Министерство образования и науки Российской Федерации
Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Факультет Информационных технологий и программирования
Направление (специальность) прикладная математика и информатика
Квалификация (степень) бакалавр прикладной математики и информатики
Специализация ----------
Кафедра Компьютерных технологий **Группа** *4539*

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Разработка метода восстановления фрагментов нуклеотидной последовательности по парным чтениям

Автор квалификационной работы *Сергушичев А.А.* (подпись)

Руководитель *Царев Ф.Н.* (подпись)

К ЗАЩИТЕ ДОПУСТИТЬ

Зав. кафедры *Васильев В.Н.* (подпись)

“____” _____ 2011 г.

Санкт-Петербург, 2011 г.

Квалификационная работа выполнена с оценкой _____

Дата защиты “____” _____ 2011 г.

Секретарь ГАК _____

Листов хранения _____

Чертежей хранения _____

Содержание

Введение

В настоящее время все большее значение в биологии и медицине приобретает анализ ДНК и расшифровка геномов различных живых организмов.

Во-первых, очень важной областью применения анализа генома является медицина. Так, например, знание генома конкретного человека помогает прогнозировать предрасположенности к болезням, что позволяет эффективно проводить профилактику. Знание генома раковой опухоли или организмов-паразитов позволяет эффективно с ними бороться.

Во-вторых, перспективным направлением является генетическая инженерия, методами которой можно изменять генотип организмов, создавая тем самым организмы с более привлекательными характеристиками. Для этого важно иметь расшифрованные геномы организмов, с которыми проводится работа.

В-третьих, анализ геномов различных видов позволяет улучшить познания в области эволюции и истории.

Хотя геном человека уже был почти полностью восстановлен в ходе проекта «Геном человека» [1] и теперь, зная образец, получить геном любого другого здорового человека стало проще, оставшиеся задачи все еще требуют свего решения.

В ходе физического считывания молекул ДНК возникают огромные объемы данных — сотни гигабайт, которые затем необходимо обработать и получить несколько строчек суммарным размером в несколько гигабайт — восстановленный геном. К сожалению, существующее программное обеспечение не справляется с этой задачей полностью, поэтому следует искать новые пути к решению этой проблемы.

Целью данной работы является развитие метода восстановления фрагментов, предложенного в работе [2], чтобы он мог работать с боль-

шими геномами в несколько миллиардов нуклеотидов, используя при этом относительно небольшой объем памяти и тратя на это относительно небольшое время.

Глава 1. Задача секвенирования ДНК

В данной главе приведено описание предметной области. Приведены основные понятия, используемые технологии. В конце главы находится формальная постановка задачи.

1.1. Геном

В ходе различных экспериментов к середине двадцатого века было установлено, что вся наследственная информация об организме находится в молекулах дезоксирибонуклеиновой кислоты (ДНК). В 1953 году Уотсоном и Криком была предложена модель двухцепочечной спирали [3], что затем нашло подтверждение.

Молекула ДНК состоит из двух цепочек, являющихся последовательностями нуклеотидов: аденина (A), тимина (T), гуанина (G) и цитозина (C). Нуклеотиды в парах A-T и G-C комплементарны друг другу и в этих цепочках на одинаковых позициях находятся комплементарные нуклеотиды. Два конца любой цепочки ДНК отличаются с химической точки зрения, один из них называется $5'$, другй — $3'$. Таким образом, у любой цепочки можно определить направления, обычно прямы считается направление от $5'$ к $3'$, что обусловлено тем, что большинство биологических процессов по обработке цепочки ДНК протекают именно в этом направлении. Две цепочки в одной спирали противонаправлены.

Наследственная информация расположена не в одной молекуле ДНК, а, вообще говоря, в нескольких, обычно расположенных в хромосомах. Кроме того, у некоторых организмов, к которым, например, относится большинство млекопитающих, набор хромосом удвоен, такие организмы называются диплоидными. В каждой паре находится по одной хромосоме от каждого родителя, причем эти хромосомы, за исключением пары половых хромосом, отличаются в относительно небольшом числе нуклеотидов,

такое отличние в одном нуклеотиде называется однонуклеотидным полиморфизмом (*англ.* single nucleotide polymorphism), в дальнейшем будем их называть SNP .

1.2. Существующие методы секвенирования

В данном разделе кратко описаны существующие методы секвенирования. В основе любого метода лежит технология, позоляющая физически узнавать, в какой последовательности находятся нуклеотиды в какой-то части генома. С развитием этих технологий изменялись и методы секвенирования геномов.

1.2.1. Метод обрыва цепи

Фредериком Сенгером к 1977 году был разработан метод обрыва цепи [4]. Образец цепочки ДНК, которую необходимо секвенировать, делится на четыре группы, по одной на каждый нуклеотид. Затем в каждой группе происходит de novo синтез молекулы нуклеиновой кислоты из, например, радиоактивно меченных нуклеотидов, причем этот синтез может завершится только в нуклеотиде, соответствующем группе. После этой операции в каждой группе получаются молекулы разных весов, в зависимости от того, в каких позициях находится соответствующий группе нуклеотид в исходной цепочке. Далее в каждой группе с помощью электрофореза происходит разделение молекул по весам с точностью до одного нуклеотида, что позволяет узнать в каких позициях этот нуклеотид встречается в исходном цепочке.

В настоящее время этот метод позволяет напрямую секвенировать последовательности порядка 800–1000 нуклеотидов.

1.2.2. Метод дробовика

Для того чтобы секвенировать геномы большие тысячи нуклеотидов, которые позволяет сделать метод обрыва цепи, было предложено два

метода: метод обхода хромосомы (*англ.* chromosome walking) [5] и метод дробовика (*англ.* shotgun sequencing) [6].

Метод обхода хромосомы заключается в том, чтобы сначала секвенировать первую тысячу нуклеотидов, затем, зная последние нуклеотиды первого прочитанного фрагмента и зацепившись за его конец, секвенировть очередную тысячу нуклеотидов и т.д. Такой метод оказываются неэффективным для хоть сколько-нибудь больших геномв.

Более удобным для применения к большим геномам оказывается метод дробовика, в котором сначала молекулы ДНК случайным образом разделяются на небольшие фрагменты, которые затем секвенируются с помощью, например, метода обрыва цепи. За счет большого числа итераций получается множество последовательностей, покрывающих весь геном, причем на основе анализа их наложений друг на друга можно попытаться восстановить весь геном. Таким образом, задача секвенирования становится не только биохимической, но и вычислительной.

1.2.3. Высокопроизводительные методы секвенирования

В связи с ростом производительности вычислительной техники появилась возможность использовать методы, которые дают менее качественные результаты, зато больше и дешевле.

Например, перспективной является следующая достаточно дешевая и эффективная технология: сначала вычленяется случайно расположенный в геноме фрагмент длиной около 500 нуклеотидов, а затем происходит считывание двух последовательностей с его концов (длиной примерно по 100 нуклеотидов каждая). Эти последовательности называются парными чтениями. Процесс повторяется такое число раз, чтобы обеспечить достаточно большое покрытие генома чтениями. Такая технология реализуется, например, в секвенаторах компании Illumina [7]. Заметим, что при физическом чтении могут возникать ошибки, но для каждого прочитанного нуклеотида известно его качество — веичина, на основе которой можно вычислить

вероятность того, что он был прочитан неправильно.

1.3. Существующие методы сборки

Так как после секвенирования фрагментов ДНК получается множество отдельных частей генома, процесс восстановления генома из этих данных называется сборкой генома. Существующие программные средства для сборки не позволяют восстановить геном целиком, поэтому результатом работы сборщиков является набор контигов, объединенных в скэффолды. Контигами называются максимальные подпоследовательности генома, которые удалось восстановить. Скэффолдами называются такие последовательности контигов с оценками на расстояния между ними, про которые предполагается, что они в той же последовательности и на таких же расстояниях находятся в геноме.

На данный момент существует множество сборщиков геном из парных чтений, большинство из которых полностью основаны на использовании графа де Брюина. Граф де Брюина размерности k над алфавитом Σ — это граф, вершинами которого являются все строки над алфавитом Σ длины k (они называются k-мерами), а ребра соединяют пары таких вершин, что суффикс длины $k-1$ первой вершины является префиксом второй вершины. Можно заметить, что есть однозначное соответствие между ребрами и $(k+1)$-мерами — каждому ребру соответствует $(k+1)$-мер, полученный конкатенацией k-мера начальной вершины ребра и последнего символа k-мера конечной вершины ребра. Будем в дальнейшем говорить, что на ребре, входящем в некоторый k-мер, стоит последний символ этого k-мера. В графе де Брюина над алфавитом $\{A, T, G, C\}$ геном представляет из себя набор путей (возможно не простых), соответствующих хромосомам.

В этом разделе приведено описание подходов, применямых в ралличных существующих сборщиках для сборки контигов. Все описанные методы в начале своей работы выполняют исправление ошибок. Так как это не относится напрямую к предмету исследования, этот шаг будет опущен.

1.3.1. ABySS

В статье [8] изложен подход к сборке контигов в программном средстве ABySS. Этот подход состоит из двух этапов:

- сборка контигов без учета парной информации;
- разрешение неоднозначностей с помощью парной информации и наращивание контигов.

В основе всего подхода лежит распределенный граф де Брюина.

Для того, чтобы собрать первоначальные версии контигов происходит объединение последовательностей смежных однозначных ребер — ребро называется однозначным, если исходящая степень его начальной вершины и входящая степень конечной вершины равны единице.

На втором этапе между контигами устанавливаются связи, используя парную информацию. Пара чтений называется связывающей два контига, если первое чтение картируется на первый контиг, а второе — на второй. Между двумя контигами устанавливается связь, если число связывающих их чтений больше некоторой константы p (по умолчанию используется $p = 5$). Для каждого контига C_i строится множество связанных с ним контигов P_i. Затем в графе связей контигов ищется уникальный путь, проходящий через все контиги из P_i. В качестве ограничений при поиске выступают оценка на расстояния между контигами на основе принципа максимального правдоподобия и эвристическая оценка на число посещенных вершин. После того, как поиск таких путей для каждого контига завершился (успешно или нет), согласующиеся пути сливаются, образуя конечные контиги.

1.3.2. ALLPATHS

Алгоритм, лежащий в основе сборщика ALLPATHS изложен в статье [9]. Он также основан на графе де Брюина и состоит из нескольких этапов.

Первым этапом является создание начального приближения контигов. Для этого, как и в ABySS, рассматриваются последовательности смежных однозначных ребер, которые и объявляются приближением контигов.

Вторым шагом является выбор контигов, вокруг которых будет происходить локализация — попытка выделить те чтения, которые картируются на реальный геном близко к положению контига. В качестве таких контигов лучше всего брать длинные контиги, уникальные в геноме. Для проверки уникальности можно использовать ожидаемое и реальное покрытие чтениями.

Затем для каждого выбранного контига — центра локализации — рассматривается множество последовательностей (чтений и контигов), лежащих в пределах, например, 10000 нуклеотидов от краев этого контига — окрестность контига. Если построить приближение этого множества, то можно будет собрать небольшой участок генома, практически только из тех чтений, которые в него картируются. Это позволяет существенно уменьшить объем данных. Для начала выделяется набор почти уникальных в геноме первоначальных контигов, лежащих в рассматриваемом промежутке. Это происходит с помощью итеративного добавления новых контигов, связанных с уже найденными (рис. 1.1), при этом для каждого контига можно узнать его смещение относительно центра (с какой-то погрешностью), что позволяет отбросить далеко отстоящие контиги. Затем строятся два множества чтений: первичное, состоящее из чтений картирующихся в рассматриваемую область генома, но, возможно, не всех таких чтений, и вторичное, которое кроме интересующих чтений, возможно, содержит и некоторые другие. В первичное множество входят, все пары чтений, хотя бы одно из которых картируется на какой-либо из выбранных контигов. Во вторичное множество чтений входят те пары чтений, которые могут быть собраны из чтений из первичного множества. Затем перекрывающиеся пары чтений из вторичного множества сливаются, образуя пары более длинных последовательностей.

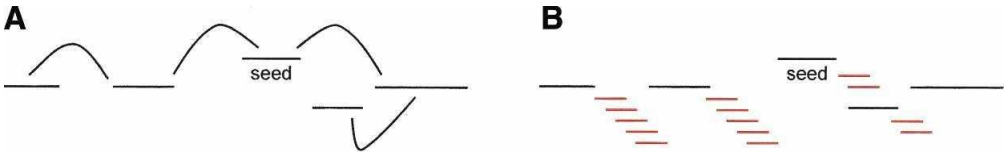

Рис. 1.1. Построение окрестности контига

Для каждой окрестности контига в полученных на предыдущем этапе парах последовательностей ищутся все пути, соединяющие первую последовательность со второй, состоящие из последовательностей из того же множества. Путем является последовательность последовательностей, перекрывающихся с предыдущей не менее чем на K нуклеотидов.

На следующем этапе происходит склеивание путей, найденных на предыдущем этапе, если они имеют длинный общий подпуть (рис. 1.2). Таким образом, осуществляется локальная сборка генома. Затем результаты локальных сборок сливаются между собой, образуя конечный вариант сборки генома.

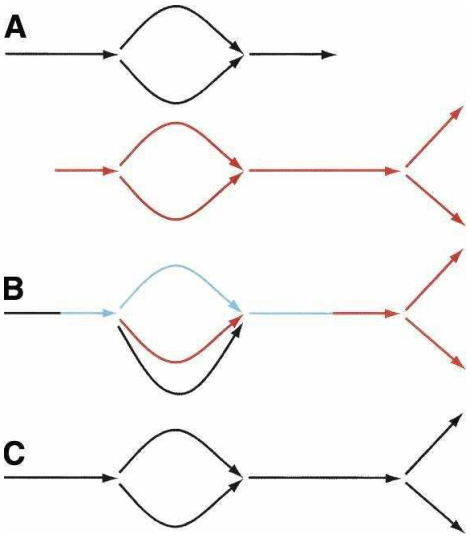

Рис. 1.2. Слияние путей

1.3.3. Velvet

Алгоритм работы Velvet изложен в статье [10].

После исправления ошибок остается граф, вершинами которого являются последовательности, соответствующие вершинам графа де Брюина, объединенным по однозначным ребрам. С этого момента начинает работу

модуль Breadcrumb, ответственный за разрешение проблемных участков с помощью парной информации.

Зная распределение длин пропусков в парных чтениях, можно выделить длину L такую, что число парных чтений, имеющих пропуск длиннее L, ничтожно мало. Можно выделить «длинные» вершины — вершины, длины которых не меньше чем L. Затем выделяются парные «длинные» вершины, связанные несколькими парными чтениями (рис. 1.3). Если после «длинной» вершины может следовать несколько «длинных» вершин, то она называется неоднозначной. Все узлы, на которые картируется чтение, парное к которому картируется на однозначную «длинную» вершину, помечаются. После этого уникальные узлы последовательно наращиваются, добавлением помеченных узлов, связанных с текущим. Этот процесс продолжается до тех, пор пока не встречается узел, у которого нет продолжения или есть несколько вариантов продолжения. Если несколько «длинных» вершин последовательно соединились такими путями, они объединяются в контиг.

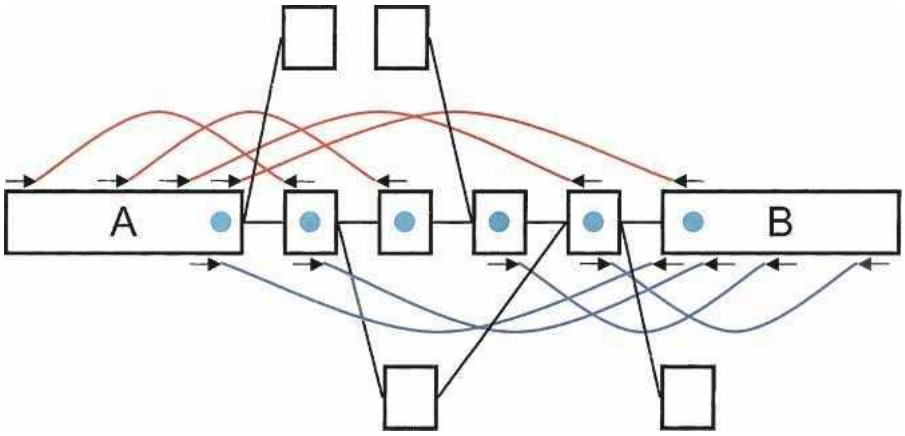

Рис. 1.3. Процесс работы алгоритма Breadcrumb

1.3.4. SOAPdenovo

Способ сборки, используемый в SOAPdenovo, описан в статье [11].

Как и многие другие сборщики из коротких чтений SOAPdenovo использует граф де Брюина. После исправления ошибок пути, не содержа-

щие вершин с ветвлениями, объявляются первоначальным приближением контигов (рис. 1.4).

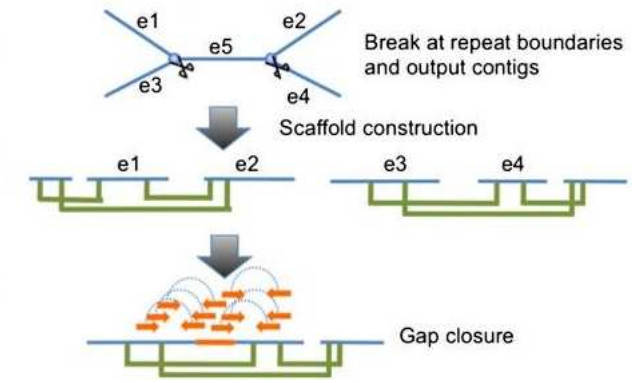

Рис. 1.4. Схема последних этапов работы SOAPdenovo

Затем происходит картирование парных чтений на контиги. Между двумя контигами устанавливается порядок и оценивается расстояние между ними, если их соединяют как минимум три пары чтений. Далее наборы контигов с совместимыми связями линеаризуются, образуя скэффолды.

Во время следующего этапа происходит заполнение промежутков между соседними контигами в скэффолдах. Для этого для каждой пары соседних контигов выбирается множество парных чтений, из котрых одно картируется на один из контигов. Далее проводится локальная сборка только из этих чтений. Если она завершается успешно, тогда получается соединить два контига в один, более длинный.

1.3.5. EULER-USR

В статье [12] изложены основы работы средства EULER-USR из набора EULER.

Сборщик EULER использует для сборки сведение к задаче о поиске эйлерова суперпути — пути, проходящем через все ребра и содержащем в качестве подпутей все пути из наперед заданного множества P. Эта задача, в свою очередь, сводится к задаче поиска эйлерова пути. Такая постановка задачи была сформулирована в статье [13]. В этой статье предлагается в качестве графа использовать граф де Брюина, а в качестве множества P

для начала использовать пути, соответствующие чтениям. Затем, после некоторых преобразований, упрощающих граф, рассмотреть все парные чтения (r_1, r_2) и найти все пути, соединяющие r_1 и r_2 и имеющие длину около ожидаемой, определенной из априорного распределения длин. Если такой путь единственный, то он добавляется в множество P. Этот процесс повторяется до тех пор, пока в графе происходят изменения.

В статье [12] предлагается способ учета и тех парных чтений, для которых нашлось несколько путей. Пусть для парных чтений $(read_{start}, read_{end})$ нашлось несколько путей (рис. 1.5). Обозначим за e_{start} и e_{end} ребра, в которых начинаются и заканчиваются $read_{start}$ и $read_{end}$ соответственно. Для каждого пути p, соединяющего $read_{start}$ и $read_{end}$, рассмотрим величину $support(e_{start}, e_{end}, p)$ — число парных чтений, поддерживающих этот путь — таких парных чтений (r_1, r_2), что одно из них картируется на ребро e_{start} или e_{end}, а другое на путь p. Из всех путей выбирается путь с максимальным значением $support$, если оно не меньше некоторого порогового значения $MinSupport$.

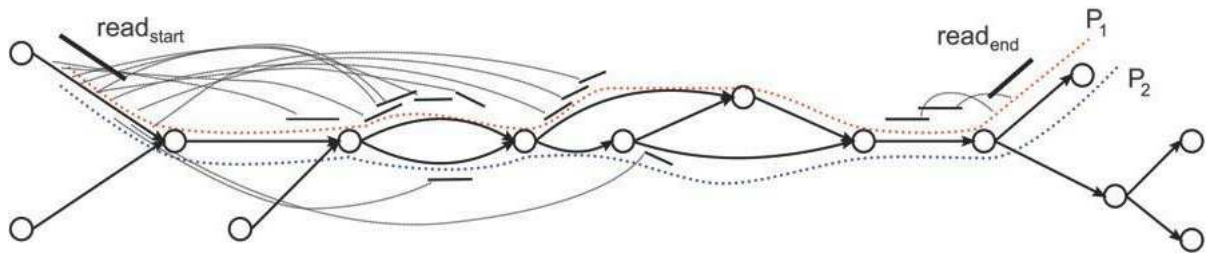

Рис. 1.5. Пути между $read_{start}$ и $read_{end}$

1.3.6. Достоинства и недостатки приведенных методов

Все приведенные методы достаточно хорошо работают на тех размерах генома, на которые они расчитаны. К сожалению, из приведенных выше программ только две — ABySS и SOAPdenovo — способны к сборке больших геномов, но они требуют большого памяти — например, для сборки генома объемом около двух миллиардов нуклеотидов первому необходимо около 200 ГБ оперативной памяти, возможно в сумме на кластере,

второму — около 130 ГБ, но физически на одной машине.

1.4. Постановка задачи

В работе [2] была предложена идея с помощью графа де Брюина восстанавливать фрагменты, отвечающие парным чтениям, а затем использовать полученные последовательности для сборки генома с помощью имеющихся сборщиков из «длинных» чтений, основанных, например, на алгоритме Overlap-Layout-Consensus [14], таких как Newbler [15] или Celera Assembler [16]. К сожалению, предложенный в этой работе алгоритм восстановления фрагментов оказался неприменим для больших геномов, размером от миллиарда нуклеотидов.

В данной работе делается попытка развить метод восстановления фрагментов, чтобы он стал эффективным по используемой памяти и времени работы и его можно было применить к геномам размером до нескольких миллиардов нуклеотидов в услових достаточно ограниченного объема памяти.

Таким образом, стояли следующие задачи:

- Разработка метода для восстановления фрагментов для больших геномов.
- Реализация этого метода на каком-либо из языков программирования.
- Тестирование этой реализации на чтениях, полученных из большого генома.

Глава 2. Предлагаемый метод

В этой главе описан предлагаемый метод.

2.1. Общее описание

В предлагаемом методе используется подграф графа де Брюина, в котором множество ребер состоит только из «надежных» $(k+1)$-меров — тех, которые встречаются в чтениях достаточно большое число раз, не меньшее некоторого порогового значения, чтобы их можно было с очень большой вероятностью считать входящими в геном. Множество вершин состоит из тех вершин графа де Брюина, которым инцидентно хотя бы одно из выбранных ребер. Если участок нуклеотидной последовательности покрылся достаточно хорошо, то есть все входящие в него $(k+1)$-меры по много раз входят в исходные данные, то в этом подграфе существует путь между первым и последним k-мерами участка

Предлагаемый метод основан на поиске такого пути для фрагмента, соответствующего парным чтениям. Из всех путей нас интересуют только те, которые укладываются в априорные границы длин фрагментов, поэтому слишком короткие и слишком длинные пути можно отбросить. Из оставшихся путей следует выбрать те, из которых действительно могли получится имеющиеся парные чтения. Такие пути – хорошие кандидаты на роль пути, соответствующего фрагменту в действительности. Если нашелся единственный такой путь, то можно с очень большой уверенностью сказать, что он отвечает реальному пути в геноме, поэтому этот фрагмент считается восстановленным, а найденный путь выводится. Если таких путей несколько, то не ясно какой из них на самом деле соответствует фрагменту, поэтому этот фрагмент не восстанавливается. Если не нашлось ни одного такого пути, то данный фрагмент ДНК был плохо покрыт чтениями и его восстановление невозможно.

2.2. Используемые алгоритмы

В этом разделе приводится описание алгоритмов, используемых в предлагаемом методе.

2.2.1. Оценка «правдоподобности» пути

Пусть s — строка из нуклеотидов. Будем обозначать строку, обратно-комплементарную s как $\overleftarrow{s}$.

Рассмотрим парные чтения r_1 и r_2 длины l_r, s_1 и s_2 — строки их нуклеотидов ($s_{i,j}$ — j-й нуклеотид i-го чтения), q_1 и q_2 — векторы вероятностей ошибок ($q_{i,j}$ – вероятность неправильного прочтения в j-й позиции i-го чтения. Пусть a_1 и a_2 — их первые k-меры, а p — путь длины $l-k$, соединяющий в рассматриваемом графе a_1 и $\overleftarrow{a_2}$. Этому пути естественным образом соответствует строка s длины l, полученная конкатенацией a_1 и $l-k$ символов, последовательно стоящих на ребрах этого пути. В дальнейшем, будем это преобразование будет обозначать $s(p)$.

Необходимо оценить «правдоподобность» утверждения, что s является последовательностью нуклеотидов в том фрагменте, который породил чтения r_1 и r_2. Для этого сэмулируем безошибочные чтения $\tilde{s}_1$ и $\tilde{s}_2$ — в качестве $\tilde{s}_1$ возьмем первые l_r символов s, а в качестве $\tilde{s}_2$ — обратно-комплементированные последние l_r символов. Посчитаем вероятность того, что при чтении ошибки были допущены именно в тех позициях, где s_1 не совпадает с $\tilde{s}_1$ и s_2 не совпадает с $\tilde{s}_2$, обозначим это событие как $R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}$. Событие, произошедшее в отдельной позиции j чтения i, обозначим $R'_{i,j}$. Если исходить из того, что ошибки в каждой позиции возникают независимо (что в действительности не совсем так, но для оценки этого достаточно), то вероятность $P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2})$ будет находится по формуле:

$$P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}) = \prod_{i=1}^{2} \prod_{j=1}^{|\tilde{s}_i|} P(R'_{i,j}) \tag{2.1}$$

где

$$P(R'_{i,j}) = \begin{cases} q_{i,j}, & \text{если } \tilde{s}_{i,j} \neq s_{i,j}, \\ 1 - q_{i,j}, & \text{иначе.} \end{cases} \tag{2.2}$$

Теперь рассмотрим математическое ожидание этой вероятности и ее дисперсию, чтобы оценить, насколько произошедшее событие обычно. Для начала заметим, что из (2.2) следуют равенства:

$$E(P(R'_{i,j})) = q_{i,j}^2 + (1 - q_{i,j})^2 \tag{2.3}$$

и

$$E(P(R'_{i,j})^2) = q_{i,j}^3 + (1 - q_{i,j})^3. \tag{2.4}$$

Из того, что в нашей модели происхождение ошибок в разных позициях независимо, и из формул (2.1) и (2.3) для математического ожидания $E(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))$ можно получить следующую формулу:

$$E(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2})) = E(\prod_{i=1}^{2} \prod_{j=1}^{|\tilde{s}_i|} P(R'_{i,j})) = \prod_{i=1}^{2} \prod_{j=1}^{|\tilde{s}_i|} E(P(R'_{i,j})) =$$

$$= \prod_{i=1}^{2} \prod_{j=1}^{|\tilde{s}_i|} (q_{i,j}^2 + (1 - q_{i,j})^2). \tag{2.5}$$

Формула для вычисления дисперсии $D(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))$ получается из формул (2.4) и (2.5):

$$D(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2})) = E(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2})^2) - E(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))^2 =$$

$$= \prod_{i=1}^{2} \prod_{j=1}^{|\tilde{s}_i|} E(P(R'_{i,j})^2) - E(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))^2 =$$

$$= \prod_{i=1}^{2} \prod_{j=1}^{|\tilde{s}_i|} (q_{i,j}^3 + (1 - q_{i,j})^3) - \prod_{i=1}^{2} \prod_{j=1}^{|\tilde{s}_i|} (q_{i,j}^2 + (1 - q_{i,j})^2)^2. \tag{2.6}$$

Для оценки «правдоподобности» можно использовать отношение:

$$L_{s_1,s_2}(\tilde{s}_1, \tilde{s}_2) = \frac{|P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}) - E(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))|}{\sqrt{D(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))}}. \tag{2.7}$$

Если для пути p величина $L_{s_1,s_2}(\tilde{s}_1, \tilde{s}_2)$ меньше некоторого заданного порога L_{max}, то есть

$$L_{s_1,s_2}(\tilde{s}_1, \tilde{s}_2) < L_{max}, \tag{2.8}$$

то будем считать его «правдоподобным». В частности, из неравенства Чебышева следует, что если реальными последовательностями нуклеотидов являются $\tilde{s}_1$ и $\tilde{s}_2$, то вероятность того, что будут прочитаны последовательности $\overline{s}_1$ и $\overline{s}_2$, для которых $L_{\overline{s}_1,\overline{s}_2}(\tilde{s}_1, \tilde{s}_2)$ не меньше чем L_{max}, не больше чем $\frac{1}{L_{max}^2}$.

Отметим, что соотношение (2.7) определено и для строк, которые короче, чем s_1 и s_2.

2.2.2. Поиск путей

Для начала введем некоторые обозначения. Пусть путь p_1 длины l_1 ведет из вершины v_1 в вершину v_2, а путь p_2 длины l_2 ведет из вершины v_2 в вершину v_3. Будем обозначать конкатенацию этих путей, то есть путь длины $l_1 + l_2$, соединяющий вершины v_1 и v_3, проходящий сначала по пути p_1, а затем — по p_2, как $p_1 \cdot p_2$. Рассмотрим множества путей P_1 и P_2. С помощью $P_1 \cdot P_2$ будем обозначать все пути, которые можно получить конкатенацией путей $p1$ и p_2 из P_1 и P_2 соответственно, то есть $P_1 \cdot P_2 = \{p_1 \cdot p_2 \mid p_1 \in P_1, p_2 \in P_2\}$.

Теперь рассмотрим задачу поиска путей, соединяющих две заданные вершины v_1 и v_2, длины которых лежат в промежутке $[l_{min}; l_{max}]$. Будем обозначать множество всех путей из v_1 в v_2 длины l как P^l, тогда искомое множество всех путей из v_1 в v_2 будет получаться объединением множеств P^l:

$$P = \bigcup_{l=l_{min}}^{l_{max}} P^l.$$

Для поиска путей будем применять методику meet-in-the-middle, в которой происходит одновременный поиск путей, ведущих из первой вершины, и путей, ведущих во вторую вершину. Обозначим множество всех путей длины l_1, ведущих из первой вершины, за $P_1^{l_1}$, а множество всех путей длины l_2, ведущих во вторую вершину за $P_2^{l_2}$. Применение данной методики к решаемой задаче обусловлено тем, что верно равенство $P^l = P_1^{l_1} \cdot P_2^{l_2}$ для всех $l_1 \in \{0, 1, \ldots, l\}$ и $l_2 = l - L_1$. Действительно, любой путь p длины

l из v_1 в v_2 можно разбить на два более коротких пути p_1 и p_2 длиной l_1 и l_2 соответственно, $p = p_1 \cdot p_2$, $l = l_1 + l_2$, верно и обратное — любые два пути p_1 из $P_1^{l_1}$ и p_2 из $P_2^{l_2}$, если их можно конкатенировать, после конкатенации образуют путь из v_1 в v_2 длины $l = l_1 + l_2$.

Для реализации такого подхода удобно запустить одновременно два обхода в ширину: из первой вершины по прямым ребрам и из второй – по обратным. Тогда на каждом шаге l можно поддерживать следующий инвариант: для первой вершины будем хранить множество $P_1^{l_1}$ всех исходящих из нее путей длины l_1, а для второй – множество $P_2^{l_2}$ всех входящих путей длины l_2, причем $l_1 + l_2 = l$. Таким образом, на l-ом шаге мы можем получить все пути длины l из v_1 в v_2 путем конкатенацией множеств $P_1^{l_1}$ и $P_2^{l_2}$:

$$P^l = P_1^{l_1} \cdot P_2^{l_2}.$$

Начальным шагом для этого алгоритма является l, l_1 и l_2, равные нулю, а P_1^0 и P_2^0, содержащие по одному пути нулевой длины, состоящих из вершин v_1 и v_2 соответственно.

Если E — это множества всех ребер графа, то шаг в первом обходе осуществляется по формуле:

$$P_1^{l_1+1} = P_1^{l_1} \cdot E,$$

а шаг во втором обходе по формуле:

$$P_2^{l_2+1} = E \cdot P_1^{l_1}.$$

Для перехода к следующей итерации необходимо выбрать, в каком из обходов делать шаг. Самым простым является поочередной переход, например, когда номер итерации l четный, делать шаг в первом обходе, а когда нечетный — во втором. Для более эффективного использования памяти и времени лучше производить увеличение в том обходе, в котором на данный момент $P_i^{l_i}$ в каком-то смысле меньше. Сравнение может происходить, например, по числу путей или, если использовать для хранения путей структуру, описанную в разд. 2.3.2, по числу различных концевых вершин.

2.2.3. Обработка путей

В конце работы алгоритма требуется оставить только «правдоподобные» пути, то есть те, для которых выполняется неравенство (2.8). Заметим, что «неправдоподобные» пути можно отсекать уже на ранней стадии. Для этого будем оценивать «правдоподобность» путей из $P_1^{l_1}$ и $P_2^{l_2}$, подставив в неравенство (2.8) для пути p_1 из $P_1^{l_1}$ значения $\tilde{s}_1 = s(p_1)$ и $\tilde{s}_2 = \varepsilon$, где ε – это пустая строка, и для пути p_2 из $P_2^{l_2}$ значения $\tilde{s}_1 = \varepsilon$ и $\tilde{s}_2 = s(p_2)$.

Важно отметить, что значение (2.7) для путей из $P_1^{l_1}$ и $P_2^{l_2}$ может быть пересчитано за $O(1)$ из значений для $P_1^{l_1-1}$ и $P_2^{l_2-1}$, так как это верно для входящих в него значений $P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2})$, $E(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))$ и $D(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))$. Это следует из того, что выражения (2.1) и (2.5) для вычисления значения величин $P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2})$ и $E(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))$ являются произведениями значений функций, зависящих только от одной позиции, а выражение (2.6) для вычисления $D(P(R_{\tilde{s}_1 \to s_1, \tilde{s}_2 \to s_2}))$ является разностью таких произведений.

Из оставшихся «правдоподобных» путей требуется выбрать один, который будет выведен как восстановленный фрагмент. Вообще говоря, если нашлось несколько таких путей, то это сделать сложно, но заметим, что в частном случае, когда нашлось небольшое число близких путей, можно взять любой из них. Это обусловлено тем, что алгоритм overlap-layout-consensus позволяет проверять и неточные совпадения. Необходимость обработки этого случая следует из того, что в геноме встречаются SNP, из-за которых соответствующий путь раздваивается. Поэтому, если на каком-то шаге нашлось несколько путей разной длины или несколько сильно различающихся путей одинаковой длины, то алгоритм можно прервать не дожидаясь l_{max}-ой итерации.

2.2.4. Общий алгоритм

Обобщим вышесказанное в один алгоритм.

В начале инициализируем l_1 и l_2 нулем, а $P_1^{l_1}$ и $P_2^{l_2}$ множествами

состоящими из одного пути нулевой длины.

Затем для каждого l из промежутка $1, 2, \ldots, l_{max}$ выполним следующее:

- Если $P_1^{l_1}$ меньше чем $P_2^{l_2}$, то построим $P_1^{l_1+1} = P_1^{l_1} \cdot E$ и увеличим l_1 на единицу, иначе построим $P_2^{l_2+1} = E \cdot P_2^{l_2}$ и увеличим l_2 на единицу.
- Если требуется, пересчитаем «правдоподобность» путей из $P_1^{l_1}$ и $P_2^{l_2}$ и отбросим «неправдоподобные» пути.
- Если l лежит в промежутке $[l_{min}, l_{max}]$, то построим множество $P^l = P_1^{l_1} \cdot P_2^{l_2}$. Если оно пусто, перейдем к следующей итерации. Если до этого уже был найден путь-кандидат, то завершаем алгоритм — этот фрагмент восстановить не удалось. Если в множестве P^l есть сильно различающиеся пути, то, опять же, выходим. В множестве P^l один или несколько похожих путей — выбираем 1000. любой из них и запоминаем как путь-кандидат.

Если после всех итераций был найден только один путь кандидат, то выводим соответствующую строку, как восстановленный фрагмент.

Так как в некоторых местах графа де Брюина в связи со сложностью генома может быть достаточно высокая средняя степень вершин, то к алгоритму добавляется еще отсечение по размеру множеств $P_1^{l_1}$ и $P_2^{l_2}$ — если хотя бы в одном из них число конечных вершин больше некоторого порога, то алгоритм прерывается.

2.3. Используемые структуры

В этом разделе описаны структуры, которые позволяют эффективно работать алгоритмам, приведенным выше.

2.3.1. Хранение графа

Для того, чтобы потребление памяти предлагаемого метода было не очень большим, необходимо иметь компактное представление используемого подграфа графа де Брюина. Для его хранения достаточно хранить

только множество его ребер, что можно эффективно делать, используя, например, хеш-таблицу с открытой адресацией. Преимуществами такого подхода хранения перед другими являются его простота, быстродействие и возможность балансировки между используемой памятью и скоростью. Более эффективными с точки зрения потребляемой памяти являются rank/select словари [17], которые позволяют сделать ее использование близким к энтропии, но из-за этого увеличивается время доступа.

Важным является и то, что каждый $(k+1)$-мер входит в ребро вместе с обратно-комплементарным. Тогда вместо пары $(k+1)$-меров s и $\overleftarrow{s}$ можно хранить только один, определяемый по некоторому правилу — например, таким правилом может быть выбор лексикографически минимального $(k+1)$-мера. В этом случаем необходимый объем памяти уменьшается примерно в два раза для четных k и ровно в два раза — для нечетных (только для четных k бывают обратно комплементарные себе $(k+1)$-меры).

2.3.2. Хранение путей

Самым простым способом хранения путей является множество, состоящего из строки $s(p)$ для каждого пути p. К сожалению, часто у одного пути возникает несколько продолжений, из-за чего строки приходится копировать, что требует $O(l)$ времени, где l — это длина строки (если бы продолжений было бы не больше одного, то при соответствующей реализации строк на добавление одного символа требовалось бы амортизировано $O(1)$ времени). Этого можно избежать, если для хранения путей использовать граф.

Рассмотрим множество $P_1^{l_1}$ всех путей из v_1 длиной l_1. Построим граф, в котором есть вершины вида (v, l) тогда и только тогда, когда в исходном графе де Брюина существует путь из v_1 в v длиной l. Направленным ребром в этом грфе будет соединены вершины (u, l) и $(v, l+1)$ тогда и только тогда, когда в исходном графе де Брюина есть ребро $u \to v$. Будем называть множество всех вершин вида (v, l) слоем с номером l. Отметим,

что для любого пути из v_1 в v длины l_1 есть соответствующий путь из $(v_1, 0)$ в (v, l_1). Верно и обратное, любому пути из $(v_1, 0)$ в (v, l_1) естественным образом соответствует путь в исходном графе из v_1 в v длины l_1. Обновление этого графа до графа путей длины $l_1 + 1$ требует $O(n)$ времени, где n — число конечных вершин, то есть вершин в l_1-ом слое. Для учета «правдоподобности» будем для каждой вершины (v, l) хранить информацию о значении $L_{s_1,s_2}(s(p), \varepsilon)$ и информацию, необходимую для пересчета этого значения для пути на единицу большей длины, только для одного пути p, который определяется по индукции из предыдущих вершин. Для вершины $(v_1, 0)$ — это путь состоящий из одной вершины v_1. Для вершины (v, l) рассмотрим все пути, заканчивающиеся в этой вершине. Из них выберем только те, префикс которых хранится в одной из вершин слоя $l - 1$. Из оставшихся путей выберем тот, значение $L_{s_1,s_2}(s(p), \varepsilon)$ для которого ближе к нулю, будем его обозначать за $L(v, l)$. При образовании нового слоя, если в какой-то вершине значение $L(v, l_1 + 1)$ будет не меньше чем порог L_{max}, то удалим эту вершину. Заметим, что такое отбрасывание не является эквивалентным отбрасыванию только «неправдоподобных» путей, являясь приближением.

Для множества $P_2^{l_2}$ построение аналогично. В качестве вершин используются пары вида (v, l) тогда и только тогда, когда есть путь длины l из v в v_2. Вершины $(u, l + 1)$ и (v, l) соединены ребром, если в графе де Брюина есть ребро $u \to v$. В качестве $L(v, l)$ используется значение $L_{s_1,s_2}(\varepsilon, s(p))$ для некоторого пути p определяемого из индукции, аналогичной предыдущему случаю.

Глава 3. Экспериментальные результаты

В этой главе приведены результаты экспериментальной проверки программы, написанной на языке *Java*, реализующей предложенный метод.

3.1. Исходные данные

Данный подход был разработан и применен в рамках проекта dnGASP [18]. Тестирование проводилось на данных этого проекта.

В этом проекте предлагалось восстановить синтетический диплоидный геном размером в 1,8 миллиардов нуклеотидов. Частота SNP — около одной тысячной. Геном состоял из 14 хромосом, из которых 11 были получены путем небольших изменений, не нарушающих структуру, из геномов различных организмов, в частности, три из них принадлежали человеку. Оставшииеся три хромосомы были полностью искусственными, созданными для проверки возможности восстановления различных образований, таких как повторы и SNP.

Получение парных чтений из этих данных состояло из нескольких этапов:

- Были сэмулированы безошибочные чтения длиной 114.
- Для каждого чтения было сэмулировано парное ему чтение на случайном расстоянии, исходя из эмпирического распределения длин фрагментов (рис. 3.1), полученного путем картирования существующих чтений генома человека на референсный геном.
- С вероятностью 1/2 выбиралось направление чтений.
- Нуклеотидам чтений приписывались значения качества, на основе профиля качеств реальных чтений.
- В чтения вносились ошибки в соответствии с выставленным качеством.

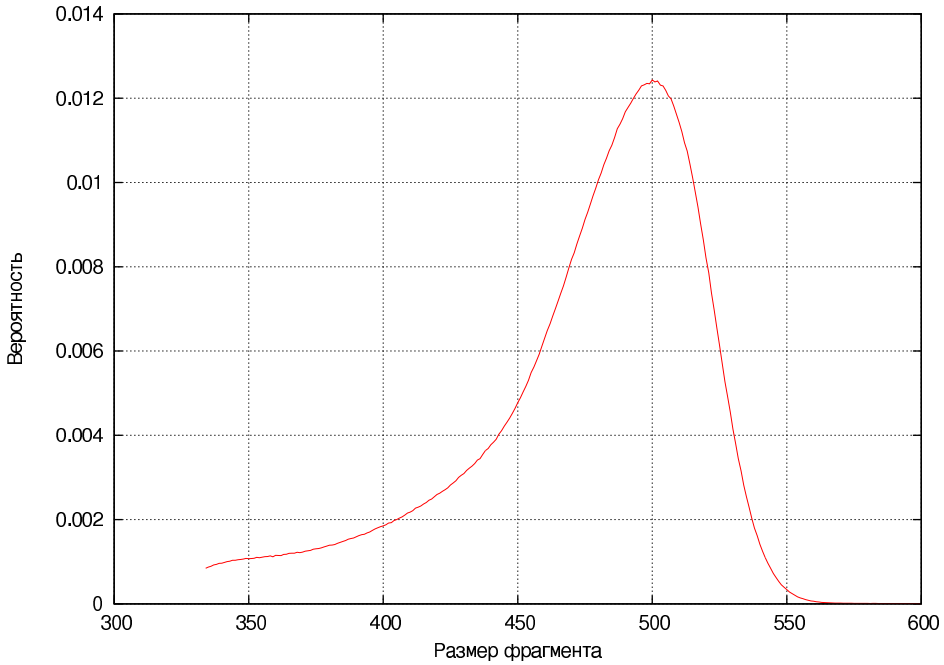

Рис. 3.1. График эмпирического распределения длин фрагментов

Таким образом было получено 44-кратное покрытие, что составило около 350 миллионов парных чтений.

3.2. Результаты

Восстановление фрагментов работало на 24-ядерной машине с 64 ГБ оперативной памяти. Значение k было выбрано равным 30 — в этом случае один $(k+1)$-мер можно хранить в 64-битном целом числе. После этапа исправления ошибок было получено 3,3 миллиарда «надежных» $(k+1)$-меров. «Надежными» считались $(k+1)$-меры, встречавшиеся не меньше четырех раз. Это значение было выбрано на основе частотного распределения 30-меров (рис. 3.2) — зависимости числа 30-меров от того, сколько раз они встречаются в чтениях. В качестве l_{min} было выбрано значение 334, в качестве l_{max} — 550. Максимальным числом концевых вершин в множествах $P_1^{l_1}$ и $P_2^{l_2}$ было выбрано 1000. За сутки были обработаны все 350 миллионов парных чтений, для 67% которых удалось восстановить исходные фрагменты. Для 6% не удалось найти ни одного пути, для 27% фрагментов однозначно восстановить не получилось из-за большого числа путей.

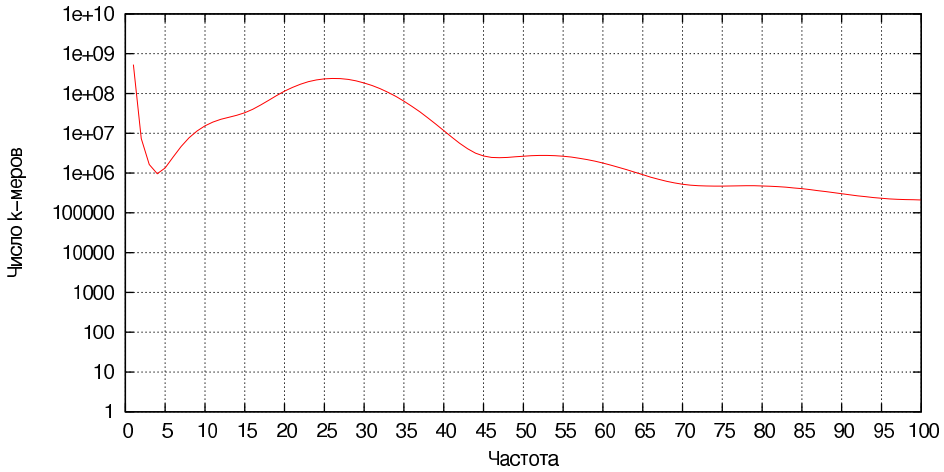

Рис. 3.2. Частотное распределение 30-меров после исправления ошибок

Заметим, что распределение длин восстановленных фрагментов (рис. 3.3) практически полностью совпадает с исходным, при этом от исходного распределения используются только граничные значения. Из этого следует, что восстановится фрагмент или нет, не зависит от длины фрагмента, а, видимо, зависит только от его положения в геноме.

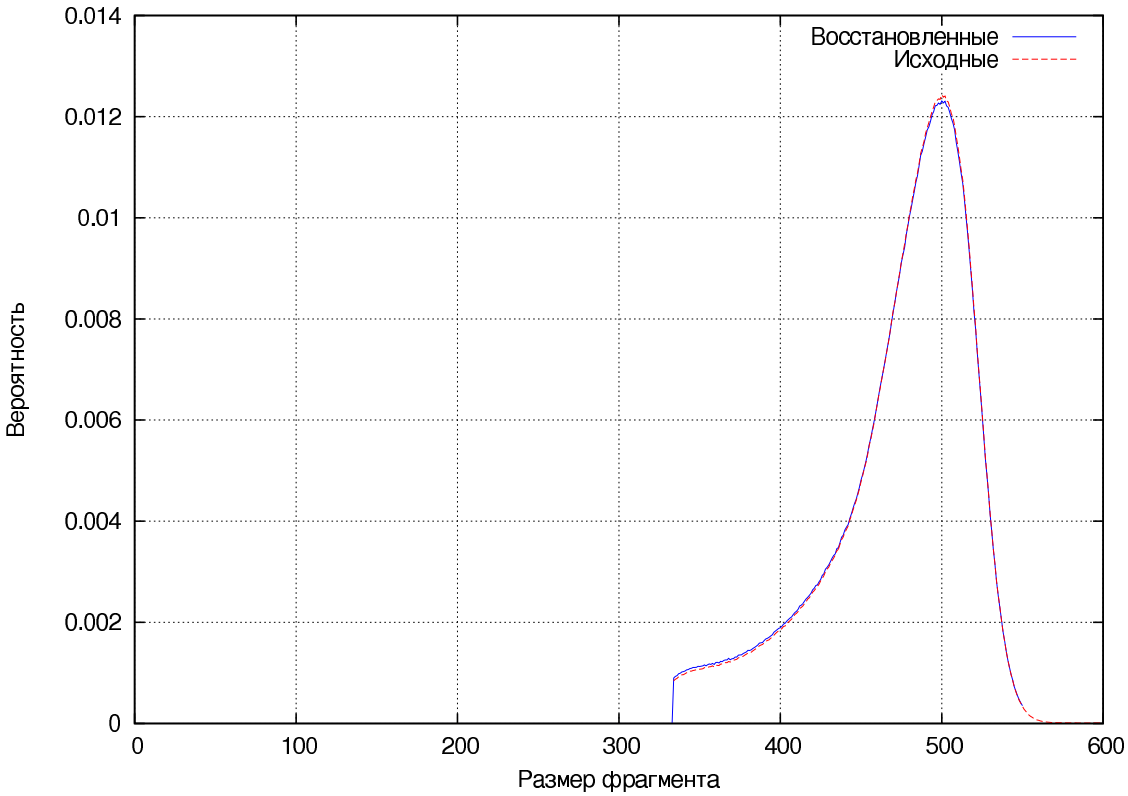

Рис. 3.3. Графики распределения длин восстановленных фрагментов и исходных

Заключение

В ходе работы все поставленные задачи были выполнены:

- Был предложен алгоритм и соответствующие структуры данных, позволяющие достаточно эффективно восстанавливать фрагменты, полученные из больших геномов.
- Была создана реализация предложенного метода на языке *Java*.
- Созданная реализация была протестирована на данных проекта dnGASP.

В дальнейшем имеет смысл сделать реализацию предложенного метода на языке *C++*, что позволит увеличить k до 60. Также следует исследовать наличие других возможностей для повышения доли восстанавливаемых фрагментов.

Источники

[1] *International Human Genome Sequencing Consortium.* Initial sequencing and analysis of the human genome. // *Nature.* Feb 2001. Vol. 409, no. 6822. Pp. 860–921. http://www.nature.com/nature/journal/v409/n6822/pdf/409860a0.pdf.

[2] *Исенбаев В. В., Шалыто А. А.* Разработка системы секвенирования ДНК с использованием paired-end данных. 2010. http://is.ifmo.ru/genom/_isenbaev_thesis.pdf.

[3] *Watson J. D., Crick F. H.* Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. // *Nature.* Apr 1953. Vol. 171, no. 4356. Pp. 737–738. http://profiles.nlm.nih.gov/SC/B/B/Y/W/_/scbbyw.pdf.

[4] *Sanger F., Nicklen S., Coulson A. R.* Dna sequencing with chain-terminating inhibitors. // *Proc Natl Acad Sci U S A.* Dec 1977. Vol. 74, no. 12. Pp. 5463–5467. www.pnas.org/content/74/12/5463.full.pdf.

[5] *Chinault A. C., Carbon J.* Overlap hybridization screening: isolation and characterization of overlapping dna fragments surrounding the leu2 gene on yeast chromosome iii. // *Gene.* Feb 1979. Vol. 5, no. 2. Pp. 111–126.

[6] *Staden R.* A strategy of dna sequencing employing computer programs. // *Nucleic Acids Res.* Jun 1979. Vol. 6, no. 7. Pp. 2601–2610. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC327874/pdf/nar00448-0225.pdf.

[7] Illumina, Inc. http://www.illumina.com/.

[8] *Simpson J. T., Wong K., Jackman S. D., Schein J. E., Jones S. J. M., Birol I.* Abyss: a parallel assembler for short read sequence data. // *Genome Res.* Jun 2009. Vol. 19, no. 6. Pp. 1117–1123. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2694472/pdf/1117.pdf.

[9] *Butler J., MacCallum I., Kleber M., Shlyakhter I. A., Belmonte M. K., Lander E. S., Nusbaum C., Jaffe D. B.* Allpaths: de novo assembly of whole-genome shotgun microreads. // *Genome Res.* May 2008. Vol. 18, no. 5. Pp. 810–820. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2336810/pdf/810.pdf.

[10] *Zerbino D. R., Birney E.* Velvet: algorithms for de novo short read assembly using de bruijn graphs. // *Genome Res.* May 2008. Vol. 18, no. 5. Pp. 821–829. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2336801/pdf/821.pdf.

[11] *Li R., Zhu H., Ruan J., Qian W., Fang X., Shi Z., Li Y., Li S., Shan G., Kristiansen K., Li S., Yang H., Wang J., Wang J.* De novo assembly of human genomes with massively parallel short read sequencing // *Genome Research.* 2010. Vol. 20, no. 2. Pp. 265–272. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2813482/pdf/265.pdf.

[12] *Chaisson M. J., Brinza D., Pevzner P. A.* De novo fragment assembly with short mate-paired reads: Does the read length matter? // *Genome Research.* 2009. Vol. 19, no. 2. Pp. 336–346. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2652199/pdf/336.pdf.

[13] *Pevzner P. A., Tang H., Waterman M. S.* An eulerian path approach to dna fragment assembly. // *Proc Natl Acad Sci U S A.* Aug 2001. Vol. 98, no. 17. Pp. 9748–9753. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC55524/pdf/pq009748.pdf.

[14] *Pevzner P. A.* Computational molecular biology: an algorithmic approach. MIT Press, 2000.

[15] Products & Solutions — Analysis Tools — GS De Novo Assembler : 454 Life Sciences, a Roche Company. http://454.com/products-solutions/analysis-tools/gs-de-novo-assembler.asp.

[16] SourceForge.net: wgs-assembler. http://wgs-assembler.sourceforge.net/.

[17] *Okanohara D., Sadakane K.* Practical entropy-compressed rank/select dictionary // *Computing Research Repository, arXiv:cs/0610001v1.* 2006. http://arxiv.org/pdf/cs/0610001v1.

[18] De novo Genome Assembly Project (dnGASP). http://cnag.bsc.es/.