данных совпадений, и, как следствие, их локализацию.

Для того, чтобы избавиться от этого «шума», мы поступаем следующим образом. Для каждой из исследуемых последовательностей мы считаем частоту появления каждого символа (в нашем случае это A, C, G и T; соответствующие им частоты равны $p_{\mathsf{A}}$, $p_{\mathsf{C}}$, $p_{\mathsf{G}}$ и $p_{\mathsf{T}}$), а затем считаем коэффициент $\rho$ по следующей формуле:

$$\rho = p^1_{\mathsf{A}}\, p^2_{\mathsf{A}} + p^1_{\mathsf{C}}\, p^2_{\mathsf{C}} + p^1_{\mathsf{G}}\, p^2_{\mathsf{G}} + p^1_{\mathsf{T}}\, p^2_{\mathsf{T}}\,, \tag{15}$$

где $p^1_\alpha$ соответствует частотам символов первой последовательности, а $p^2_\alpha$ — символов второй последовательности. В общем смысле, $\rho$ — это вероятность того, что в двух случайных последовательностях при их укладке на одном и том же месте два символа совпадут. Если мы умножим $\rho$ на длину укладки, то получим среднее количество тех самых случайных совпадений.

Чтобы вычислить настоящую длину совпадающей части $l^*$, воспользу-

емся следующей формулой:

$$l^* = \frac{B - \rho M}{1 - \rho}\,, \tag{16}$$

где $B$ — ненормированное (исходное) значение свёртки, $M$ — длина укладки (она легко вычисляется, зная лишь длины сравниваемых последовательностей).

Рассмотрим более подробно на примере: возьмем две случайно сгенерированные последовательности, длины $10^5$ и $8 \cdot 10^4$ символов соответственно. Вероятности появления каждого символа возьмем одинаковыми в каждой последовательности — $p_{\mathsf{A}} \approx 0{,}1$, $p_{\mathsf{C}} \approx 0{,}3$, $p_{\mathsf{G}} \approx 0{,}4$, $p_{\mathsf{T}} \approx 0{,}2$. Тогда $\rho = p_{\mathsf{A}}^2 + p_{\mathsf{C}}^2 + p_{\mathsf{G}}^2 + p_{\mathsf{T}}^2$. Из первой последовательности возьмем участок длины $10^4$ символов и вставим с заменой во вторую последовательность. Результат вычисления свёртки представлен на Рис. 7.

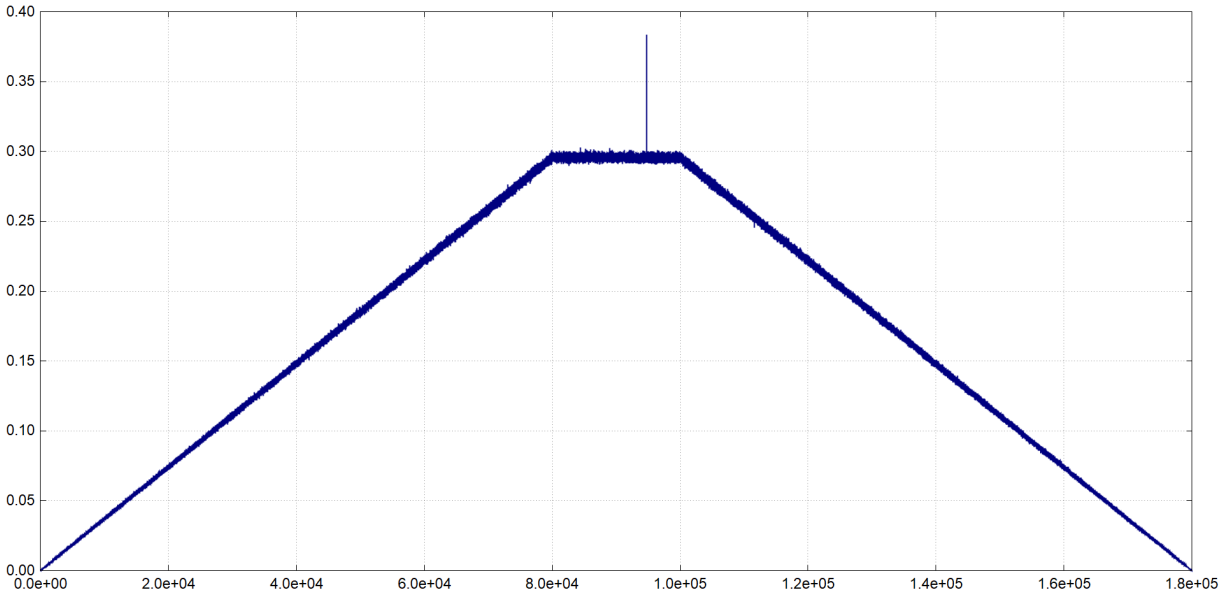



Рисунок 7 – Сравнение двух случайных последовательностей длины $10^5$ и $8 \cdot 10^4$ символов, в которых присутствует общая часть длины $10^4$ символов.

Для данного случая мы имеем $\rho = 0{,}29567265$, $B = 0{,}3835249 \cdot 8 \cdot 10^4 = 3{,}0681992 \cdot 10^4$, $M = 8 \cdot 10^4$. Воспользовавшись формулой (16), получим

$$l^* = \frac{3{,}0681992 \cdot 10^4 - 0{,}29567265 \cdot 8 \cdot 10^4}{1 - 0{,}29567265} = 9\,978{,}57\,.$$

Можно заметить, что это довольно близко к искомым 10 000. Более того, чем меньше длина сравниваемых последовательностей (хотя бы одной из них),

тем больше точность вычисления. В данном случае, относительная погрешность составляет 0,214 %.

## 3.6 Обсуждение

Обсуждение полученных результатов начнём с анализа влияния ошибок округления на результаты вычисления быстрого преобразования Фурье. Мы используем БПФ для нахождения произведения многочленов, причем не просто с целыми, а с $(0, 1)$ коэффициентами. Следовательно, все коэффициенты многочлена, являющегося их произведением, также будут являться целыми числами, независимо от того, вычисляли мы их напрямую, или с помощью прямого и обратного БПФ. В процессе вычисления БПФ мы переходим к комплексным числам, которые представляют собой пару действительных, и все операции с комплексными числами сводятся к операциям с действительными. Для хранения действительных чисел используется тип `double`. На практике, при вычислении БПФ в процессе выполнения большого числа арифметических операций накапливается некоторая ошибка округления, которая не выходит за пределы 9 знаков после запятой. На данном этапе работы такого количества достаточно для того, чтобы не учитывать данные ошибки.

Перейдём теперь к обсуждению содержательной стороны представленного метода. Как уже было отмечено выше, большинство методов и программ, основывающихся на методе выравнивания (BLAST и ему подобные), обладают существенным недостатком — эти методы являются крайне неэффективными в случае необходимости распознать неточность в виде вставок и/или выпадений. Непосредственное применение любой из версий методов, основанных на выравнивании, ошибки типа вставок и/или выпадений приводят к потере обширных совпадающих фрагментов. Имеющиеся в настоящее время реализации таких методов, используемых при поиске несовпадений типа вставок и/или выпадений, являются высокозатратными алгоритмами (и их программными реализациями) и обладают низким уровнем общности: как правило, такие алгоритмы хорошо срабатывают на вполне определенном классе последовательностей и плохо — на других. Основным достоинством метода сравнения символьных последовательностей, представленного в настоящей

работе, является отсутствие указанного здесь недостатка.

Предварительные результаты проведённых вычислительных экспериментов показывают, что рассмотренный метод сравнения символьных последовательностей, основанный на вычислении свёрточных функций по специально преобразованным последовательностям, обладает высокой эффективностью и отсутствием ряда ключевых недостатков, характерных для методов, основывающихся на выравнивании. Тем не менее, предварительные вычислительные эксперименты выявили ряд вопросов, требующих дальнейших исследований.

Одной из основных проблем является проблема локализации. Даже в тех случаях, когда мы можем сказать, что в исследуемых последовательностях однозначно имеется совпадающий участок, его местоположение в этих последовательностях описанным выше методом определяется с трудом. В различных ситуациях мы можем лишь сузить зону поиска, которая, впрочем, всё ещё остаётся во много раз длиннее самого совпадения. Одним из возможных подходов к решению этой проблемы является разработанный нами метод, основанный на бисекции исходных последовательностей. Кроме этого, для упрощения задачи локализации использовался следующий метод: мы «нарезали» одну из последовательностей на сравнительно малые части (длиной $10^3 - 10^4$ символов, в зависимости от длин исходных последовательностей), и сравнивали каждую такую часть отдельно со второй последовательностью. Так как влияние шума и сложность локализации напрямую зависят от соотношения длины совпадения с наименьшей из длин последовательностей, во многих случаях это помогало обнаружить те совпадения, которые были не видны в бóльшем масштабе, а также локализовать их местоположение.

Одним из способов решения задачи локализации и выделения значащего сигнала была борьба с короткомерным шумом. Для этого мы поступали следующим образом: от каждого значения свёртки отнимали произведение коэффициента $\rho$, рассчитанного по формуле (15) (см. стр. 27), на длину соответствующей укладки. Другими словами, мы отнимаем некое подобие математического ожидания данного значения, или ожидаемое количество случайных совпадений отдельных символов. Таким образом, те значения свёртки,

которые стали отрицательными после данного преобразования, почти наверняка не содержат значащих совпадений. Те же значения, которые стали выделяться, напротив, могут оказаться значащими. Пример результата данного преобразования представлен на Рис. 8. Здесь были взяты такие же последо-

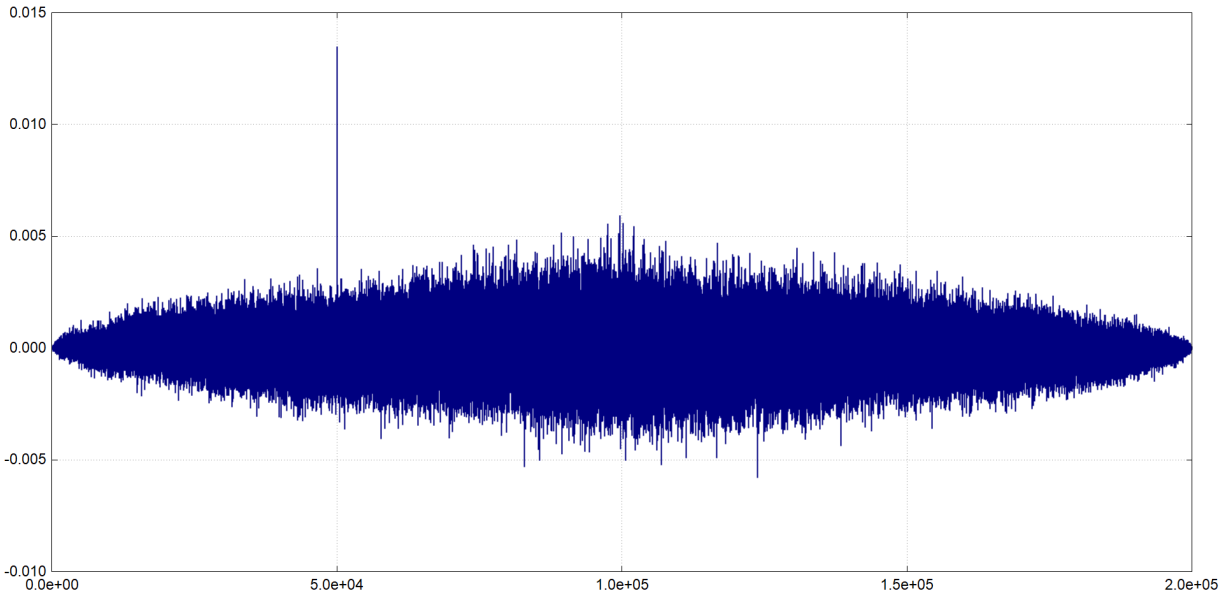



Рисунок 8 – Пример преобразованной свёртки

вательности, что и в примере Рис. 2 (обе длины $10^5$ символов с общей частью $2 \cdot 10^3$ символов), и к вычисленной свёртке применено описанное выше преобразование. Можно заметить, что пик, который раньше выделялся не столь сильно, стал намного более ярко выраженным.

Другим подходом может послужить использование скользящего среднего [30]. Анализируемая последовательность заменяется на другую, в которой каждое значение заменяется на среднее, определяемое по некоторому интервалу. Длина интервала является свободным параметром и выбирается исходя из потребностей задачи. Для выделения сигнала следует определить стандартное отклонение по интервалу, и сигналом считать те значения в исходной последовательности, которые выходят за пределы данного стандартного отклонения.

Кроме этого, содержательной проблемой также является визуализация полученных результатов. Все последовательности, которые мы сравнивали между собой в данной работе, имели небольшую длину (по сравнению с теми, с которыми мы будем работать), поэтому на данном этапе трудностей не

возникало. Более того, на данный момент мы в состоянии более-менее приемлемо представить результаты сравнения последовательностей длины порядка $10^6$ символов. Однако, с увеличением длины на порядок те средства, которые мы используем, перестают быть достаточно эффективными — «отрисовка» свёртки занимает время бóльшее, чем её вычисление, а также существенно усложняется её визуальная обработка.

Ближайшим направлением развития работы является разработка версии метода, использующей крупномодульный параллелизм. Действительно, характерная длина генетических последовательностей составляет $10^7 \div 10^8$ символов; характерное число последовательностей, которые требуется проанализировать одновременно, составляет $10^3 \div 10^4$ единиц. Очевидно, что для таких больших объёмов данных обойтись без использования алгоритмов распараллеливания не удастся.

В данной работе рассматривались неточности типа замена/вставка/исключение. Однако этим возможные типы неточностей не исчерпываются. Нуклеотидные последовательности могут иметь так называемые «блочные перестановки», то есть, в одной последовательности две подстроки символов идут в одном порядке, а в другом они поменялись местами. Кроме этого, в пределах одной и той же последовательности могут встречаться инвертированные участки самой себя, например, когда в одной части последовательности есть одна подпоследовательность символов, а в другой части этой же самой последовательности данная подпоследовательность направлена в обратном порядке. Анализ неточностей этого типа не входил в задачи данной работы.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы все поставленные задачи были выполнены, а цель достигнута.

1. Разработан метод сравнения символьных последовательностей, основанный на вычислении свёртки двух полиномов, построенных на базе данных последовательностей, и использовании быстрого преобразования Фурье.

2. Написана компьютерная программа, реализующая данный метод, проведено её тестирование и отладка на большом классе входных последовательностей.

3. Проведены серии вычислительных экспериментов на специально сгенерированных последовательностях с целью определения пределов применимости разработанного метода и его алгоритмической реализации.

4. Проведены серии вычислительных экспериментов на реальных генетических последовательностях для сравнения с другими распространёнными инструментами, использующимися для сравнения и поиска в последовательностях.

**Апробация работы**. Основные результаты работы докладывались на следующих конференциях и семинарах:

1) XXVIII Всероссийском семинаре «Нейроинформатика, её приложения и анализ данных», Красноярск, 3 – 5 октября 2019 г.;
2) Международной молодёжной научной конференции «Проспект Свободный — 2020», Красноярск, 24 апреля 2020 г.;
3) 1-ом Сибирском научном семинаре по технологиям анализа данных и приложениям (The 1st Siberian Scientific Workshop on Data Analysis Technologies with Applications, SibDATA-2020), Красноярск, 10 июня 2020 г;
4) 8-ой Международной рабочей конференции по биоинформатике и биоинженерии (8th International Work-Conference on Bioinformatics and Biomedical Engineering, IWBBIO 2020), Гранада (Испания).

Основные результаты выпускной квалификационной работы опубликованы в:

1) A. Molyavko, V. Shaidurov, Eu. Karepova, M. Sadovsky Highly Parallel Convolution Method to Compare DNA Sequences with Enforced In/Del and Mutation Tolerance // LNBI, 2020, vol.12108, pp.472 – 481;
2) В. В. Шайдуров, А. А. Молявко, А. А. Поплаухин, А. И. Аксенова, М. Г. Садовский. Об одном алгоритме поиска общих подпоследовательностей с неточным совпадением (включая вставки и выпадения) на основе свёрточных функций и быстрого преобразования Фурье // Материалы XXVIII Всероссийского семинара *Нейроинформатика, её приложения и анализ данных*, Красноярск, 2019, с.138 – 144.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


[1] *Hintzsche, J. D.* A survey of computational tools to analyze and interpret whole exome sequencing data / J. D. Hintzsche, W. A. Robinson, A. C. Tan // *International journal of genomics.* — 2016. — Vol. 2016.

[2] *Hetland, M. L.* A survey of recent methods for efficient retrieval of similar time sequences / M. L. Hetland // Data mining in time series databases. — World Scientific, 2004. — Pp. 23–42.

[3] *Левенштейн, В. И.* О совершенных кодах в метрике выпадений и вставок / В. И. Левенштейн // *Дискретная математика.* — 1991. — Vol. 3, no. 1. — Pp. 3–20.

[4] *Levenshtein, V. I.* Efficient reconstruction of sequences from their subsequences or supersequences / V. I. Levenshtein // *Journal of Combinatorial Theory, Series A.* — 2001. — Vol. 93, no. 2. — Pp. 310–332.

[5] *Levenshtein, V. I.* Bounds for deletion/insertion correcting codes / V. I. Levenshtein // Proceedings IEEE International Symposium on Information Theory / IEEE. — 2002. — P. 370.

[6] *Ng, H.-C.* Reconfigurable acceleration of genetic sequence alignment: A Survey of two decades of efforts / H.-C. Ng, S. Liu, W. Luk // 2017 27th International Conference on Field Programmable Logic and Applications (FPL) / IEEE. — 2017. — Pp. 1–8.

[7] *Ning, K.* Finding patterns in biological sequences by longest common subsequences and shortest common supersequences / K. Ning, H. K. Ng, H. W. Leong // Sixth IEEE Symposium on BioInformatics and BioEngineering (BIBE'06) / IEEE. — 2006. — Pp. 53–60.

[8] *Iliopoulos, C. S.* Pattern matching algorithms with don't cares / C. S. Iliopoulos, M. S. Rahman // Proc. 33rd SOFSEM. — 2007. — Pp. 116–126.

[9] Alignment-free sequence comparison: benefits, applications, and tools / A. Zielezinski, S. Vinga, J. Almeida, W. M. Karlowski // *Genome biology.* — 2017. — Vol. 18, no. 1. — P. 186.

[10] A Survey of Methods and Tools for Large-Scale DNA Mixture Profiling / E. Alamoudi, R. Mehmood, A. Albeshri, T. Gojobori // Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies. — Cham: Springer International Publishing, 2020. — Pp. 217–248.

[11] Basic local alignment search tool / S. F. Altschul, W. Gish, W. Miller et al. // *Journal of molecular biology*. — 1990. — Vol. 215, no. 3. — Pp. 403–410.

[12] *Левенштейн, В. И.* Двоичные коды с исправлением выпадений, вставок и замещений символов / В. И. Левенштейн // *ДАН СССР*. — 1965. — Vol. 163, no. 4. — Pp. 845–848.

[13] *Miller, F.P.* Levenshtein Distance / F.P. Miller, A.F. Vandome, J. McBrewster. — VDM Publishing, 2009.

[14] *Marteau, P.-F.* On recursive edit distance kernels with application to time series classification / P.-F. Marteau, S. Gibet // *IEEE transactions on neural networks and learning systems*. — 2015. — Vol. 26, no. 6. — Pp. 1121–1133.

[15] Efficient genome-wide, privacy-preserving similar patient query based on private edit distance / X. S. Wang, Y. Huang, Y. Zhao et al. // Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security / ACM. — 2015. — Pp. 492–503.

[16] *Tarhio, J.* Technology beats algorithms (in exact string matching) / J. Tarhio, J. Holub, E. Giaquinta // *Software: Practice and Experience*. — 2017. — Vol. 47, no. 12. — Pp. 1877–1885.

[17] *Chakraborty, D.* Streaming algorithms for embedding and computing edit distance in the low distance regime / D. Chakraborty, E. Goldenberg, M. Koucký // Proceedings of the forty-eighth annual ACM symposium on Theory of Computing / ACM. — 2016. — Pp. 712–725.

[18] *Saha, S.* ERGC: an efficient referential genome compression algorithm / S. Saha, S. Rajasekaran // *Bioinformatics*. — 2015. — Vol. 31, no. 21. — Pp. 3468–3475.

[19] *Al Kawam, A.* A survey of software and hardware approaches to performing read alignment in next generation sequencing / A. Al Kawam, S. Khatri, A. Datta // *IEEE/ACM transactions on computational biology and bioinformatics.* — 2017. — Vol. 14, no. 6. — Pp. 1202–1213.

[20] *Freschi, V.* A faster algorithm for the computation of string convolutions using LZ78 parsing / V. Freschi, A. Bogliolo // *Information Processing Letters.* — 2010. — Vol. 110, no. 14. — Pp. 609 – 613.

[21] A survey of multiple sequence alignment techniques / X. Wang, J. Liu, Y. Xu, J. Zhang // International Conference on Intelligent Computing / Springer. — 2015. — Pp. 529–538.

[22] MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform / K. Katoh, K. Misawa, K. Kuma, T. Miyata // *Nucleic acids research.* — 2002. — Vol. 30, no. 14. — Pp. 3059–3066.

[23] *Sadovsky, M. G.* The method to compare nucleotide sequences based on the minimum entropy principle / M. G. Sadovsky // *Bulletin of Mathematical biology.* — 2003. — Vol. 65, no. 2. — Pp. 309–322.

[24] *Садовский, М. Г.* О сравнении символьных последовательностей / М. Г. Садовский // *Вычислительные технологии.* — 2005. — Vol. 10, no. 3.

[25] *Tsarev, S. P.* New error tolerant method for search of long repeats in DNA sequences / S. P. Tsarev, M. G. Sadovsky // International Conference on Algorithms for Computational Biology / Springer. — 2016. — Pp. 171–182.

[26] *Tsarev, S. P.* Fast Algorithm for Vernier Search of Long Repeats in DNA Sequences with Bounded Error Density / S. P. Tsarev, M. Y. Senashova, M. G. Sadovsky // International Conference on Algorithms for Computational Biology / Springer. — 2018. — Pp. 88–99.

[27] *Benson, D. C.* Fourier methods for biosequence analysis / D. C. Benson // *Nucleic acids research.* — 1990. — Vol. 18, no. 21. — Pp. 6305–6310.

[28] *Newell, W.* Profile searching in nucleic acid sequences using the fast Fourier transformation. — 2001. — US Patent 6,287,773.

[29] *Yin, C.* A measure of DNA sequence similarity by Fourier Transform with applications on hierarchical clustering / C. Yin, Y. Chen, S. S.-T. Yau // *Journal of theoretical biology.* — 2014. — Vol. 359. — Pp. 18–28.

[30] *Fukunaga, K.* Introduction to statistical pattern recognition / K. Fukunaga. — London: Academic Press, 1990.

Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

**УТВЕРЖДАЮ**
заведующий кафедрой
_____ В. В. Шайдуров
"22" июня 2020 г.

# БАКАЛАВРСКАЯ РАБОТА

**Направление** 02.03.01 Математика и компьютерные науки

# ИСПОЛЬЗОВАНИЕ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ И СВЁРТОЧНЫХ ФУНКЦИЙ ДЛЯ СРАВНЕНИЯ НУКЛЕОТИДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Научный руководитель
к. ф.-м. н., доцент _____ 22 июня 2020 Е.Д. Карепова

Выпускник _____ 22 июня 2020 А.А. Молявко

Красноярск 2020