Федеральное государственное бюджетное учреждение науки
Институт системного программирования им. В. П. Иванникова РАН

На правах рукописи

Карпулевич Евгений Андреевич

Построение программного конвейера для выравнивания последовательностей в приложениях биоинформатики

Специальность 2.3.5 —
"Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей"

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
кандидат физ.-мат. наук
Турдаков Денис Юрьевич

Москва — 2023

Оглавление

Стр.

Введение

Актуальность проблемы. Данные в ряде прикладных и научных областей могут быть представлены в виде последовательностей. Задача "выравнивания" последовательностей находит применение в таких прикладных областях как сжатие данных[1], информационный поиск[2], обработка естественных языков[3] и анализ генетических последовательностей[4].

Выравнивание последовательностей – это математический метод, используемый для определения сходства или различия между двумя или более последовательностями, обычно строками символов, путем их расположения таким образом, чтобы максимизировать совпадения и минимизировать различия.

Методы решения задачи выравнивания последовательностей стали активно развиваться во второй половине 20-го века. В 1965 году В.И. Левенштейн, сотрудник Института прикладной математики им. М.В. Келдыша, ввел понятие метрики редакционного расстояния. Эта метрика (также известная как расстояние Левенштейна[5]) определяется как минимальное количество односимвольных операций (вставки, удаления, замены), необходимых для превращения одной последовательности символов в другую. В 1970 году Сол Б. Нидлман и Кристиан Д. Вунш представили алгоритм глобального выравнивания последовательностей. Алгоритм Нидлмана-Вунша[6] решает задачу наилучшего (оптимального) выравнивания между двумя последовательностями с использованием их полной длины. В 1981 году Т.Ф. Смит и М.С. Уотерман предложили алгоритм локального выравнивания последовательностей. Алгоритм Смита-Уотермана[7] применяется для идентификации похожих подпоследовательностей в последовательностях.

С развитием возможностей вычислительного анализа научным сообществом были предложены алгоритмы на основе эвристик, алгоритмы с использованием машинного обучения и алгоритмы выравнивания на граф. Так, математик Стивен Альтшул из Национального центра биотехнологической информации США в соавторстве со специалистами из области вычислительной биологии в 1990 году

разработал алгоритм и программу[8] (базовый инструмент поиска локального выравнивания). В ранних 2000-х годах получили развитие методы выравнивания на граф последовательностей[9]. В настоящее время также развиваются алгоритмы выравнивания последовательностей с применением машинного обучения[10].

Алгоритмы выравнивания последовательностей нашли свое применение в области обработки генетических данных, в частности, данных секвенирования ДНК (ДНК – это молекула, которая может быть представлена в виде последовательности символов из множества {A, C, G, T}). Секвенирование ДНК – экспериментальный метод определения последовательности расположения символов ДНК в биологическом образце организма. На данный момент, наиболее широко распространенная из существующих технологий секвенирования – секвенирование следующего поколения (NGS[11]). Для получения последовательности ДНК с помощью технологии NGS необходимо произвести несколько шагов: подготовить биологические образцы к секвенированию, получить цифровые данные через обработку подготовленных биологических образцов на специальном приборе (секвенаторе), провести вычислительную обработку[12][13] выходных данных секвенатора (коротких последовательностей ДНК длиной 50-250 символов). Конечным результатом обработки данных NGS является набор генетических вариантов организма (отличий от референсного генома – заранее известной последовательности ДНК абстрактного организма того же биологического вида). Генетические варианты бывают нескольких типов: однонуклеотидные замены (SNP[14]), вставки и делеции[15].

Вычислительная обработка данных NGS обычно состоит из нескольких разнородных (с точки зрения требований к вычислительным ресурсам и возможностей распараллеливания) этапов. Одним из ключевых этапов вычислительной обработки данных NGS является выравнивание коротких подпоследовательностей ДНК, полученных от секвенатора, на референсный геном. В инструментах, которые реализуют этап выравнивания генетических последовательностей, могут применяться алгоритмы двух классов: выравнивание

на линейный референсный геном и выравнивание на граф, составленный по ДНК нескольких организмов. Первый класс алгоритмов обладает высокой скоростью выравнивания, а второй класс алгоритмов обладает большей точностью. Разработка метода и алгоритмов выравнивания, которые сочетают в себе преимущества обоих подходов, является актуальной задачей.

Количество больших данных NGS (объем данных NGS, полученных из одного биологического образца, составляет от нескольких единиц/десятков до сотен гигабайт) постоянно растет благодаря совершенствованию и удешевлению технологии NGS. Проведение масштабных исследований на тысячах биологических образцов, зачастую с участием нескольких лабораторий, находящихся в разных частях мира, порождает ряд требований к вычислительной обработке данных NGS: автоматизация, масштабируемость, воспроизводимость, контроль качества, поддержка совместной работы и передача накопленного опыта в анализе данных.

Для того чтобы организовать непрерывный цикл разработки, тестирования и эксплуатации масштабируемых биоинформатических программных конвейеров, необходимо использовать современные IT-технологии: системы управления программными конвейерами, облачные вычисления[16], контроль версий, контейнеризацию, планировщики задач. Требования к оптимизации вычислительных мощностей, снижению затрат на разработку и развитию программных конвейеров делают актуальной разработку архитектуры воспроизводимых масштабируемых систем анализа данных NGS.

Целью данной работы является разработка алгоритмов и метода выравнивания последовательностей для решения задачи секвенирования ДНК, а также разработка и реализация архитектуры воспроизводимых биоинформатических программных конвейеров обработки данных секвенирования ДНК человека. Разработанная реализация программного конвейера для обработки данных секвенирования ДНК человека должна

превосходить существующие реализации по качеству идентификации однонуклеотидных полиморфизмов.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Разработать метод выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах
2. Разработать алгоритмы в составе метода выравнивания генетических последовательностей и получить аналитические оценки их вычислительной и пространственной сложности
3. Разработать и реализовать архитектуру системы анализа данных NGS на базе программного конвейера, реализующего предложенный метод выравнивания генетических последовательностей, а также экспериментально оценить метрики качества идентификации генетических вариантов на данных NGS

Основные положения, выносимые на защиту:

1. Новый метод выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах
2. Алгоритмы в составе метода выравнивания генетических последовательностей и аналитические оценки их вычислительной и пространственной сложности через доказательство соответствующих теорем
3. Архитектура и реализация системы анализа данных NGS на базе программного конвейера для обработки данных секвенирования ДНК человека с использованием модифицированного индекса

Научная новизна. Разработан новый метод выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах, который сочетает в себе преимущества

методов выравнивания на линейный референсный геном и выравнивания на граф, составленный по ДНК нескольких организмов. Разработаны алгоритмы в составе метода выравнивания генетических последовательностей. Доказаны теоремы о их вычислительной и пространственной сложности. Оценки, полученные в результате доказательства теорем, показывают, что вычислительная сложность алгоритмов построения создания модифицированного индекса референсной генетической последовательности остается линейной, а вычислительная сложность алгоритмов выравнивания генетических последовательностей на модифицированный индекс не изменяется по сравнению с выравниванием на индекс референсного генома. Теорема об оценке пространственной сложности позволяет оценить количество оперативной памяти необходимой для работы реализации алгоритмов.

Практическая значимость. Разработана и реализована архитектура системы анализа данных NGS на базе программного конвейера, реализующего предложенный метод выравнивания генетических последовательностей. Предложенные метод выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах реализован посредством модификации функций существующего инструмента выравнивания генетических последовательностей на референсный геном minimap2[17]. Инструмент для выравнивания последовательностей minimap2 используется в том числе в коммерческих решениях (например, MGI MegaBOLT). Добавление информации об известных генетических вариантах в индекс инструмента minimap2 позволило повысить качество выравнивания ридов, что показано экспериментально. Реализация программного конвейера анализа данных NGS секвенирования ДНК человека с использованием модифицированного инструмента minimap2 позволило снизить количество ложноотрицательных срабатываний на 25% (274 SNP) по сравнению с программным конвейером bgallagher-sentieon, победившем в конкурсе PrecisionFDA Truth Challenge[18]. Описана процедура развертывания разработанного программного конвейера на SLURM[19] кластере в облачной среде Asperitas, проведена оценка

функционирования программного конвейера на SLURM-кластере. Результаты работы могут быть использованы в научных исследованиях и промышленных проектах, которые предполагают массовое секвенирование ДНК с помощью технологии NGS.

Апробация работы. Результаты работы докладывались на следующих конференциях:

1. Открытая конференция ИСП РАН, Москва, Россия, декабрь 2021
2. Конференция «MACSPro», Москва, Россия, декабрь 2021
3. Конференция «SIBS» (Сеченовский международный биоинформатический саммит), Москва, Россия, ноябрь 2022
4. Конференция «Ломоносовские чтения» - 2023, Москва, Россия, апрель 2023
5. Конференция «Анализ данных в медицине», Великий Новгород, Москва, июнь 2023

Личный вклад. Все выносимые на защиту результаты получены лично автором.

Публикации. Основные результаты по теме диссертации изложены в трех работах, опубликованных в изданиях, рекомендованных ВАК, кроме того, получено свидетельство о государственной регистрации программы для ЭВМ.

В статье [1] поставлена задача совместно с соавтором, автору принадлежит основная часть: разделы 2-4, реализация инструмента и финальное редактирование текста также выполнены автором.

В статьях [2;3] вместе с соавторами поставлена задача и проводилась редакторская правка, разработка программных конвейеров выполнена автором.

На основе разработанного программного конвейера получено свидетельство о государственной регистрации программы для ЭВМ [4].

Объем и структура работы. Диссертационная работа состоит из введения, четырех глав, заключения и списка литературы, содержащего 96 ссылок. Работа изложена на 123 страницах, содержит 10 рисунков, 17 листингов и 10 таблиц.

Глава 1. Обзор подходов к обработке данных полногеномного секвенирования человека

Первая глава состоит из семи разделов. В разделе 1.1 приведены основные понятия и определения биоинформатического домена, в частности, обсуждаются особенности терминологии, приведен краткий обзор технологий секвенирования и вводится ключевое понятие референсного генома.

Раздел 1.2 полностью посвящен обзору методов и подходов, применяемых для решения биоинформатической задачи ресеквенирования. Выполнен обзор методов выравнивания ридов (подпоследовательностей ДНК длиной 50-250 нуклеотидов). Приведена оценка вычислительной сложности алгоритмов выравнивания коротких прочтений на линейный референсный геном и на пангеномный граф. В разделе 1.3 описаны способы оценки качества анализа выходных данных секвенатора для задачи полногеномного секвенирования (или секвенирования ДНК).

В разделе 1.4 приведен обзор подходов к созданию масштабируемых биоинформатических алгоритмов и программных конвейеров. В разделе 1.5 сделан краткий анализ преимуществ применения облачных вычислений в биоинформатике. В разделе 1.6 описана актуальность создания новых методов для обработки выходных данных секвенатора. Раздел 1.7 содержит выводы по первой главе.

1.1 Основные понятия и определения

1.1.1 Особенности терминологии

В научной литературе по биоинформатике может возникнуть неоднозначное соответствие русских терминов англоязычным. Кроме того, из-за того, что область анализа генетических данных быстро развивается иногда существует несколько названий или вариантов определений одного и того же понятия.

Далее в работе будем придерживаться следующих терминологических тождеств:

- "рид" = "короткое прочтение" = **"read" = "short read"**
- "выравнивание" = "картирование" = **"alignment" = "mapping"**
- "якорь" = **"seed"**
- "конвейер" = "программный конвейер" = "пайплайн" = **"pipeline"**
- "нуклеотид" = "база" = "буква" = **"nucleotide" = "base"**
- "генетический вариант" = "вариант" = "мутация" = **"genetic variant"**
- "идентификация генетических вариантов" = "вызов вариантов" = "определение генетических вариантов" = **"variant calling"**
- "Однонуклеотидный полиморфизм" = "замена" = **"Single-nucleotide polymorphism" = "SNP"**

Отдельно отметим, что у ряда живых организмов хромосомы (структуры внутри клеток, которые содержат гены человека) дублируются, например у человека 23 парные хромосомы (одна из пар - половые хромосомы).. Поэтому в общем случае в одной позиции ДНК человека может быть два разных нуклеотида (буквы). Зачастую в результате анализа расположение нуклеотидов на парных хромосомах не идентифицируется. В случае если взаимное расположение нуклеотидов на хромосомах известно говорят, что геном фазирован.

Кроме того, определим соотношение терминов секвенирование и ресеквенирование. Секвенирование это общее название метода определения последовательности молекулы ДНК РНК или белка, задача ресеквенирования является одним из видов задачи секвенирования, когда речь идет о задаче ресеквенирования подразумевается, что последовательность генома секвенируемого организма известна.

1.1.2 Обзор технологий NGS и проблем обработки данных

Технологии секвенирования нового поколения (Next-generation sequencing, NGS) произвели революцию в исследованиях в области геномики, обеспечив

возможность быстрого и сравнительно недорогого получения последовательностей ДНК и РНК живых организмов, что в свою очередь позволяет решать широкий спектр задач от определения наличия или рисков различных заболеваний до идентификации человека в юридических целях.

Метод секвенирования первого поколения, по Сэнгеру (Sanger sequencing[20]) был предложен еще в 1977 году. Метод секвенирования по Сэнгеру – это классический метод, используемый для определения последовательности нуклеотидов в ДНК или РНК образце. Данный метод обладает рядом преимуществ, такими как высокая точность и относительно низкая стоимость при исследовании небольших фрагментов ДНК. Из недостатков стоит отметить низкую пропускную способность и дороговизну при исследовании большого объема данных.

Для проведения экспериментов по массовому секвенированию в настоящее время широко используются два семейства технологий секвенирования, которые отличаются технологическим процессом, в частности, длиной ридов: технологии NGS и технологии секвенирования третьего поколения.

Подробнее остановимся на технологиях NGS. Технологии секвенирования следующего поколения (NGS) – это современные методы биологического анализа, которые позволяют определить последовательности нуклеотидов в ДНК живых организмов. Алгоритм работы NGS включает в себя следующие этапы:

- Подготовка образца:
    - Из биологического образца, например, крови, ткани или клеток выделяется ДНК
    - ДНК фрагментируется (разбивается на короткие участки)
    - Добавляются адаптерные последовательности к концам фрагментов ДНК. Адаптерные последовательности позволяют фрагментам ДНК связываться с поверхностью проточной ячейки (небольшой физический контейнер, в котором происходит химическая реакция

секвенирования), а также служат для идентификации отдельного образца

- Фрагменты ДНК размещаются на поверхности специальных матриц в проточной ячейке, образуя кластеры, где каждый кластер состоит из одинаковых молекул ДНК.

- Секвенирование:
  - Кластеры подвергаются многократному циклическому секвенированию. За каждый цикл добавляется один нуклеотид
  - Каждый цикл секвенирования определяет один нуклеотид в каждом фрагменте.
- Регистрация сигнала:
  - В процессе секвенирования фиксируется интенсивность света или другой сигнал, который ассоциирован с каждым видом нуклеотидов (A, T, G, C)
  - Результатом этапа регистрации сигнала являются последовательности нуклеотидов для каждого кластера.
- Компьютерный анализ данных.

NGS является сложным процессом, начиная с подготовки образца и завершая анализом полученных данных с использованием компьютерных алгоритмов. Технологии NGS, или так называемые технологии следующего поколения, используются в секвенаторах компаний Illumina и BGI. Выходные данные секвенаторов NGS представляют собой риды длинной 50-300 нуклеотидов, при этом достигается высокое качество прочтения отдельных нуклеотидов. Платформы NGS[21], также известные как секвенаторы, производят сотни гигабайт выходных необработанных данных за один запуск. Чтобы преобразовать выходные данные NGS секвенатора в генетические последовательности, пригодные для дальнего анализа биологами и медицинскими генетиками, требуется произвести ресурсоемкие вычисления.

Технологии секвенирования третьего поколения достаточно сильно отличаются между собой по принципам работы: так, секвенирование Pacific Biosciences[22] (PacBio) использует метод одномолекулярного секвенирования в реальном времени (SMRT[23]), а секвенаторы компании Oxford Nanopore[24] используют секвенирование на основе нанопор, при котором нити ДНК проходят через нанопоры и производят характерные электрические сигналы. В качестве общей черты разных технологий секвенирования третьего поколения можно отметить невысокое по сравнению с NGS качество прочтения отдельных нуклеотидов при значительной длине ридов до десятков килобаз (килобаза – тысяча пар нуклеотидов).

На данный момент наибольшее распространение для задач секвенирования генома человека получили технологии NGS благодаря высокому качеству прочтения нуклеотидов[25] и постоянно снижающейся стоимости секвенирования (сегодня стоимость полногеномного секвенирования в России составляет менее ста тысяч рублей).

Тем не менее, данные, генерируемые платформами NGS, обладают рядом особенностей, которые могут создавать проблемы при их хранении и анализе:

- Объем данных и вычислительная сложность обработки. Платформы NGS генерируют огромные объемы данных, от нескольких гигабайт до нескольких терабайт за цикл секвенирования (два-три дня). Обработка и хранение таких больших наборов данных требует значительных вычислительных ресурсов и емкости хранилища.
- Хранение и управление данными. Зачастую необходимо не только хранить сами данные, но также хранить связанные с ними метаданные, обеспечивать оперативный доступ к данным и метаданным, хранить информацию о версиях инструментов и биологических баз данных, используемых для анализа. Кроме того, необходимо обеспечить репликацию данных для надежного хранения, а также возможность их резервного копирования.

- Качество данных. Ошибки и неточности могут возникать на разных этапах лабораторного процесса: на этапе подготовки библиотеки, этапах амплификации и секвенирования. Данные могут содержать артефакты, такие как ошибки прочтения базовых пар, химерные чтения или загрязнения адаптерных последовательностей, которые необходимо идентифицировать и исправлять.
- Сложности выравнивания ридов на референсную последовательность в процессе ресеквенирования: короткая длина прочтений, полученных при использовании технологий NGS, и маленькая мощность алфавита генетических последовательностей (A, T, G, C) затрудняет сопоставление ридов с эталонным геномом или транскриптомом, особенно в повторяющихся областях или областях ДНК с генетическими вариантами (заменами, вставками, делециями и др.).
- Сложности De Novo сборки[26]. Процесс получения полной последовательности генома организма из данных NGS без наличия эталонного генома, известный как de novo сборка, требует больших вычислительных ресурсов и является сложной задачей, особенно для больших и повторяющихся геномов. Кроме того, в результате зачастую все равно получается не полный геном от и до, а только набор собранных отдельных участков генома, контигов.
- Сложности идентификации генетических вариантов[27]: корректная идентификация генетических вариантов (однонуклеотидные полиморфизмы (SNP) или структурные генетические варианты (вставки, делеции, инверсии)) в геноме секвенируемого организма на основе выходных данных NGS, также является нетривиальной задачей: помимо сравнения нуклеотидов в выровненных ридах с нуклеотидами в эталонном геноме необходимо каким-то образом отличать истинные генетические варианты от ошибок секвенирования и артефактов, что зачастую требует дополнительных шагов, таких как фильтрация и локальная сборка de novo.

- Сложности интеграции и интерпретации данных. Интеграция данных NGS с другими типами омикс-данных[28], такими как протеомные[29] или эпигеномные[30] данные, и извлечение релевантных биологических знаний из генетических данных требуют передовых инструментов и опыта экспертов в области информационных технологий, биологии и медицины.

Для решения этих и других проблем биоинформатики и исследователи разработали большое количество инструментов, алгоритмов и программных конвейеров для обработки и анализа данных NGS, новые инструменты продолжают появляться. Биоинформатические решения направлены на повышение качества анализа данных, вычислительной эффективности обработки и облегчение последующей биологической интерпретации.

1.1.3 Референсный геном человека

Референсный геном[31] – последовательность нуклеотидов абстрактного представителя биологического вида. Также называется референсной сборкой генома. Референсный геном не является результатом секвенирования какого-либо конкретного индивидуума. Референсный геном обычно хранится в файле формата FASTA[32] (файл представлен в виде пар строк: первая строка содержит уникальное описание последовательности, вторая строка – саму последовательность).

Референсные геномы человека играют важную роль в исследованиях геномики, поскольку они предоставляют стандартизированную исходную точку для сравнения и анализа генетической информации. Два наиболее широко используемых референсных генома[33] человека – это hg19 и hg38[34]. Суммарная длина расшифрованного генома составляет около 3 млрд нуклеотидов.

Hg19, также известный как референсный геном человека версии 19, был опубликован в 2009 году и является первым референсным геномом, в котором были собраны и аннотированы все геномные данные человека. Он был разработан Геномным исследовательским институтом США и является результатом многолетних усилий международного научного сообщества. Hg19 построен на

основе данных секвенирования генома нескольких людей и содержит около 3 миллиардов нуклеотидов. Этот референсный геном широко использовался во многих исследованиях и стал основой для многих научных открытий.

1.2 Задача ресеквенирования

Как отмечено ранее процесс получения полной последовательности генома организма из данных NGS, известный как de novo сборка, в общем случае не позволяет получить из выходных данных NGS полную последовательность ДНК секвенируемого организма. Кроме того полученный в результате de novo сборки набор контигов не будет иметь привязки к определенным позициям в ДНК, что не позволит проводить сравнение ДНК различных организмов. Несмотря на то, что сборка генома de Novo обладает некоторыми преимуществами (например, позволяет устранить ошибки референсной последовательности), для исследования организмов с длинным геномом обычно применяется подход повторного секвенирования (ресеквенирования[35]) генома.

Повторное секвенирование возможно при наличии референсного генома секвенируемого организма. При ресеквенировании необходимо выровнять риды нового организма на известную референсную последовательность. Основная задача при повторном секвенировании состоит в том, чтобы точно идентифицировать генетические варианты, такие как однонуклеотидные варианты (SNP), вставки, делеции и структурные варианты в целевом геноме по сравнению с эталонным геномом. Для решения этой задачи необходимо собрать программный конвейер из биоинформатических инструментов для обработки и анализа выходных данных секвенирования, который удовлетворяет пороговым значениям метрик качества (таких как количество выровненных ридов, количество корректно идентифицированных генетических вариантов и др.) на эталонных и сгенерированных данных.

Преимущества ресеквенирования генома человека:

1. Ресеквенирование основывается на уже существующих референсных геномных сборках, таких как hg19 и hg38. Это позволяет исследователям проводить сравнительный анализ, идентифицировать генетические варианты, связанные с конкретными заболеваниями или фенотипами, и

уточнять местоположение генов и других функциональных элементов в геноме.

2. Ресеквенирование может быть более экономически выгодным в сравнении с сборкой de novo. Поскольку оно не требует полной повторной сборки генома, ресеквенирование требует меньше вычислительных ресурсов.
3. Ресеквенирование позволяет проводить анализ на основе существующей геномной информации. Исследователям не нужно собирать и аннотировать новую геномную последовательность. Это упрощает процесс анализа, особенно при работе с большими объемами данных.

1.2.1 Формат выходных данных секвенатора

Выходные данные NGS секвенатора обычно представлены в формате FASTQ[36]. Формат FASTQ содержит помимо последовательности нуклеотидов короткого прочтения дополнительные данные.

Рисунок 1.1 – Формат FASTQ файла

Формат FASTQ (Рисунок 1.1) представляет каждый рид в виде четырехстрочной структуры:

- Строка 1 начинается с символа "@", за которым следует идентификатор короткого прочтения. Для каждого прочтения в файле идентификатор является уникальным.
- Строка 2 содержит фактическую последовательность ДНК или РНК короткого прочтения. Она состоит из символов A, C, G и T (или U в случае РНК), которые с точки зрения биологии являются нуклеотидными основаниями.
- Строка 3 начинается с символа "+" и обычно содержит повторение идентификатора последовательности из строки 1. Она часто используется для обозначения начала строки качества (строка 4).
- Строка 4 содержит показатели качества прочтения каждого основания в последовательности строки 2. Показатели качества отражают вероятность неправильного прочтения нуклеотида в определенной позиции в последовательности, показатели качества Phred логарифмически связаны с вероятностью ошибки. Например, показатель качества 30 соответствует вероятности ошибки 1 к 1000 (0.1%), а показатель качества 20 соответствует вероятности ошибки 1 к 100 (1%). Показатели качества представлены с помощью символов ASCII, где каждый символ соответствует числовому значению. Обычно используется шкала Phred[37][38], где значение ASCII символа равно показателю качества плюс фиксированное смещение (обычно 33). Более высокие показатели качества указывают на более высокую уверенность в корректном прочтении нуклеотида секвенатором.

Само короткое прочтение состоит из адаптерной последовательности и прочитанной последовательности нуклеотидов. Адаптерная последовательность – это короткий участок ДНК, который связывается с концами фрагментов ДНК или РНК во время подготовки образца перед секвенированием. Последовательность адаптера служит различным целям, таким как предоставление стартовых сайтов (мест) для амплификации (процесс увеличения числа копий ДНК) в процессе

секвенирования, обеспечение возможности мультиплексирования для различения разных образцов и облегчение операций управления секвенированием.

1.2.2 Формат файла генетических вариантов

Формат VCF[39] (Variant Call Format) – это стандартизированный файловый формат, используемый для хранения информации о генетических вариантах в геномах различных организмов. VCF-файлы широко используются в биоинформатике для обмена данными и анализа генетических вариантов. Ниже приведено описание основных полей и структуры VCF-файла:

1. Заголовок (Header): Заголовок VCF-файла содержит метаинформацию и описывает структуру и параметры VCF-файла. Заголовок начинается с символа #. В нем могут быть определены следующие типы метаданных:
    - ##fileformat: Версия формата файла (например, "VCFv4.2").
    - ##INFO: Описание информации о генетическом варианте (например, его функции и частоты).
    - ##FORMAT: Описание формата данных о генетическом варианте (например, генотипы).
    - ##FILTER: Описание фильтров, применяемых к генетическим вариантам.
    - ##ALT: Описание альтернативных генетических вариантов.
    - ##SAMPLE: Описание образцов, если используются идентификаторы образцов.
2. Записи (Records): Записи в VCF-файле представляют собой строки, каждая из которых содержит информацию о конкретном генетическом варианте. Запись состоит из следующих полей:
    - #CHROM: Название хромосомы, на которой находится генетический вариант.
    - POS: Позиция в хромосоме (число).
    - ID: Уникальный идентификатор варианта (необязательное поле).

- REF: Основная (референсная) последовательность нуклеотидов на данной позиции.
- ALT: Альтернативная последовательность нуклеотидов (генетический вариант).
- QUAL: Качество идентификации варианта (число).
- FILTER: Фильтры, применяемые к варианту (например, "PASS" или название фильтра).
- INFO: Информация о варианте (например, частота встречаемости генетического варианта в определенной популяции, функциональная аннотация).
- FORMAT: Формат данных о генотипах образцов (например, "GT" для генотипов).

3. Генотипы (Genotypes): После полей FORMAT и INFO идут данные о генотипах образцов (индивидов) для данного генетического варианта. Эти данные представлены в виде строк, каждая из которых содержит информацию о генотипе для конкретного образца. Например, "0/1" обозначает гетерозиготу, а "1/1" – гомозиготу для альтернативного варианта. "0|0" и "1|1" соответственно для фазированных генетических вариантов.

Пример записи в VCF-файле представлен в Листинге 1.1

Листинг 1.1 – Пример записи в VCF-файле

```
#CHROM  POS    ID  REF  ALT  QUAL  FILTER  INFO  FORMAT  Sample1  Sample2
1       10001  .   A    G    20    PASS    .     GT      0/1      1/1
```

В этом примере:

- Генетический вариант находится на хромосоме "1" на позиции "10001".
- Основная последовательность (референсная) – "A", а альтернативная – "G".
- Качество идентификации генетического варианта равно 20, и вариант прошел фильтрацию "PASS".
- INFO поле может содержать дополнительную информацию о варианте.

- Формат данных о генотипах – "GT", и для образца "Sample1" указан генотип "0/1", что означает гетерозиготу для данного варианта.

Формат VCF (Рисунок 1.2) обеспечивает универсальный способ представления генетических вариантов и их аннотации для одного или нескольких образцов сразу.

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS     ID        REF    ALT     QUAL FILTER INFO                              FORMAT      NA00001        NA00002        NA00003
20     14370   rs6054257 G      A       29   PASS   NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,.
20     17330   .         T      A       3    q10    NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3   0/0:41:3
20     1110696 rs6040355 A      G,T     67   PASS   NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2   2/2:35:4
20     1230237 .         T      .       47   PASS   NS=3;DP=13;AA=T                   GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20     1234567 microsat1 GTC    G,GTCT  50   PASS   NS=3;DP=9;AA=G                    GT:GQ:DP    0/1:35:4       0/2:17:2       1/1:40:3
```

Рисунок 1.2 – Пример VCF-файла[40]

1.2.3 Обзор точных методов выравнивания генетических последовательностей

Одним из первых шагов для анализа данных в рамках задачи ресеквенирования является выравнивание ридов полученных от секвенатора на референсный геном. Для точного выравнивания подстрок используются такие методы выравнивания строк как алгоритм Нидлмана-Вунша (Needleman-Wunsch) и алгоритм Смита-Уотермана (Smith-Waterman).

Алгоритм Нидлмана-Вунша (Needleman-Wunsch) – это алгоритм динамического программирования, используемый для выравнивания строковых последовательностей. Этот алгоритм находит оптимальное глобальное выравнивание двух последовательностей, минимизируя штрафы за вставки, удаления и замены.

Описание шагов алгоритма Нидлмана-Вунша:

1. Инициализация матрицы:

   Создается матрица размером $(n+1)\times(m+1)$, где n и m – длины выравниваемых последовательностей.

2. Инициализация крайних значений:

   Заполняются крайние значения матрицы. В первой строке и первом столбце будут находиться значения, соответствующие штрафам за вставки и удаления. Пусть gap_penalty – штраф за вставку или удаление, а match_score и mismatch_penalty – балл и штраф за совпадение и несовпадение соответственно (Листинг 1.2). Значения gap_penalty, match_score и mismatch_penalty определяются исходя из конкретной задачи и, в совокупности, определяют модел ошибки.

Листинг 1.2 – Инициализация крайних значений

```
matrix[0][0] = 0
for i from 1 to n:
   matrix[i][0] = matrix[i-1][0] + gap_penalty
for j from 1 to m:
   matrix[0][j] = matrix[0][j-1] + gap_penalty
```

3. Расчет значений внутри матрицы:

   Значения остальных ячеек матрицы рассчитываются следующим образом (Листинг 1.3).

Листинг 1.3 – Расчет значений внутри матрицы

```
for i from 1 to n:
    for j from 1 to m:
        match = matrix[i-1][j-1] + (match_score if sequence1[i] ==
sequence2[j] else mismatch_penalty)
        delete = matrix[i-1][j] + gap_penalty
        insert = matrix[i][j-1] + gap_penalty
        matrix[i][j] = max(match, delete, insert)
```

   Где "sequence1[i]" и "sequence2[j]" – символы в позициях i и j соответственно.

4. Построение выравнивания:

   Построим выравнивание начиная с правой нижней ячейки матрицы ($i=n$,

j=m), продвигаясь вверх и влево по пути, который привел к текущему значению. Если текущее значение было получено из ячейки "matrix[i-1][j-1]", это означает совпадение (или несовпадение) символов, если из "matrix[i-1][j]", то это означает удаление, и если из "matrix[i][j-1]", то вставку.

5. Возврат оптимального выравнивания и показателя качества выравнивания: Оптимальное выравнивание будет состоять из символов, выбранных на каждом шаге во время построения выравнивания. Показатель качества выравнивания будет равен значению в правом нижнем углу матрицы ("matrix[n][m]").

Модель ошибки, то есть параметры, такие как "gap_penalty", "match_score" и "mismatch_penalty", определяются в зависимости от конкретной задачи.

Пример выравнивания, полученного с применением алгоритма Нидлмана-Вунша:

- Последовательность 1: `AGTACGCA`
- Последовательность 2: `AGCACACA`
- Пример выравнивания:
  ```
  AGTAC-GCA
  AG-CA-C-A
  ```

Алгоритм Смита-Уотермана (Smith-Waterman) – еще один метод динамического программирования для выравнивания последовательностей, который часто используется в биоинформатике для поиска локальных выравниваний между биологическими последовательностями, такими как ДНК. В отличие от алгоритма Нидлмана-Вунша, который находит глобальное выравнивание, алгоритм Смита-Уотермана находит локальные выравнивания, то есть выравнивания, которые начинаются и заканчиваются на любой позиции внутри последовательностей.

Описание шагов алгоритма Смита-Уотермана:

1. Инициализация матрицы:

   Создается матрица размером $(n+1)^{\times}(m+1)$, где n и m – длины выравниваемых последовательностей.

2. Инициализация крайних значений:

   Заполняются крайние значения матрицы, аналогично алгоритму Нидлмана-Вунша

3. Расчет значений внутри матрицы:

   Значения внутри матрицы рассчитываются по формуле (Листинг 1.4)

Листинг 1.4 – Расчет значений внутри матрицы

```
for i from 1 to n:
   for j from 1 to m:
      match = matrix[i-1][j-1] + (match_score if sequence1[i] == sequence2[j] else
mismatch_penalty)
      delete = matrix[i-1][j] + gap_penalty
      insert = matrix[i][j-1] + gap_penalty
      matrix[i][j] = max(0, match, delete, insert)
```

4. Построение выравнивания:

   Начиная с ячейки с максимальным значением в матрице, аналогично описанию в предыдущем алгоритме двигаясь по направлению влево и вверх, определяется путь в матрице, который привел к текущему значению. Построение выравнивания завершается, когда достигнута ячейка со значением 0.

5. Возврат оптимального выравнивания и показателя качества выравнивания:

   Оптимальное локальное выравнивание будет состоять из символов, выбранных на каждом шаге во время построения выравнивания. Показатель качества выравнивания будет равен значению в ячейке с максимальным скором.

Как и в предыдущем алгоритме, необходимо определить параметры, такие как gap_penalty, match_score и mismatch_penalty.

Пример выравнивания полученного с применением алгоритма Смита-Уотермана:

- Последовательность 1: `AGTACGCA`
- Последовательность 2: `TACGTTGCA`
- Пример выравнивания:

  `GTC`

  `GTC`

Алгоритмы Нидлмана-Вунша и Смита-Уотермана представляют собой алгоритмы точного выравнивания строковых последовательностей. Основное отличие состоит в том, что алгоритм Нидлмана-Вунша находит глобальное выравнивание, выравнивая последовательности целиком, в то время как алгоритм Смита-Уотермана идентифицирует локальные выравнивания внутри последовательностей. Данные алгоритмы неэффективно применять напрямую для решения задачи ресеквенирования из-за высокой вычислительной сложности, однако они служат основой для более сложных методов выравнивания. В этих более сложных методах на первом этапе обычно выполняется поиск потенциальных участков расположения рида в геноме, на которые на втором этапе выравниваются риды с помощью алгоритмов точного выравнивания. Двухэтапные алгоритмы используются в таких популярных биоинформатических инструментах выравнивания как Bowtie2[41], BWA[42] (Burrows-Wheeler Aligner), minimap2.

1.2.4 Подход seed-chain-align для быстрого выравнивания

Алгоритмы точного выравнивания достаточно требовательны к вычислительным ресурсам. Например, временную сложность (количество операций) и пространственную сложность (объем памяти) алгоритма Нидлмана-Вунша можно оценить $O(N \times M)$, где M и N – длины выравниваемых последовательностей. Сложность зависит от модели штрафа за разрыв и продолжение разрыва, однако в любом случае при выравнивании рида на

референсный геном с помощью алгоритмов точного выравнивания $N = 3\times10^9$ и сложность выравнивания одного рида составит $O(3\times10^9\times M)$

Для снижения сложности выравнивания ридов в биоинформатике применяется двухэтапный подход seed-chain-align. Подход seed-chain-align (или seed-chain-extend[43]) — это стратегия, используемая для эффективного выравнивания ридов на эталонный геном путем разбиения ридов на сравнительно небольшие подпоследовательности нуклеотидов (якори, seeds[44]), формирования цепочек из якорей и их последующего точного выравнивания с помощью алгоритмов точного выравнивания, например алгоритмов Нидлмана-Вунша или Смита-Уотермана. Подход seed-chain-align реализован в таких инструментах, как BWA и minimap2.

Рассмотрим основные этапы метода Seed-Chain-Align (Рисунок 1.3):

1. **Извлечение якорей из референса.** На данном этапе вычисляются все якоря для референсной последовательности.
2. **Индексация референса.** На данном этапе создается индекс референсной последовательности, вычисленные якоря и их позиции заносятся в специализированную структуру индекса (например, в хэш-таблицу, где ключом является хэш якоря, а значениями – все возможные позиции якоря).
3. **Извлечение якорей из ридов.** На данном этапе вычисляются якоря для ридов.
4. **Поиск якорей и фильтрация.** Выполняется поиск позиций якорей ридов в референсе и фильтрация по различным критериям, например, фильтрация наиболее распространенных якорей
5. **Составление цепочек якорей и предварительная фильтрация.** Для нахождения наиболее точного выравнивания вычисляется оценка оптимальной цепочки якорей. Алгоритм сначала идентифицирует все возможные цепочки совпадающих *k*-меров между последовательностями из прочтений и референсной последовательности, а затем вычисляет оценку для каждой цепочки на основе количества совпадающих *k*-меров и размера

разрывов между ними. После этого алгоритм выбирает цепочку с наибольшей оценкой в качестве основной.

6. **Выравнивание с помощью алгоритмов точного выравнивания.** На данном этапе проводится глобальное выравнивание между полученными последовательностями из якорей на референсной последовательности и якорей последовательности из рида. Точное выравнивание может проводиться, например, с помощью алгоритма Нидлмана-Вунша или алгоритма Смита-Ватермана.

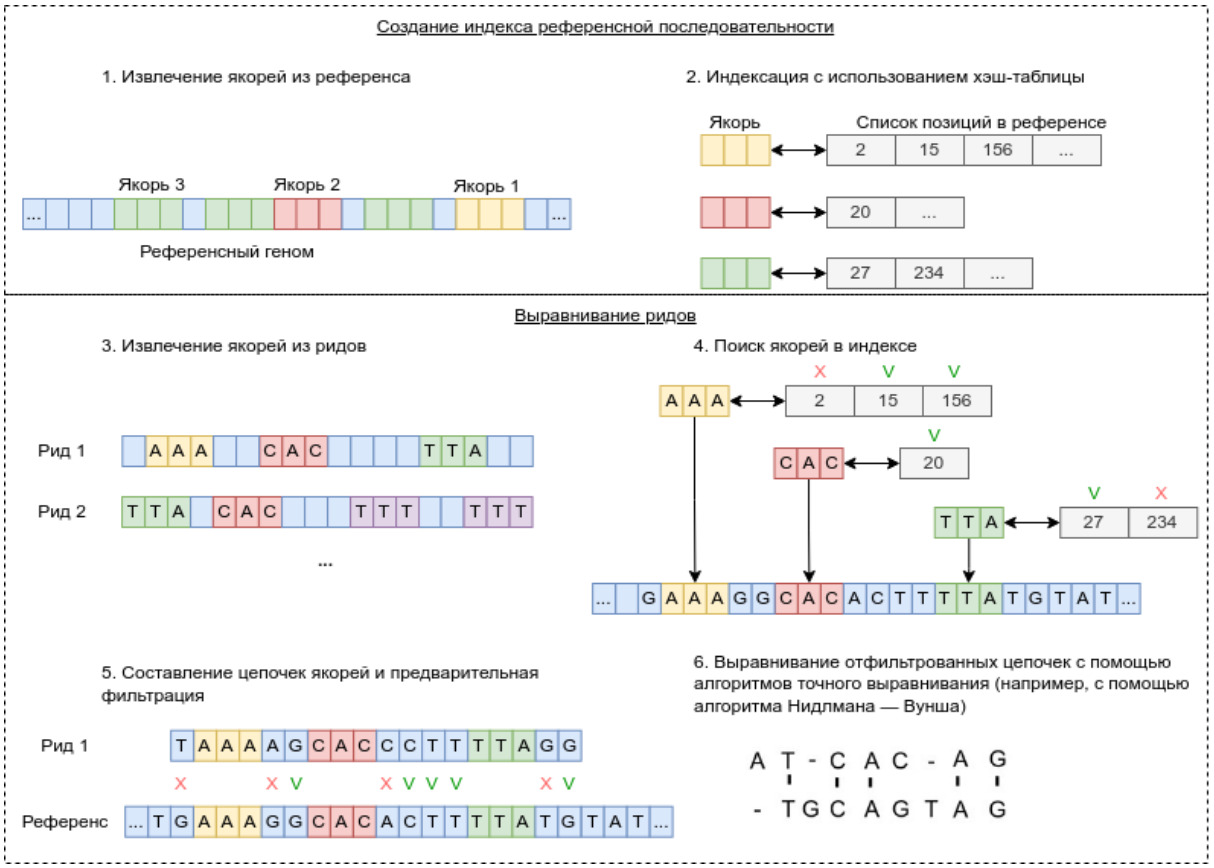

Рисунок 1.3 – Выравнивание ридов методом Seed-Chain-Align

Инструмент BWA использует для поиска якорей суффиксный массив, полученный с помощью преобразования Барроуза-Уилера (BWT[45]), и FM-индекс для ускорения поиска подстроки в суффиксном массиве. BWT представляет собой обратимую трансформацию, которая перестраивает последовательность эталонного генома для облегчения быстрого сопоставления якорей референса с якорями ридов секвенируемого образца. FM-индекс представляет из себя сжатое

представление суффиксного массива, полученного с помощью BWT, обеспечивающее эффективный поиск якорей и их позиций.

Инструмент minimap2 использует другую структуру индекса, известную как массив разреженных суффиксов (sampled suffix array[46], SSA). SSA — это разреженное представление массива суффиксов, представляющего собой структуру данных, используемую для эффективного поиска подстрок в строках. SSA позволяет minimap2 быстро находить начальные совпадения для выравнивания ридов.

И BWA, и minimap2 используют представление референсного генома в виде индексов, для того чтобы обеспечить быстрое сопоставление якорей и последующее формирование и выравнивание цепочек. Использование индексов для формирования цепочек якорей и последующего выравнивания цепочки позволяет повысить скорость при высокой точности выравнивания ридов.

1.2.5 Алгоритмы работы инструментов выравнивания ридов

Рассмотрим более подробно алгоритмы работы двух популярных инструментов выравнивания ридов: BWA и minimap2. Оба инструмента написаны на языке C и используют подход seed-chain-align для выравнивания ридов.

Как отмечено выше, в инструменте BWA для индексации используется алгоритм Барроуза-Уилера. Алгоритм Барроуза-Уилера (Burrows-Wheeler Transform, BWT) - это обратимый алгоритм сжатия текста, используемый в различных приложениях, включая сжатие данных и выравнивание последовательностей генома. Ниже представлены основные алгоритмы BWA:

1. Прямое преобразование BWT:

   - BWT получает на вход текст (или последовательность) T длиной *n* и преобразует его в новую последовательность BWT(T). переупорядочивания символы таким образом, что похожие символы группируются вместе. Ниже приведен листинг алгоритма (Листинг 1.5). Первый шаг BWT включает в себя построение суффиксного массива SA, который представляет все

циклические перестановки входного текста T, которые упорядочены лексикографически. SA[i] указывает на начальную позицию i-й наименьшей циклической подстроки.

Листинг 1.5 – Алгоритм BWT

```
function BWT (string T)
  create a list of all possible rotations of T
  let each rotation be one row in a large, square table
  sort the rows of the table alphabetically, treating each row as a string
  return the last (rightmost) column of the table
```

2. Построение FM-индекса для суффиксного массива SA

   FM-индекс (FM-Index, Full-text Minute-space Index) - это компактная структура данных, используемая для эффективного поиска подстрок в тексте. В основе FM-Index лежит специальная структура данных - дерево вейвлет-преобразования (Wavelet Tree), которое позволяет выполнять операции rank(x, i) (возвращает количество вхождений символа 'x' в SA до позиции i) и select(x, j) (возвращает позицию j-го вхождения символа 'x' в SA) за O(log σ) где σ - высота Wavelet Tree.

3. Обратное преобразование BWT

   BWT является обратимым преобразованием. Обратное преобразование Барроуза-Уилера (inverseBWT) используется для восстановления исходной строки из ее BWT-представления, например, в алгоритмах FM-индекса для выравнивания последовательностей и поиска подстрок в больших текстовых массивах.

В инструменте BWA подход seed-chain-align реализован следующим образом:

1. На первом шаге создается FM-индекс референсного генома. Индекс позволяет искать точные вхождения подстрок в референсный геном со сложностью O(log n), где n - длина референсного генома
2. Для рида создается набор якорей (подстрок рида длины k). Производится поиск их позиций в индексе.

3. После чего происходит расширение участков совпадения якорей и точное выравнивание.

Minimap2 — это универсальный инструмент для выравнивания ридов, который использует алгоритмов и методов для эффективного сопоставления коротких и длинных ридов с референсным геномом. Ключевым понятием инструмента minimap2 является минимизатор.

Минимизатор — это короткая подстрока длины *k*, которая является лексикографически минимальной строкой в окне *w* (Рисунок 1.4).

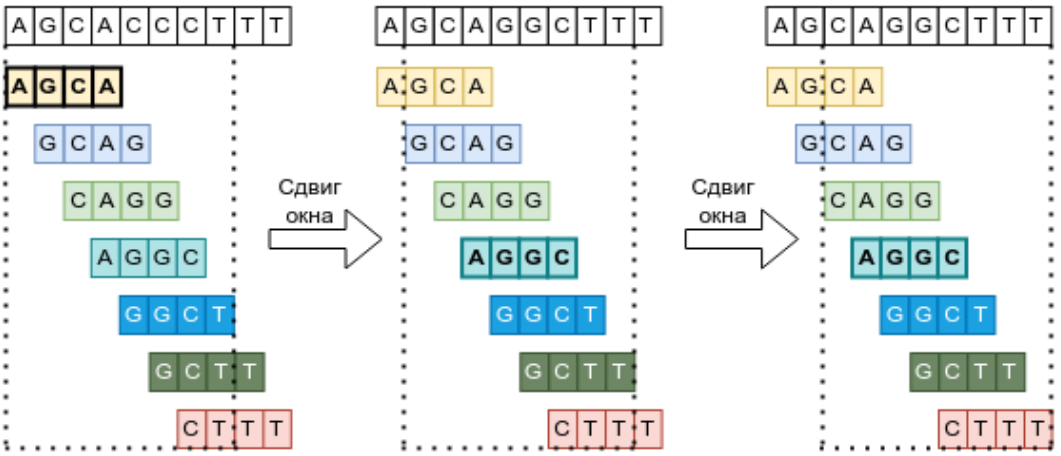

Рисунок 1.4 – Поиск минимизаторов длины *k*=4 в окне длины *w*=5.

В инструменте minimap2 подход seed-chain-align реализован следующим образом:

1. На первом шаге создается индекс референсного генома. Индекс представляет из себя хэш-таблицу минимизаторов и их позиций.
2. Для рида вычисляется набор минимизаторов. Производится поиск их позиций в индексе.
3. После чего происходит выстраивание цепочек минимизаторов, точное выравнивание и выбор наилучшего выравнивания.

1.2.6 Выравнивание на пангеномный граф

С развитием NGS секвенирования и появлением открытых баз данных, таких как проект "1000 геномов" появились такие методы выравнивания, как выравнивание на пангеномный[47] граф. Пангеномный граф – это структура данных, используемая для представления геномных данных, которая учитывает различия между индивидами (генетические варианты). Выравнивание на пангеномный граф – это метод анализа секвенированных данных, который использует пангеномный граф вместо линейного референса для более точного выравнивания и анализа геномных данных.

Преимущества выравнивания на пангеномный граф включают в себя:

1. Учет генетических вариантов.
2. Более точное выравнивание с учетом геномных различий.
3. Анализ неизвестных геномов. Пангеномные графы позволяют анализировать геномы нестандартных организмов или организмов с не полностью изученными геномами.

Однако выравнивание на пангеномный граф также является более сложным с вычислительной точки зрения и требует значительное количество памяти для создания и хранения пангеномного графа. Этот метод особенно полезен при анализе геномных данных с высокой изменчивостью, например, в медицинских исследованиях, а также при изучении популяционной генетики и эволюции.

В качестве примера отметим такие инструменты, как VG и minigraph. VG (Variation Graph) и minigraph – это два инструмента выравнивания ридов на пангеномный граф.

Основные особенности инструмента VG:

1. Внутреннее представление - пангеномный граф. Граф может включать геномные варианты различной сложности

2. Для выравнивания используется GCSA2 (Generalized Compressed Suffix Array) - индекс для направленных графов, основанный на преобразовании BWT
3. Доступен собственный механизм идентификации генетических вариантов

Основные особенности инструмента minigraph:

1. Внутреннее представление - упрощенный пангеномный граф на основе миниграфов. Миниграфы призваны упростить задачу построения графа, сохраняя при этом важную информацию.
2. Алгоритм выравнивания работает быстрее по сравнению с VG, однако сложные генетические варианты могут быть пропущены.
3. Для идентификации генетических вариантов используются только внешние инструменты.

Таким образом, и VG и minigraph являются инструментами для выравнивания ридов на пангеномный граф. Однако VG обеспечивает более комплексный подход к идентификации сложных генетических вариантов, в частности, позволяет идентифицировать варианты непосредственно на графе. Minigraph, с другой стороны, упрощает структуру графа и может быть более подходящим для прикладного использования, когда вычислительные ресурсы ограничены или в случае, когда достаточно упрощенного представления пангеномного графа.

В научной литературе[48] показано, что выравнивание на пангеномный граф позволяет улучшить качество анализа на регионах с генетическими вариантами, однако по скорости выравнивание на линейный референс превосходит выравнивание на пангеномный граф.

1.2.7 Оценка вычислительной сложности алгоритмов выравнивания коротких прочтений на линейный референсный геном и пангеномный граф

В данном разделе приводится сравнение оценок вычислительной эффективности алгоритмов выравнивания коротких прочтений методом seed-chain-align на линейный референсный геном и на пангеномный граф.

Выравнивание ридов методом seed-chain-align на линейный референсный геном:

1. Время (T) для выравнивания одного рида: $O(L)$ или $O(L \log G)$, где L - длина рида, G - длина референсного генома.

   Время выравнивания на линейный референсный геном зависит от длины рида (L) и, в некоторых случаях, от длины референсного генома (G).
2. Время (T) для выравнивания N ридов: $O(N * L)$ или $O(N * L \log G)$, где N - количество ридов.

Оценка вычислительной сложности алгоритма выравнивания ридов на пангеномный граф с использованием метода seed-chain-align является сложной задачей. Однако дадим общую оценку:

В общем случае, вычислительная сложность выравнивания ридов на пангеномный граф может быть выражена как $O(N * M * V)$, где:

- N — количество ридов.
- M — средняя длина рида.
- V является мерой сложности пангеномного графа (например, сумма количества узлов и ребер).

Эта оценка предполагает, что каждый рид обрабатывается независимо и что сложность алгоритма выравнивания линейно масштабируется в зависимости от количества ридов, длины ридов и сложности графа.

Это упрощенная оценка, фактическая сложность вычислений может варьироваться в зависимости от конкретного используемого алгоритма, его оптимизации и деталей структуры пангеномного графа.

Выравнивание на граф может занимать значительное время из-за ряда параметров:

1. Сложность графа: Пангеномный граф может иметь сложную структуру с большим количеством узлов и ребер, что увеличивает количество якорей, к которым необходимо применить метод seed-chain-align. Это увеличивает время выравнивания.
2. Поиск путей для составления цепочек якорей: в пангеномных графах необходимо исследовать множество возможных путей для выравнивания рида, особенно если есть множество альтернативных маршрутов, что может потребовать больше времени и вычислительных ресурсов.
3. Построение графа: Время, необходимое для построения пангеномного графа, зависит от числа геномов и их длины. Построение графа может быть вычислительно затратной операцией.

В целом, выравнивание на пангеномный граф является более ресурсоемкой задачей, чем выравнивание на линейный референсный геном. Однако это также позволяет учитывать генетическую изменчивость и обрабатывать регионы с большой вариабельностью более точно, что является важным в генетических исследованиях.

1.3 Возможности оценки качества анализа выходных данных секвенатора для задачи полногеномного секвенирования

Для того, чтобы утверждать что результаты анализа выходных данных секвенатора являются корректными необходимо оценить качество программного конвейера обработки выходных данных NGS секвенатора.В данном разделе будет рассмотрено два способа оценка качества анализа данных полногеномного секвенирования – анализ распределения значений метрики качества выравнивания коротких прочтений на референсный геном и сравнение VCF-файлов полученных в результате анализа FASTQ-файлов проекта "The genome in a bottle" с

эталонными VCF-файлами и результатами других участников соревнования "PrecisionFDA Truth Challenge".

1.3.1 Анализ распределения значений метрики качества выравнивания коротких прочтений

Для оценки качества выравнивания отдельного рида на референсный геном используется метрика – качество выравнивания (MAPQ[49], mapping quality). Из спецификации формата SAM[50]: качество выравнивания (MAPQ) равно значению выражения $-10\log_{10}P$ (позиция выравнивания неверна), округленному до ближайшего целого числа. Число 255 указывает, что значение не определено.

Таким образом, если известно, что вероятность правильной позиции выравнивания некоторого случайного короткого прочтения равна 0.99, то показатель MAPQ должен быть равен 20 (т. е. $\log_{10}(0.01)(-10)$). Если бы вероятность правильной позиции выравнивания увеличилась до 0.999, оценка MAPQ увеличилась бы до 30. Таким образом, чем выше оценка MAPQ, тем выше вероятность того, что рид выровнен в корректную позицию референсного генома. И наоборот, если вероятность правильного определения позиции выравнивания стремится к нулю, то же самое происходит и с показателем MAPQ.

Реализация расчета MAPQ в разных инструментах выравнивания осуществляется по-разному, максимальные значения MAPQ_MAX и значения штрафов за замены и разрывы в выравнивании значительно отличаются. Однако в рамках исследований одного инструмента модель расчета MAPQ остается неизменной что позволяет использовать данную метрики для сравнения качества работы инструмента при различных настройках. Значения штрафов в minimap2 могут быть настроены пользователем в зависимости от конкретной задачи выравнивания и характеристик данных. Например, при работе с высококачественными данными секвенирования, исследователь может использовать более строгие штрафы, чтобы получить более точные и надежные результаты выравнивания. С другой стороны, при работе с данными с низким

качеством или с повторяющимися участками генома, исследователь может использовать менее строгие штрафы для более гибкого выравнивания.

В инструменте minimap2 максимальное значение метрики MAPQ – 60. Качество выравнивания 60 говорит о том, что короткое прочтение выровнялось на определенную позицию референсного генома без замен и разрывов. Штрафы для MAPQ в minimap2 могут быть настроены с помощью различных параметров, таких как

- штрафы за несовпадения нуклеотидов (по умолчанию флаг B=4)
- штрафы за открытие разрывов (по умолчанию флаги $O1$=4; $O2$=24)
- штрафы за продолжение разрывов (по умолчанию флаги $E1$=2; $E2$=1)
- итоговая цена разрыва длины k равна по умолчанию $\min\{O1+k\times E1, O2+k\times E2\}$

Штрафы влияют на значение метрики MAPQ, то есть на вероятность определения правильной позиции короткого прочтения в референсном геноме. Меньшие значения метрики MAPQ указывают на более высокие штрафы за выравнивание, что предполагает потенциальную неоднозначность сопоставления или более низкую достоверность выравнивания. Более высокие значения MAPQ соответствуют более низким штрафам за выравнивание, что указывает на более высокую уверенность в точности выравнивания и снижение неоднозначности сопоставления. Если прочтение было выровнено в несколько мест с одинаковым MAPQ, то в SAM-файле MAPQ выставляется равным 0.

Соответственно, одним из способов сравнить несколько вариантов анализа данных NGS является анализ распределения значений MAPQ на одних и тех же данных, например, можно сравнить:

- количество идеально выровненных ридов (с MAPQ=MAPQ_MAX)
- количество точно выровненных ридов выровненных с MAPQ>MAPQ_MAX/2
- количество ридов выровненных с MAPQ>0

1.3.2 Данные проекта "The Genome in a Bottle"

Проект "Геном в бутылке"[51] (GIAB, The Genome in a Bottle) — это инициатива, координируемая Национальным институтом пищевых продуктов и медикаментов США (NIST) совместно с другими организациями, включая Национальные институты здравоохранения (NIH) и Управление по санитарному надзору за пищевыми продуктами и медикаментами США (FDA). Проект направлен на разработку справочных материалов и методов секвенирования генома, идентификации генетических вариантов и анализа геномных данных.

В рамках проекта собраны и охарактеризованы эталонные справочные образцы данных секвенирования ДНК человека. Эти справочные образцы являются ориентирами для оценки качества идентификации генетических вариантов и конвейеров анализа данных.

Эталонные образцы проходят различные этапы анализа и контроля качества. Так эталонные образцы отсеквенированы с использованием различных технологий и методов, таких как секвенирование NGS, секвенирование третьего поколения и генотипирование[52], после чего проанализированы набором популярных биоинформатических программных конвейеров для идентификации консенсусных данных по генетическим вариантам для этих эталонных образцов.

1.3.3 Проект "PrecisionFDA Truth Challenge"

Для исследования качества анализа выходных данных секвенирования важно иметь возможность сравнивать качество работы программного конвейера с существующими аналогами на эталонных данных. В 2015 году по инициативе Управления по санитарному надзору за качеством пищевых продуктов и медикаментов США (FDA), совместно с консорциумом GIAB, была реализована идея создания научно-исследовательского проекта "PrecisionFDA Truth Challenge", который позволит его участникам, членам научного сообщества, тестировать, запускать, апробировать как существующие, так и новые биоинформатические

подходы к обработке значительных объемов генетических данных NGS на данных образцов HG001 и HG002 с использованием референсного генома hg19. В 2020 году был запущен проект "PrecisionFDA Truth Challenge V2"[53], в котором был добавлен анализ высоко вариабельных регионов генома и анализ данных секвенирования третьего поколения на данных образцов HG003 и HG004 с использованием референсного генома hg38.

PrecisionFDA Truth Challenge предоставляет участникам эталонный набор данных, который включает в себя набор эталонных выходных данных секвенатора (ридов в формате FASTQ) вместе с соответствующими наборами эталонных ("истинных") генетических вариантов (файлы в формате VCF) из проекта GIAB. VCF-файлы содержат генетические варианты с их позициями идентифицированные несколькими разными способами, что служит отправной точкой для оценки точности методов идентификации генетических вариантов.

Для измерения качества работы программных конвейеров используются такие метрики[54], как точность (*precision*), полнота (*recall*) и *F*-мера (*F-score* или *F1*). Конкретные детали, контрольные показатели и показатели оценки могут различаться в разных итерациях PrecisionFDA Truth Challenge. Актуальная информация о цели, контрольных показателях и критериях оценки для каждого конкретного испытания содержатся в описании на официальном веб-сайте PrecisionFDA[55] и документации по испытаниям.

1.3.4 Инструмент сравнения VCF-файлов hap.py

Для сравнения VCF-файлов и получения метрик (точность, полнота, *F*-мера) в PrecisionFDA Truth Challenge рекомендуется использовать инструмент hap.py.

Hap.py – это биоинформатический инструмент, разработанный для оценки качества идентификации генетических вариантов (SNP, indel) в данных секвенирования. Hap.py предоставляет средства для сравнения и оценки результатов идентификации генетических вариантов, полученных с помощью

различных алгоритмов или программных конвейеров, а также сравнения вариантов с известными истинными вариантами (ground truth).

Некоторые ключевые особенности и функции Hap.py:

1. Вычисление метрик, таких как точность (*Precision*), полнота (*Recall*), *F*-мера (*F-score* или *F1*) для оценки качества идентификации генетических вариантов.
2. Интерактивный отчет: после анализа Hap.py создает интерактивный отчет, который позволяет исследователям подробно изучать результаты и метрики оценки.
3. Поддержка разных форматов данных: Hap.py поддерживает различные форматы данных, такие как VCF (Variant Call Format), BED[56] (Browser Extensible Data) и другие.

Инструмент Hap.py полезен для сравнения программных конвейеров через вычисление ряда метрик (точности, полноты и F-меры) идентификации генетических вариантов в биоинформатических исследованиях.

1.4 Актуальность создания биоинформатических алгоритмов и программных конвейеров

На начальном этапе биоинформатических исследований зачастую требования к ресурсам достаточно невелики, что позволяет использовать для вычислений сервер отдельной исследовательской лаборатории. По мере того, как проект развивается и количество NGS данных в проекте продолжает расти, ресурсов, принадлежащих одной лаборатории или учреждению, становится недостаточно. Исследователи должны искать другие способы выполнения необходимых вычислений и хранения данных.

В данном разделе приведено описание программного конвейера анализа данных полногеномного секвенирования и выполнен обзор технологий подходов, позволяющих реализовать масштабируемые и воспроизводимые программные конвейеры.

1.4.1 Программный конвейер анализа данных полногеномного секвенирования

Для того чтобы корректно преобразовать выходные данные секвенатора в формате FASTQ к файлу с набором генетических вариантов образца в VCF-формате, необходимо выполнить ряд последовательных этапов вычислительного анализа. Разные подходы предполагают наличие различного количества этапов анализа и разный набор конкретных инструментов, реализующих эти этапы.

Одним из широко распространенных программных конвейеров для анализа данных полногеномного секвенирования является биоинформатический программный конвейер "Germline short variant discovery" разработанную в Институте Броуда (Broad Institute). Институт Броуда известен своим вкладом в исследования геномики и разработку передовых методов анализа выходных данных секвенирования, широко применяемых на практике.

Основные шаги конвейера анализа данных WGS от Института Броуда, приведены ниже. Обозначим целевой геном как *G*, эталонный геном как *R*. Цель программного конвейера состоит в том, чтобы идентифицировать однонуклеотидные полиморфизмы (SNP) и структурные генетические варианты (вставки, делеции, инверсии) в целевом геноме *G* по сравнению с эталонным (референсным) геномом *R*. В листинге алгоритма: RAW_FASTQ – выходные данные секвенатора, PROCESSED_FASTQ – обработанные выходные данные секвенатора, BAM – выровненные риды, RECALIBRATED_BAM – обработанные выровненные риды, VARIANTS – генетические варианты.

Листинг 1.6 – Алгоритм программного конвейера полногеномного секвенирования

```
QC(RAW_FASTQ) -> PROCESSED_FASTQ // Контроль качества и фильтрация
Align(PROCESSED_FASTQ, R) -> BAM // Выравнивание ридов
BQSR(BAM) -> RECALIBRATED_BAM // Повторная калибровка базовых
показателей качества
CallVariants(RECALIBRATED_BAM, R) -> VARIANTS // Идентификация генетических
вариантов
```

Основные шаги конвейера (Листинг 1.6):

1. Предварительная обработка данных:
    a. Контроль качества (quality control[57]) и фильтрация (тримминг, trimming). Проверка качества необработанных данных секвенирования RAW_FASTQ с помощью таких инструментов, как FastQC. Низкокачественные риды и адаптерные последовательности обрезаются или отфильтровываются с помощью таких инструментов, как Trimmomatic или Cutadapt.
    b. Выравнивание ридов. Риды, полученные после фильтрации PROCESSED_FASTQ, выравниваются на референсный геном R с использованием биоинформатических инструментов выравнивания, таких как BWA-MEM, minimap2 или Bowtie2. На этом шаге создается файл в формате Binary Alignment Map (BAM), в котором хранится информация о последовательности нуклеотидов рида, его позиции,

качестве выравнивания (mapping quality, MAPQ), а также строка CIGAR (Compact Idiosyncratic Gapped Alignment Report), которая описывает, как рид выровнялся на референсный геном (совпадения, замены, инделы).

2. Обработка данных и идентификация генетических вариантов:
    a. Повторная калибровка базовых показателей качества (Base Quality Score Recalibration[58], BQSR): BQSR применяется для повторной калибровки показателей качества прочтения нуклеотидов, что помогает исправить систематические ошибки секвенатора. Обычно используются инструменты GATK BaseRecalibrator и ApplyBQSR, результатом является файл RECALIBRATED_BAM.
    b. Идентификация генетических вариантов: SNP и короткие инделы идентифицируются с использованием таких инструментов, как HaplotypeCaller[59] GATK или Samtools mpileup. Эти инструменты используют статистические модели для идентификации генетических вариантов на основе данных выравнивания ридов. В результате выполнения этого этапа формируется файл VARIANTS.
3. Вычисление метрик качества и аннотирование генетических вариантов:
    a. Помимо этапа первичного контроля качества входных данных для оценки корректности данных необходимо также исследовать качество анализа на различных этапах работы программного конвейера. Для оценки качества выравнивания основными метриками являются глубина и ширина покрытия:
        i. Глубина покрытия (Coverage Depth[60]) – количество ридов, покрывающих определенную позицию в геноме. Глубина покрытия определяет, сколько раз определенная позиция была прочитана в рамках данного эксперимента. Например, если определенная позиция имеет глубину покрытия 30X, это означает, что она была прочитана 30 раз.

ii. Ширина покрытия (Coverage Breadth) – процент генома, который был покрыт ридами с определенной глубиной. Например, если геном состоит из 1000 позиций, и 900 из них были покрыты прочтениями глубиной 10X, то ширина покрытия 10X составляет 90%.

Для оценки качества идентификации генетических вариантов необходимо провести сравнение с эталонным VCF-файлом.

b. Аннотирование (Фильтрованные варианты, Базы данных) –> Аннотированные варианты

Аннотирование: варианты аннотируются функциональной информацией с использованием таких баз данных, как dbSNP[61], ClinVar[62], и алгоритмов функционального прогнозирования, таких как SIFT[63] или PolyPhen[64].

Важно отметить, что параметры инструментов программного конвейера могут быть настроены в зависимости от конкретных требований исследования, доступных ресурсов и достижений в технологиях секвенирования. Передовой опыт Института Броуда обеспечивает основу для обработки и анализа данных WGS, но отдельные исследователи или учреждения могут адаптировать или модифицировать программный конвейер в зависимости от своих конкретных потребностей и опыта.

1.4.2 Инструменты для анализа данных полногеномного секвенирования

Ранее были рассмотрены инструменты bwa и minimap2, которые использовались для этапа выравнивания ридов на референсный геном.

Для выполнения повторной калибровки базовых показателей качества используется инструмент BQSR фреймворка GATK.

Показатели качества прочтения нуклеотидов, присвоенные полученные от секвенаторов, могут быть неточными или смещенными. Это связано с различными

факторами, включая различия в химическом составе наборов для секвенирования, калибровке секвенатора и других систематических ошибок. BQSR стремится исправить эти неточности.

BQSR использует статистические модели для выявления и исправления систематических ошибок в показателях качества прочтения нуклеотидов. Он учитывает несколько ковариат или факторов, которые могут повлиять на качество прочтения нуклеотидов. Эти ковариаты могут включать, среди прочего, цикл секвенирования, контекст и положение в считывании.

Процесс повторной калибровки происходит в два этапа:

1. Построение модели. На этом этапе BQSR исследует набор известных генетических вариантов вариантов (переданный на вход инструменту) и сравнивает качество прочтения нуклеотидов с ожидаемыми значениями. Происходит вычисление значений ковариат с учетом оценки ошибки на известных вариантах.
2. Применение модели. После построения модели BQSR применяет ее для повторной калибровки показателей качества для всех нуклеотидов в данных секвенирования.

Корректируя показатели качества прочтения нуклеотидов, BQSR повышает чувствительность и специфичность алгоритмов вызова вариантов, уменьшая как ложноположительные, так и ложноотрицательные результаты.

Для идентификации генетических вариантов также применяется ряд инструментов, одни из самых популярных это HaplotypeCaller и Deepvariant.

Инструмент HaplotypeCaller из фреймворка GATK для идентификации генетических вариантов использует алгоритм локальной пересборки генома, он создает из ридов в небольшом регионе набор связанных участков ДНК, что позволяет не только определять однонуклеотидные замены и инделы, но и фильтровать ошибки выравнивания. HaplotypeCaller обладает высокой точностью идентификации генетических вариантов, особенно на сложных участках генома.

DeepVariant — это инструмент идентификации генетических вариантов, разработанный Google и использующий методы глубокого обучения, в частности, глубокие сверточные нейронные сети (CNN), для идентификации генетических вариантов. Модель глубокого обучения способна распознавать сложные закономерности идентификации вариантов, однако для ее работы желательно наличие GPU. Кроме того, качество работы нейронной сети сильно зависит от данных на которых происходило обучение.

Выбор между HaplotypeCaller и DeepVariant зависит от конкретных целей анализа, типа имеющихся данных секвенирования и доступных вычислительных ресурсов.

1.4.3 Язык WDL для конвейерной разработки и его преимущества

Для того чтобы собрать отдельные инструменты в единый программный конвейер и потом иметь возможность развивать его необходимо использовать средства описания последовательности выполнения программных конвейеров и их последующего запуска.

WDL (Workflow Description Language, язык описания рабочих процессов или программных конвейеров) — это язык программирования, специально разработанный для разработки и выполнения сложных научных рабочих процессов, особенно в области геномики и биоинформатики. Он обеспечивает стандартизированный и воспроизводимый способ описания и выполнения вычислительных конвейеров. Вот некоторые преимущества использования WDL для конвейерной разработки:

- Модульность и возможность повторного использования: WDL позволяет разработчикам разбивать сложные рабочие процессы на модульные задачи или подпроцессы. Эти модульные компоненты можно легко повторно использовать и комбинировать для создания более крупных и сложных конвейеров. Это способствует модульности кода, уменьшает избыточность и повышает удобство сопровождения кода.

- Воспроизводимость и переносимость: WDL обеспечивает воспроизводимость за счет явного определения входных данных, программных зависимостей и шагов, которые необходимо выполнить. Это делает конвейеры переносимыми между различными вычислительными средами, пока доступны необходимые зависимости. Это обеспечивает согласованные и надежные результаты независимо от среды выполнения.
- Масштабируемость и производительность. WDL позволяет разработчикам распараллеливать задачи внутри конвейера, используя преимущества распределенных вычислительных ресурсов. Он поддерживает указание входных и выходных зависимостей между задачами, что позволяет эффективно использовать ресурсы и повышать общую производительность конвейера.
- Независимость от языка: WDL спроектирован так, чтобы быть независимым от языка, что означает, что его можно использовать с различными языками программирования и инструментами. Эта гибкость позволяет разработчикам выбирать наиболее подходящий язык или инструмент для каждого шага конвейера, используя существующие библиотеки и платформы, а также пользуясь преимуществами структурированного и стандартизированного характера WDL.
- Контроль версий и совместная работа. Конвейеры WDL обычно определяются как удобочитаемые текстовые файлы, что делает их подходящими для систем контроля версий, таких как Git. Это обеспечивает совместную разработку, легкое отслеживание изменений и возможность возврата к предыдущим версиям при необходимости.
- Независимость от инструментов и платформ: WDL не привязан к какому-либо конкретному механизму выполнения рабочих процессов или вычислительной платформе. Его можно выполнять с использованием различных систем управления рабочими процессами, облачных платформ и высокопроизводительных вычислительных сред. Эта гибкость позволяет

разработчикам выбирать наиболее подходящую среду выполнения в зависимости от их конкретных потребностей и доступных ресурсов.

- Поддержка сообщества и экосистема инструментов: WDL имеет растущее сообщество пользователей и разработчиков, что приводит к активной экосистеме инструментов, библиотек и ресурсов. Сюда входят механизмы выполнения рабочих процессов, инструменты визуализации и среды разработки, поддерживающие WDL, что обеспечивает надежную экосистему для разработки и выполнения конвейеров.

Используя WDL для конвейерной разработки, исследователи и разработчики могут извлечь выгоду из улучшенной модульности, воспроизводимости, переносимости, масштабируемости и совместной работы. Эти преимущества способствуют эффективной и надежной разработке конвейеров, что делает WDL ценным языком для проектирования и выполнения сложных научных рабочих процессов.

WDL и Nextflow являются популярным выбором для разработки и выполнения научных рабочих процессов, особенно в области геномики и биоинформатики. Несмотря на то, что они имеют сходство, обеспечивая создание воспроизводимых рабочих процессов, между WDL и Nextflow есть некоторые ключевые различия:

1. Синтаксис и дизайн языка

WDL использует структурированный синтаксис, вдохновленный такими языками программирования, как Python и JSON. Он фокусируется на декларативном подходе, позволяя пользователям четко и структурировано определять задачи, входные и выходные данные.

Nextflow использует предметно-ориентированный язык (DSL, Domain-Specific Language), специально разработанный для определения рабочих процессов, управляемых данными. Он обеспечивает более краткий и выразительный синтаксис с акцентом на состав конвейера и поток данных.

2. Инструментальная и языковая поддержка:

WDL не зависит от языка, что означает, что его можно использовать с разными языками программирования и инструментами. Это позволяет пользователям выбирать наиболее подходящий язык или инструмент для каждого шага конвейера.

Nextflow основан на языке программирования Groovy[65] и обеспечивает встроенную поддержку сценариев Bash, языка Python[66] и других популярных языков программирования. Он имеет встроенную поддержку популярных инструментов и библиотек биоинформатики.

3. Модель выполнения рабочего процесса:

Рабочие процессы WDL могут выполняться с использованием различных механизмов выполнения рабочих процессов, таких как Cromwell или Broad's FireCloud[67]. Эти механизмы управляют выполнением задач, распределяют ресурсы и обеспечивают поддержку распределенных вычислительных сред.

Nextflow имеет собственный встроенный механизм выполнения, который обрабатывает выполнение задач, управляет ресурсами и обеспечивает поддержку параллельных и распределенных вычислений. Он также предлагает бесшовную интеграцию с популярными планировщиками кластеров и облачными платформами.

4. Параллелизм данных и отказоустойчивость:

WDL поддерживает параллелизм данных, позволяя пользователям явно определять параллельное выполнение задач. Однако встроенные функции отказоустойчивости ограничены, а восстановление после сбоев обычно зависит от базового механизма выполнения рабочих процессов.

Nextflow отлично справляется с управлением параллелизмом данных и отказоустойчивостью. Он обеспечивает автоматическое разбиение данных, параллельное выполнение задач со встроенной поддержкой отказоустойчивости и восстановления в случае сбоев или ошибок.

5. Сообщество и экосистема:

WDL имеет растущее сообщество пользователей и разработчиков, особенно в области геномики и биоинформатики. Он получает поддержку и ресурсы, предоставляемые такими организациями, как Глобальный альянс геномики и здоровья (GA4GH[68]).

У Nextflow активное сообщество, уделяющее особое внимание биоинформатике и приложениям, требующим обработки больших объемов данных. Он имеет богатую экосистему плагинов, конвейеров, созданных сообществом, и обширную документацию.

Оба языка предоставляют мощные возможности для разработки воспроизводимых рабочих процессов, и у них есть активные сообщества, поддерживающие их разработку.

1.4.4 Системы управления программными конвейерами

Наиболее популярными системами управления программными конвейерами в биоинформатике являются Nextflow[69] и Cromwell[70]. Nextflow и Cromwell — это системы управления рабочими процессами, предназначенные для выполнения сложных научных программных конвейеров в разных вычислительных средах.

Nextflow — это система управления рабочими процессами с открытым исходным кодом, предназначенная для создания и выполнения рабочих процессов с интенсивным использованием данных переносимым и воспроизводимым способом. Nextflow использует предметно-ориентированный язык, который разработан с расчетом на то, чтобы быть простым и интуитивно понятным, но при этом достаточно мощным для выражения сложных рабочих процессов. Nextflow может запускать рабочие процессы в различных вычислительных средах, включая локальные рабочие станции, кластеры и облачные среды.

Cromwell, в свою очередь, представляет собой систему управления рабочими процессами, изначально специально разработанную для обеспечения выполнения рабочих процессов на облачной платформе Google. Cromwell использует язык описания рабочих процессов (WDL) для определения рабочих

процессов и поддерживает ряд режимов выполнения, включая локальное выполнение, выполнение в собственной облачной среде и выполнение в сторонних облачных провайдерах, таких как Amazon Web Services и Google.

И Nextflow, и Cromwell являются мощными системами управления рабочими процессами, между ними есть некоторые ключевые различия. Nextflow обладает высокой переносимостью и может запускать рабочие процессы на различных вычислительных инфраструктурах, а Cromwell оптимизирован для выполнения на облачной платформе Google. Кроме того, Nextflow предоставляет более широкий спектр встроенных инструментов и библиотек, в то время как Cromwell больше ориентирован на поддержку рабочих процессов, определенных с помощью WDL. В конечном итоге выбор между Nextflow и Cromwell будет зависеть от конкретных потребностей и требований пользователя, а также доступной вычислительной инфраструктуры.

1.4.5 Роль контейнеризации в создании воспроизводимых программных конвейеров

Контейнеризация и версионирование играют критически важную роль при разработке воспроизводимых программных конвейеров в области биоинформатики и анализа данных NGS.

Контейнеризация:

- **Изолированная среда:** все зависимости, библиотеки и инструменты, необходимые для работы программного конвейера упакованы в единый контейнер. Это обеспечивает изолированную среду, гарантирующую, что ваш код будет работать одинаково на разных операционных системах, без проблем совместимости.
- **Воспроизводимость:** Контейнер позволяет создать воспроизводимую среду, которая включает в себя все необходимое для запуска программного конвейера. Это гарантирует, что результаты будут воспроизводимыми независимо от времени и места выполнения.

- **Управление зависимостями:** все зависимости и версии программных компонентов определены внутри контейнера, что позволяет избежать проблем с совместимостью и конфликтами между версиями.

Версионирование:

- **История и отслеживание изменений:** Версионирование позволяет вам следить за изменениями в вашем программном конвейере. Вы можете точно знать,кем и какие изменения были внесены.
- **Восстановление:** Если что-то идет не так или результаты отличаются от ожидаемых, можно вернуться к предыдущей версии программного конвейера, чтобы проверить разницу и найти причины.
- **Коллаборация:** версионирование облегчает совместную разработку, позволяя каждому разработчику вносить изменения в отдельной ветке и объединять их в главной ветке по мере готовности.

Все это существенно улучшает управляемость, надежность и воспроизводимость программного конвейера в области биоинформатики и анализа данных NGS.

1.5 Применение облачных вычислений в биоинформатике

С успехом NGS совпала еще одна история технологического успеха: облачные вычисления. Хотя облачные вычисления не были изобретены с учетом научных интересов — основными коммерческими клиентами облачных вычислений являются технологические компании и другие предприятия — они все чаще играют ключевую роль в качестве базовой технологии для многих научных и инженерных усилий.

В частности, в геномике и биоинформатике облачные вычисления играют важную роль в двух задачах. Первая включает в себя повторный анализ наборов данных, содержащихся в общедоступных открытых архивах данных секвенирования. Вторая задача, для которой могут быть применены облачные

вычисления — это организация совместной работы над большими объемами общих данных. Распределенный характер облака сделал его подходящим для совместных и распределенных вычислений, а также для облегчения совместной работы. В качестве примеров можно привести ICGC[71] и связанный с ней PanCancer Analysis of Whole Genomes (PCAWG[72]), Cancer Genomics Cloud (CGC[73]) и Энциклопедию элементов ДНК (ENCODE[74]).

Облачные вычисления — это способ организации вычислительных ресурсов, позволяющий пользователям прозрачно арендовать их по запросу. Для научных пользователей облачные технологии имеют два основных преимущества: воспроизводимость и глобальный доступ (повторный анализ крупномасштабных архивных наборов данных и сотрудничество в области геномики).

1.5.1 Введение в кластерные вычисления SLURM и их актуальность для обработки данных NGS

SLURM (Simple Linux Utility for Resource Management) — это популярный менеджер рабочих нагрузок и планировщик задач, широко используемый в средах высокопроизводительных вычислений (HPC[75]). Он обеспечивает эффективное управление и распределение вычислительных ресурсов, таких как процессоры, память и графические процессоры, в кластере для выполнения крупномасштабных вычислительных задач. SLURM очень актуален для обработки данных секвенирования следующего поколения (NGS) по следующим причинам:

Масштабируемость: обработка данных NGS включает вычислительные задачи, требующие значительных вычислительных ресурсов. SLURM обеспечивает эффективное масштабирование, распределяя эти задачи между несколькими узлами в кластере. Он предоставляет механизмы для отправки задач, планирования и распределения ресурсов, обеспечивая оптимальное использование ресурсов кластера и обеспечивая высокопроизводительный анализ данных NGS.

Распараллеливание: многие рабочие процессы обработки данных NGS можно распараллелить для ускорения вычислений. SLURM облегчает

параллельное выполнение задач, позволяя пользователям определять и выделять несколько ядер или узлов для задач. Такое распараллеливание повышает эффективность и скорость обработки данных NGS, поскольку задачи могут выполняться одновременно на распределенных ресурсах.

Приоритизация задач и планирование: SLURM предоставляет гибкую систему планирования, которая позволяет пользователям расставлять приоритеты и планировать задач на основе собственных вычислительных требований. Это особенно полезно при обработке данных NGS, где разные этапы анализа или программные конвейеры могут иметь разные потребности в ресурсах или зависимости.

Управление ресурсами: SLURM эффективно управляет и отслеживает использование ресурсов в кластере. Он предоставляет функции для отслеживания хода выполнения задач, оценки потребления ресурсов и обеспечения соблюдения ограничений ресурсов. Это помогает предотвратить конкуренцию за ресурсы, обеспечивает эффективное использование ресурсов кластера и обеспечивает эффективное управление ресурсоемкими рабочими процессами обработки данных NGS.

Управление зависимостями задач: обработка данных NGS часто включает несколько последовательных шагов или зависимостей. SLURM позволяет пользователям определять зависимости между задачами, гарантируя, что каждый шаг будет выполняться только после успешного завершения его зависимостей. Эта функция управления зависимостями обеспечивает правильное выполнение рабочего процесса, предотвращает повреждение или несоответствие данных и облегчает анализ сложных конвейеров NGS.

Отказоустойчивость и восстановление задач: SLURM включает в себя механизмы отказоустойчивости для обработки сбоев выполнения задач. Он может автоматически перепланировать невыполненные или прекращенные задачи, обеспечивая непрерывность обработки данных NGS и сводя к минимуму влияние сбоев на работу кластера.

Таким образом, использование SLURM для обработки данных NGS обеспечивает эффективное управление ресурсами, планирование задач, распараллеливание и масштабируемость в средах кластерных вычислений. Планировщик SLURM позволяет выполнять ресурсоемкие рабочие процессы программных конвейеров обработки данных NGS, ускоряет анализ данных и повышает общую эффективность и производительность при запуске разнородных программных конвейеров обработки данных NGS.

1.6 Актуальность создания новых методов для обработки выходных данных секвенатора

Анализ больших данных (big data[76]) секвенирования нового поколения имеет огромное значение в медицине[77] и дает возможность расширить понимание природы заболеваний, разработать более точные методы диагностики и персонализированного лечения. NGS позволяет анализировать геном человека на предмет наличия генетических вариантов, связанных с генетическими заболеваниями, идентифицировать генетические варианты, ассоциированные с различными видами рака, создавать индивидуализированные планы лечения и исследовать новые молекулярные мишени для лекарств. Также анализ данных NGS используется для мониторинга динамики заболеваний, исследований состава и функций микробиома, определения генетических факторов, влияющих на метаболизм лекарств, и диагностики инфекций. Кроме того, NGS позволяет изучать эпигенетические изменения, которые играют роль в регуляции генов и развитии болезней. Эти примеры демонстрируют, что NGS и анализ данных NGS играют ключевую роль в современной биомедицине, способствуя более точной диагностике[78], лечению и исследованию заболеваний.

Создание новых методов обработки выходных данных секвенаторов нового поколения актуально и важно по нескольким причинам:

1. Улучшение качества данных: секвенаторы на базе технологии NGS генерируют огромные объемы данных с присущими им ошибками и погрешностями. Разработка новых методов обработки помогает улучшить качество данных за счет устранения ошибок секвенирования, уменьшения шума и повышения точности и надежности последующих анализов.
2. Точность идентификации генетических вариантов: точная идентификация вариантов имеет решающее значение для выявления генетических вариантов и их связи с заболеваниями. Новые методы обработки могут улучшить алгоритмы идентификации вариантов, что приведет к повышению

чувствительности и специфичности при идентификации однонуклеотидных полиморфизмов (SNP) и структурных генетических вариантов.

3. Вычислительная эффективность: анализ данных NGS требует больших вычислительных ресурсов, и объем данных продолжает расти в геометрической прогрессии. Разработка новых методов обработки, оптимизирующих вычислительную эффективность, может сократить время анализа, использование ресурсов и затраты, сделав геномные исследования и клинические приложения более доступными.
4. Разработка новых алгоритмов: обработка данных NGS включает сложные алгоритмы для выравнивания чтения, сборки de novo, идентификации генетических вариантов и интеграции данных. Создание новых методов открывает возможности для совершенствования алгоритмов, позволяя использовать инновационные подходы для решения уникальных задач и повышения точности и масштабируемости анализа данных NGS.
5. Интеграция и интерпретация данных. Интеграция данных NGS с другими омиксными данными, клинической информацией и внешними базами данных имеет решающее значение для комплексной интерпретации данных. Новые методы обработки могут обеспечить эффективную интеграцию, гармонизацию и интерпретацию данных, облегчая многомерный анализ и понимание сложных биологических систем.
6. Использование новых технологий секвенирования. По мере развития технологий секвенирования требуются новые методы для адаптации и оптимизации обработки данных для новых платформ. Методы, которые могут эффективно работать с секвенированием с длинным считыванием, секвенированием отдельных клеток, пространственной транскриптомикой и другими достижениями в технологиях NGS, имеют решающее значение для использования всего потенциала этих технологий.
7. Стандартизация и воспроизводимость. Разработка стандартизированных методов обработки помогает установить лучшие практики и рекомендации

для анализа данных, способствуя воспроизводимости и сопоставимости между различными исследованиями и исследовательскими группами. Это расширяет сотрудничество, облегчает обмен данными и обеспечивает надежность и целостность исследований на основе NGS.

Таким образом, создание новых методов обработки выходных данных NGS имеет важное значение для улучшения качества данных, повышения точности идентификации генетических вариантов, оптимизации эффективности вычислений, обеспечения алгоритмических улучшений, облегчения интеграции данных, обращения к новым технологиям и содействия стандартизации и воспроизводимости. Эти достижения способствуют развитию геномных исследований, персонализированной медицины и нашего понимания сложных биологических систем.

1.7 Выводы к первой главе

В первой главе произведен обзор ключевых аспектов и понятий, связанных с обработкой данных полногеномного секвенирования человека. Рассмотрены существующие технологии секвенирования: секвенирование по Сенгеру, NGS и секвенирование третьего поколения, история развития технологий секвенирования и их особенности.

Определено понятие референсного генома, описана роль референсного генома в решении задачи ресеквенирования. Описаны наиболее популярные на данный момент референсные геномы hg19 и hg38, описаны их различия и особенности. Для анализа данных NGS широко применяется метод выравнивания данных на референсный геном. В первой главе перечислены преимущества и недостатки ресеквенирования перед методом de novo сборки генома.

Описаны форматы данных, используемые для представления выходных данных секвенатора и хранения генетических вариантов. Рассмотрены методы точного выравнивания генетических последовательностей и двухэтапный подход

seed-chain-align для ускоренного выравнивания ридов. Рассмотрены инструменты, применяемые для анализа данных полногеномного секвенирования.

Также в первой главе описаны особенности и сложности анализа выходных данных секвенатора:

1. **Объем данных NGS.** В процессе секвенирования нового поколения возникают огромные объемы выходных данных секвенатора (до нескольких сотен гигабайт на один образец). Для обработки данных NGS на потоке требуется создание масштабируемых программных конвейеров. Эффективная обработка такого объема данных требует значительных вычислительных ресурсов и применения оптимизированных алгоритмов.
2. **Выравнивание на референсный геном.** Одной из первых и важных задач анализа NGS данных является выравнивание ридов на референсный геном. Качество этого выравнивания[79] имеет прямое влияние на точность всех последующих этапов анализа, таких как интерпретация генетических вариантов.
3. **Биологическая релевантность.** Точность и надежность анализа NGS данных[80] имеют критическое значение для идентификации генетических вариантов, связанных с различными характеристиками индивида (болезнями, поведению и другими). Даже небольшие ошибки на этапе выравнивания могут привести к неверной интерпретации результатов[81] и упущению значимых генетических вариантов.
4. **Разнообразие приложений.** NGS используется в различных областях, включая медицинскую диагностику, исследования популяционной генетики, агробиотехнологии[82] и другие. Каждая из этих областей требует специфического подхода к анализу данных, что подчеркивает важность разносторонней и универсальной методологии.
5. **Точность и воспроизводимость.** Помимо того, что методы анализа NGS данных должны обеспечивать необходимую точность исследования, они также должны быть воспроизводимыми[83]. Это особенно важно в

клинических исследованиях[84], где принимаются решения, влияющие на здоровье пациентов.

Создание биоинформатического программного конвейера – это долгосрочный и итеративный процесс, который требует внимания к множеству деталей, начиная от выбора инструментов и заканчивая оптимизацией производительности. Для того чтобы обеспечить системный подход к решению различных задач биоинформатического анализа данных, необходимо разработать метод построения воспроизводимых программных конвейеров обработки выходных данных секвенаторов. Кроме того, что разрабатываемый программный конвейер должен обладать свойствами масштабируемости и воспроизводимости, он также должен удовлетворять ограничениям на значения метрик качества и вычислительные ресурсы, продиктованным спецификой решаемой задачи.

Для достижения цели настоящей диссертации, разработанные методы и программные средства для обработки данных полногеномного секвенирования человека должны превосходить существующие методы по качеству выявления однонуклеотидных полиморфизмов.

Для достижения поставленной цели во второй главе решаются следующие задачи:

1. Разработка метода выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах
2. Разработка алгоритмов в составе метода выравнивания генетических последовательностей и получение аналитической оценки их вычислительной и пространственной сложности

Глава 2. Разработка метода выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах

Методы выравнивания генетических последовательностей на референсный геном и на пангеномный граф обладают каждый своими преимуществами и недостатками (раздел 1.2.7). Для того чтобы объединить преимущества данных методов, разработан метод выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах.

2.1 Описание и постановка задачи

Для эффективного выравнивания последовательностей в таких задачах, как полногеномное ресеквенирование, в современных биоинформатических инструментах выравнивания ридов применяется двухэтапный подход seed-chain-align, описанный в разделе 1.2.4, позволяющий ускорить процесс выравнивания большого количества ридов на референсную последовательность.

При выполнении выравнивания рида для каждого генетического варианта возможны две ситуации: генетический вариант в секвенируемом организме не попадает в якорь в риде или генетический вариант попадает в якорь в риде.

В случае если генетический вариант в секвенируемом организме не попадает в якорь в риде – позиции якорей определяются верно и при выполнении ряда дополнительных условий (отсутствие множественного выравнивания, отсутствие участка, куда рид с вариантом выравнивается без замен) рид будет выровнен в корректную позицию, при этом качество выравнивания MAPQ будет снижено за наличие замены.

В случае если генетический вариант в секвенируемом организме попадает в якорь в риде – позиция якоря может быть определена неверно, что может привести к тому, что рид не будет выровнен или будет выровнен в неверную позицию на

геноме из-за ухудшения оценки качества составления цепочки и возможному выбору другой цепочки вместо необходимой. Это в дальнейшем может привести к ухудшению идентификации генетических вариантов в данном регионе последовательности.

Качественное и точное выравнивание ридов при анализе данных полногеномного секвенирования позволяет правильно идентифицировать генетические варианты секвенируемого образца. Как показано в статье "A comparison of seed-and-extend techniques in modern DNA read alignment algorithms", выравнивание на пангеномный граф позволяет улучшить качество выравнивания в вариабельных участках генома по сравнению с выравниванием на линейный референс. Для того чтобы сохранить преимущества обоих методов (эффективность выравнивания на линейный референс и точность выравнивания на пангеномный граф), необходимо реализовать возможность добавления информации о генетических вариантах в индекс референсного генома.

Выбор якорей можно сделать разными способами. Один из вариантов выбора якорей – все k-меры (последовательность нуклеотидов длины k) референсного генома. Однако при таком способе выбора якорей их количество будет равно длине референсного генома и будет занимать значительное количество памяти и влиять на скорость выравнивания. Один из альтернативных способов выбора якорей — использование минимизаторов (раздел 1.2.5).

В качестве удобного для модификаций внутреннего представления индекса может быть использована хэш-таблица позиций минимизаторов. В таком случае существует возможность модификации индекса путем добавления новых минимизаторов. Модификация алгоритма построения индекса (Рисунок 2.1) для последующего более качественного выравнивания ридов заключается в том, что на этапе поиска якорей после индексации референсного генома дополнительно выполняется добавление минимизаторов и их позиций для известных генетических вариантов.

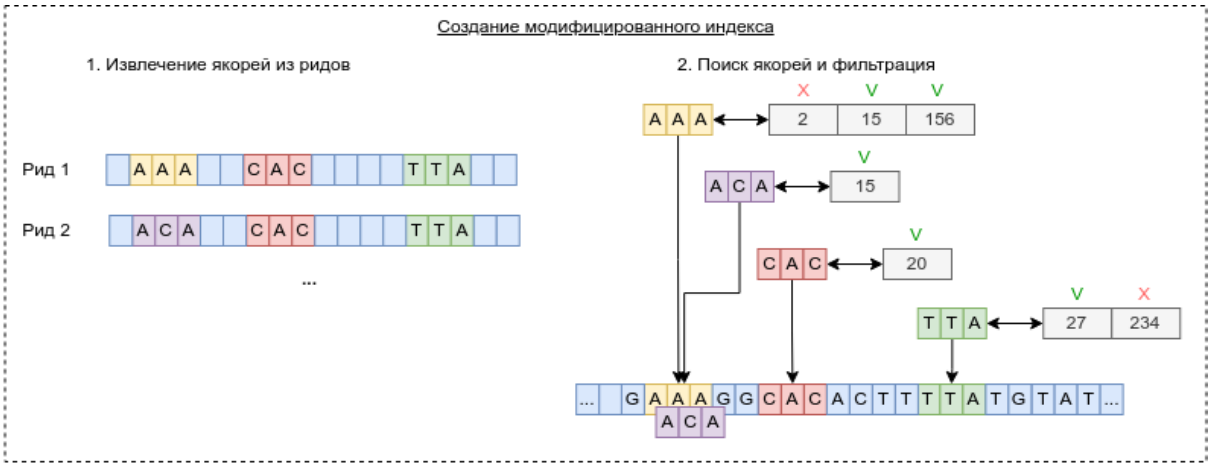

Рисунок 2.1 – Поиск якорей в модифицированном индексе

2.2 Разработка метода выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах

Введем следующие обозначения и определения:

Σ = {A; C; G; T} – алфавит нуклеотидов. Для символа (нуклеотида) $a \in \Sigma$ $\bar{a}$ – символ (нуклеотид), комплементарный по Уотсону-Крику.

Строка $s = a_1 a_2 .. a_n$ из символов множества Σ называется последовательностью ДНК. Ее длина $|s| = n$, она обратно комплементарна, т.е. для $s = a_1 a_2 .. a_n$ $\bar{s} = \bar{a}_n \bar{a}_{n-1} .. \bar{a}_1$.

k-мером называется последовательность ДНК длины k, таким образом, s_i^k = $= a_i \ldots a_{i+k-1}$ – k-мер, начинающийся в i-й позиции, Σ^k – множество всех k-меров.

Для удобства также определим функцию направления π: $\Sigma^* \times \{0,1\} \rightarrow \Sigma^*$ такое, что $\pi(s, 0) = s$, $\pi(\bar{s}, 1) = \bar{s}$. Здесь Σ^* – набор всех последовательностей ДНК.

Файл с генетическими вариантами (VCF_file) содержит N однонуклеотидных полиморфизмов (SNP) для X человек (для каждого человека известны фазированные генотипы для каждого генетического варианта из VCF-файла).

В качестве якорей для создания индекса и последующего выравнивания на него предложено использовать минимизаторы – короткие подстроки длины k, которые являются лексикографически минимальными в окне w.

Комбинацией SNP является набор из одного и более SNP.

Комбинацией SNP в окне длины k является комбинация SNP, позиции которых расположены в одной хромосоме в интервале [min_pos, $min\_pos + k$], где min_pos – минимальная позиция SNP в комбинации.

Допустимой комбинацией SNP в окне длины k является комбинация SNP в окне длины k, которая встречается хотя бы у одного из X человек в VCF_file (в случае если VCF_file содержит информацию о фазировании SNP, то с учетом данной информации).

Метод выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах состоит в последовательном решении двух задач, первая из которых разбивается на три шага, а вторая на четыре шага, где выполняются вычисления по разработанным алгоритмам, описанным ниже:

1. Создание модифицированного индекса референсной генетической последовательности с добавлением данных об известных генетических вариантах:
    a. Поиск минимизаторов для референсной генетической последовательности s
    b. Чтение генетических вариантов из VCF-файла и поиск допустимых комбинаций SNP в окне длины k
    c. Модификация участка исходной последовательности референсного генома заменой нуклеотидов в позициях с генетическими вариантами допустимой комбинации SNP и пересчет минимизаторов на модифицированных участках для каждой допустимой комбинации SNP.

2. Выравнивание генетических последовательностей на модифицированный индекс, для каждого рида:
    a. Вычисление минимизаторов для рида R
    b. Поиск позиций минимизаторов рида R в модифицированном индексе референсного генома
    c. Составление цепочек минимизаторов в референсном геноме в соответствии с их порядком в риде R
    d. Точное выравнивание последовательностей с помощью алгоритма Смита-Уотермана и определение позиции наилучшего выравнивания

2.3 Разработка алгоритма создания модифицированного индекса референсной генетической последовательности с добавлением данных об известных генетических вариантах

Для того чтобы создать модифицированный индекс для референсной генетической последовательности s и файла с генетическими вариантами VCF_file, выполняются следующие шаги:

1. Выполняется поиск минимизаторов для референсной генетической последовательности s (Листинг 2.1)
2. Выполняется чтение генетических вариантов из VCF-файла и поиск допустимых комбинаций SNP в окне длины *k* (Листинг 2.2)
3. Для каждой допустимой комбинации SNP выполняется модификация участка исходной последовательности референсного генома заменой нуклеотидов в позициях с генетическими вариантами допустимой комбинации SNP и пересчет минимизаторов на модифицированных участках (Листинг 2.3).

Листинг 2.1 – Алгоритм поиска минимизаторов для референсной генетической последовательности *s*

```
1 | Входные данные: параметры для поиска минимизаторов w и k;
```

```
2  последовательность s (|s| >= w+k-1)
3  Выходные данные:  (w,k)-минимизаторы, и их позиции
4
5  Function GetMinimizer (s, k, w) begin
6  M = ∅      // M - множество без дубликатов
7  for (i = 1 to |s| - w - k - 1) do
8    m = ∞
9    for (j = 0 to w - 1) do // найти минимальное значение
10     u = si+jk
11     v = s⃗i+jk
12     if (u != v) then // пропустить если направление не определено
13       m = min (m, min (u,v))
14   for (j = 0 to w - 1) do // собрать минимизаторы
15     u = si+jk
16     v = s⃗i+jk
17     if (u < v and u = m) then
18       M = M U {(m, i+j, 0)}
19     if (u > v and v = m) then
20       M = M U {(m, i+j, 1)}
21 return M
```

Листинг 2.2 – Алгоритм чтения генетических вариантов из VCF-файла и поиск допустимых комбинаций SNP в окне длины k

```
1  Входные данные: VCF_file с SNP, параметры w и k
2  Выходные данные: (w,k)-минимизаторы, и их позиции
3
4  List <SNP> SNP_list = NULL
5
6  Function GetMinimizersForSNPs(k, w, VCF_file)) begin
7    // считать SNP в однонаправленный список
8    for (str in readline(VCF_file)) do
9      SNP_list.insert_at_begin(SNP)
10
11   w_beg_ptr = SNP_list //window_start_pointer
12   cur_ptr = SNP_list // current_pointer
13   w_end_ptr = SNP_list // window_end_pointer
14   gap = k - 1
15   // перемещение по списку скользящим окном
16   while (w_end_ptr.next) do
17     while (w_end_ptr and w_end_ptr.pos > (cur_ptr.pos - gap))do
18       w_end_ptr = w_end_ptr.next
19     while (w_beg_ptr.pos > (cur_ptr.pos + gap)) do
```

```
    w_beg_ptr = w_beg_ptr.next
  return GetMinimizersW(w_beg_ptr, w_end_ptr, cur_ptr, k, w)
  cur_ptr = cur_ptr.next
  w_end_ptr = cur_ptr
 // последняя группа SNP
 while(w_beg_ptr.pos > cur_ptr.pos + gap)do
  w_beg_ptr = w_beg_ptr.next
return GetMinimizersW(w_beg_ptr, w_end_ptr.next, cur_ptr, k, w)
```

Листинг 2.3 – Алгоритм поиска минимизаторов для допустимых комбинаций SNP в окне длины k

```
Входные данные: параметры w и k, указатели на начало, конец окна и текущий
SNP
Выходные данные:  (w,k)-минимизаторы, и их позиции

Function GetMinimizersW (beg_ptr, end_ptr, cur_ptr) begin
 COMBS = ∅ // Combinations - множество допустимых комбинаций SNP
 Size = size(genotypes)

 for (i = 0 to Size - 1) do
  w_pointer = beg_ptr
  CombinationWithPos = {}
  while (w_pointer != end_ptr) do
   if (w_pointer.genotypes[i] == 0) then
    CombinationWithPos.append((w_pointer.REF, w_pointer.pos))
   if (w_pointer.genotypes[i] == 1) then
    CombinationWithPos.append((w_pointer.ALT, w_pointer.pos))
   w_start_pointer = w_start_pointer.next
  COMBS = COMBS U (CombinationWithPos)
  return AddVariants(Combination, cur_ptr.pos, k, w)
```

Введем определение $\bar{c}_n^k$ – число перестановок n элементов на k позициях с повторениями. $\bar{c}_n^k = (k+n-1)!/((n-1)!\times k!)$

В случае если n=4, можно упростить и аппроксимировать: $\bar{c}_4^k = (k+4-1)!/((4-1)!\times k!) = (k+3)!/(3!\times k!) = (k+1)(k+2)(k+3)/6 < (k+3)^3/6 = O(k^3/6)$

Лемма 1. Вычислительная сложность этапа добавления минимизаторов для строки длины $|s| = n$ составляет $O(n)$

Доказательство. При заданных коэффициентах k и w для вычисления минимизатора необходимо пройти по строке от 0 до $n - w - k - 1 < n$, при этом, на каждом шаге необходимо w раз прочитать строки длины k и определить минимальную, т.е. $O(k \times w \times n)$. Так как k и w – фиксированные константы, то вычислительная сложность алгоритма добавления минимизаторов равна $O(n)$.

Лемма 2. Вычислительная сложность этапа поиска минимизаторов для допустимых комбинаций SNP в окне длины k составляет $O(1)$

Доказательство. Согласно доказательству леммы 1 вычислительная сложность алгоритма добавления минимизаторов можно оценить сверху $O(k \times w \times n)$, где n – длина строки s. В случае поиска минимизаторов для одной комбинаций SNP в окне длины k (следовательно, $n = k$), сложность алгоритма можно оценить как $O(k^2 \times w)$. Число всех возможных комбинаций из четырех нуклеотидов в строке длины k равно $\bar{c}_4^k$. Общая сложность алгоритма поиска минимизаторов для допустимых комбинаций составляет $O(\bar{c}_4^k \times k^2 \times w)$. Так как $\bar{c}_4^k$, k и w фиксированные константы, то вычислительная сложность этапа поиска минимизаторов для допустимых комбинаций SNP в окне длины k равна $O(1)$.

Лемма 3. Вычислительная сложность этапа поиска минимизаторов для VCF-файла с N генетическими вариантами составляет $O(N)$

Доказательство. Для того, чтобы получить набор всех минимизаторов для всех комбинаций SNP исходного VCF-файла необходимо пройти окном длины k по VCF-файлу, так как вычислительная сложность этапа поиска минимизаторов для допустимых комбинаций SNP в окне длины k составляет $O(1)$, то общая сложность этапа поиска минимизаторов для VCF-файла с N генетическими вариантами составляет $O(N) \times O(1)$, то есть $O(N)$.

Вычислительную сложность алгоритма построения модифицированного индекса можно оценить с помощью следующей теоремы:

Теорема 1. Пусть Σ = {A; C; G; T} – алфавит нуклеотидов, строка $s = a_1 a_2 .. a_n$ из символов множества Σ – референсная последовательность длины n. Файл с генетическими вариантами (VCF-файл) содержит N SNP для X человек. Общая

вычислительная сложность алгоритма построения индекса референсной последовательности s модифицированного добавлением N генетических вариантов составляет $O(n)+O(N)$.

Доказательство. Добавление элемента в хэш-таблицу обладает вычислительной сложностью $O(1)$ в среднем. Согласно лемме 1 вычисление минимизаторов для референсной последовательности s обладает сложностью $O(n)$. Согласно леммам 1, 2 и 3 вычисление минимизаторов для генетических вариантов обладает сложностью $O(N)$. Следовательно, общая сложность алгоритма построения модифицированного индекса составляет $O(n)\times O(1)+O(N)\times O(1)$, то есть $O(n)+O(N)$, что и требовалось доказать.

Лемма 4. Общее количество потребляемой памяти для этапа добавления минимизаторов длины k с окном длины w для строки длины $|s| = n$ составляет $O(2\times k\times n)$ бит.

Доказательство. Для хранения минимизатора необходимо $2k$ битов памяти. В строке длины n содержится не более n минимизаторов. Тогда общее количество потребляемой памяти для этапа добавления минимизаторов длины k с окном длины w для строки длины $|s| = n$ составляет $O(2k\times n)$ бит.

Лемма 5. Общее количество потребляемой памяти для этапа поиска минимизаторов для допустимых комбинаций SNP в окне длины k составляет $O(2\times \bar{c}_4^{\,k}\times k^2)$ бит.

Доказательство. Согласно доказательству леммы 1 общее количество потребляемой памяти для этапа добавления минимизаторов длины k с окном длины w для строки длины $|s| = n$ составляет $O(2k\times n)$ бит, где n – длина строки s. В случае поиска минимизаторов для одной комбинаций SNP в окне длины k (следовательно, $n = k$), общее количество потребляемой памяти можно оценить как $O(2k^2)$ бит. Число всех возможных комбинаций из четырех нуклеотидов в строке длины k равно $\bar{c}_4^{\,k}$. Общее количество потребляемой памяти для этапа поиска минимизаторов для допустимых комбинаций SNP в окне длины k составляет $O(2\bar{c}_4^{\,k}\times k^2)$ бит.

Лемма 6. Общее количество потребляемой памяти для этапа поиска минимизаторов для VCF-файла с N генетическими вариантами составляет $O(2N\times\bar{c}_4^{\,k}\times k^2)$ бит

Доказательство. Для того чтобы получить набор всех минимизаторов для всех комбинаций SNP исходного VCF-файла, необходимо пройти окном длины k по VCF-файлу, так как количество потребляемой памяти для этапа поиска минимизаторов для допустимых комбинаций SNP в окне длины k составляет $O(2\bar{c}_4^{\,k}\times k^2)$ бит, то общая сложность этапа поиска минимизаторов для VCF-файла с N генетическими вариантами составляет $O(2N\times\bar{c}_4^{\,k}\times k^2)$ бит.

Количество потребляемой памяти предложенным алгоритмом построения модифицированного индекса можно оценить с помощью следующей теоремы:

Теорема 2. Пусть Σ = {A; C; G; T} – алфавит нуклеотидов, строка $s = a_1a_2..a_n$ из символов множества Σ – референсная последовательность длины n. Файл с генетическими вариантами (VCF-файл) содержит N SNP для X человек. Общее количество потребляемой памяти алгоритма построения индекса референсной последовательности s модифицированного добавлением N генетических вариантов для этапа поиска минимизаторов длины k оценивается как $O(2k^2\times n+k^5\times N/3)$ бит.

Доказательство. Добавление элемента в хэш-таблицу добавляет минимизатор и m бит (размер ключа). Согласно лемме 4 Общее количество потребляемой памяти для этапа добавления минимизаторов длины k с окном длины w для строки длины $|s| = n$ составляет $O(2k\times n)$ бит. Согласно леммам 4, 5 и 6 вычисление минимизаторов для генетических вариантов требует по памяти $O(2N\times\bar{c}_4^{\,k}\times k^2)$ бит. Пусть размер ключа в хэш-таблице составляет m бит. Для удобства последующих вычислений можно аппроксимировать $m=O(k)$ (k не может превышать длину рида, то есть 50-250 бит, при этом размер ключа хэш-таблицы также составляет несколько байт. Следовательно, общая сложность алгоритма построения модифицированного индекса составляет

$O(2k\times n+m\times n)+O(m\times N+2\bar{c}_4^{\,k}\times k^2\times N)$, то есть $O(2k^2\times n+(k+2\bar{c}_4^{\,k}\times k^2)\times N) = O(2k^2\times n+2\bar{c}_4^{\,k}\times k^2\times N) = O(2k^2\times n+k^5\times N/3)$ бит, что и требовалось доказать.

Пример подсчетов оценки потребляемой памяти согласно Теореме 2:

Пусть k=21; n=3×10^9 нуклеотидов в референсном геноме; N= 3×10^5 генетических вариантов.

Тогда алгоритм построения модифицированного индекса потребует не более $O(2k^2\times n+k^5\times N/3) = O(2\times21^2\times3\times10^9 + 21^5\times3\times10^5/3)= 2646\times10^9 + 408.41\times10^9 < 3047\times10^9$ бит, то есть примерно 38010^9 байт или примерно 380 гигабайт. В реальности количество минимизаторов для строки длины n значительно меньше n, что позволяет строить индекс на мощном персональном компьютере.

2.4 Разработка алгоритма выравнивания генетических последовательностей на модифицированный индекс

В качестве модельной реализации алгоритма выравнивания N_reads ридов на референсный геном s предложим следующий алгоритм, для каждого рида R необходимо:

1. Вычислить минимизаторы для рида R
2. Найти позиции минимизаторов рида R в индексе референсного генома
3. Составить цепочки минимизаторов в референсном геноме в соответствии с их порядком в риде R
4. Для не более чем N_chains цепочек произвести точное выравнивание последовательностей с помощью алгоритма Смита-Уотермана и вернуть позицию наилучшего выравнивания

Для того чтобы выровнять риды на модифицированный референсный геном модифицируем предложенный выше алгоритм следующим образом, для каждого рида R необходимо:

1. Вычислить минимизаторы для рида R

2. Найти позиций минимизаторов рида R в модифицированном индексе референсного генома
3. Составить цепочки минимизаторов в референсном геноме в соответствии с их порядком в риде R
4. Для не более чем N_chains цепочек произвести точное выравнивание последовательностей с помощью алгоритма Смита-Уотермана и вернуть позицию наилучшего выравнивания

Лемма 7. Вычислительная сложность этапа поиска позиций минимизаторов рида R длины $|R| = R\_len$ в индексе референсного генома s составляет $R\_len*O(1)$

Доказательство. Поиск значения в хэш таблице индекса референсного генома s можно оценить в среднем как $O(1)$. Количество минимизаторов в риде R длины $|R| = R\_len$ не превышает R_len. Соответственно, вычислительная сложность этапа поиска позиций минимизаторов рида R длины $|R| = R\_len$ в индексе референсного генома s составляет $R\_len*O(1)$

Лемма 8. Вычислительная сложность этапа позиции минимизаторов рида R длины $|R| = R\_len$ в модифицированном индексе референсного генома s составляет $O(R\_len)$

Доказательство. Поиск значения в хэш таблице модифицированного индекса референсного генома s можно оценить в среднем как $O(1)$. Количество минимизаторов в риде R длины $|R| = R\_len$ не превышает R_len. Соответственно, вычислительная сложность этапа поиска позиций минимизаторов рида R длины $|R| = R\_len$ в модифицированном индексе референсного генома s составляет $R\_len*O(1)$

Вычислительную сложность предложенного алгоритма выравнивания ридов на модифицированный референсный можно оценить с помощью следующей теоремы:

Теорема 3. Пусть $\Sigma = \{A; C; G; T\}$ – алфавит нуклеотидов, строка $s = a_1a_2..a_n$ из символов множества Σ – референсная последовательность длины n строка $R = b_1b_2..b_n$ из символов множества Σ – рид длины $|R|=R\_len$. Вычислительная

сложность алгоритма выравнивания ридов на модифицированный референсный геном и алгоритм выравнивания ридов на модифицированный референсный геном по сравнению с вычислительной сложностью алгоритма выравнивания ридов на референсный геном остается неизменной.

Доказательство. Алгоритм выравнивания ридов на референсный геном и алгоритм выравнивания ридов на модифицированный референсный геном различаются только в шаге 2. Согласно леммам 7 и 8 вычислительная сложность шага 2 эквивалентна и равна *R_len*O(1)*. Соответственно, вычислительная сложность алгоритма выравнивания ридов на модифицированный референсный геном и алгоритм выравнивания ридов на модифицированный референсный геном по сравнению с вычислительной сложностью алгоритма выравнивания ридов на референсный геном остается неизменной, что и требовалось доказать.

2.5 Выводы ко второй главе

Во второй главе описана разработка метода выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах, разработка алгоритмов в составе метода выравнивания генетических последовательностей и получение аналитической оценки их вычислительной и пространственной сложности.

Метод выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах является компромиссным по скорости и качеству между методом выравнивания на линейный референсный геном и методом выравнивания на пангеномный граф.

Далее во второй главе описаны алгоритм создания модифицированного индекса референсной генетической последовательности с добавлением данных об известных генетических вариантах и алгоритм выравнивания генетических последовательностей на модифицированный индекс.

Доказаны теоремы о вычислительной сложности алгоритмов построения модифицированного индекса и выравнивания на модифицированный индекс и оценка сложности алгоритма построения модифицированного индекса по памяти.

Оценки, полученные в результате доказательства теорем, показывают, что вычислительная сложность алгоритмов построения создания модифицированного индекса референсной генетической последовательности остается линейной, а вычислительная сложность алгоритмов выравнивания генетических последовательностей на модифицированный индекс не изменяется по сравнению с выравниванием на индекс референсного генома. Теорема об оценке пространственной сложности позволяет оценить количество оперативной памяти необходимой для работы реализации алгоритмов.

Глава 3. Разработка и реализация системы анализа данных NGS на базе программного конвейера, реализующего предложенный метод выравнивания генетических последовательностей

Третья глава посвящена разработке и реализации архитектуры программного конвейера, реализующего предложенный метод и алгоритмы.

Создание качественного биоинформатического программного конвейера – это долгосрочный и итеративный процесс, который требует внимания к множеству деталей, начиная от выбора инструментов и алгоритмов и заканчивая оптимизацией производительности.

Зачастую разработка программных конвейеров ведется не систематизированно, без использования систем управления программными конвейерами и версионирования, кроме того возникают проблемы при масштабировании. В таком случае достаточно сложно развивать существующую кодовую базу программного конвейера.

На рисунке 3.1 представлена архитектура системы анализа данных NGS на базе программного конвейера, которая состоит из следующих частей:

- Реестр контейнеров, содержит docker контейнеры инструментов с фиксированными версиями, что позволяет обеспечить воспроизводимость анализа
- Облачное хранилище данных NGS, содержит данные секвенирования
- Облачное хранилище справочных данных, содержит данные референсных геномов и генетических баз данных различных версий
- Инструмент жизненного цикла ПО, содержит версионированный код программного конвейера (позволяет обеспечить воспроизводимость анализа) и инструменты поддержания жизненного цикла разработки
- Программный конвейер

- Система управления программными конвейерами, это система, которая управляет запуском программного конвейера и логированием вывода биоинформатических инструментов
- Облачная среда осуществляет предоставление вычислительных ресурсов по запросу, что обеспечивает масштабируемость системы
- Вычислительный кластер развернутый в облачной среде обеспечивает возможность непрерывного анализа данных с возможностью быстрого масштабирования подключением новых вычислительных узлов
- Система управления ресурсами кластера позволяет ставить в очередь на выполнение программные конвейеры и отслеживать статус выполнения

Рисунок 3.1 – Архитектура системы анализа данных NGS на базе программного конвейера

Для построения масштабируемого программного конвейера анализа выходных данных секвенирования необходимо выполнить следующую последовательность действий:

1. Определить решаемую задачу и цели биоинформатического анализа, провести выбор наборов данных для тестирования программного конвейера, методики оценки качества программного конвейера и допустимых порогов значений метрик качества на тестовых данных.
2. Определить требования к вычислительным мощностям, к производительности программного конвейера и возможности его масштабирования. Настроить физические или виртуальные вычислительные узлы.
3. Исходя из ограничений по вычислительным мощностям и анализа научной литературы выбрать набор прикладных инструментов для создания программного конвейера, по-возможности опираясь на руководства ведущих биоинформатических лабораторий и организаций.
4. Выбрать способ реализации программного конвейера, в частности, определить необходимость использования фреймворков управления программными конвейерами и домен-специфичный язык (DSL[85]) описания программного конвейера.
5. Реализовать программный конвейер, исследовать его работу на тестовых данных, при необходимости провести подбор параметров инструментов программного конвейера, выбор версий инструментов и биоинформатических баз данных, добавление дополнительных инструментов в программный конвейер.
6. Исследовать производительность программного конвейера путем запуска анализа на тестовых и/или реальных данных, определив узкие места и возможности распараллеливания программного конвейера.
7. При необходимости реализовать контейнеризацию отдельных инструментов программного конвейера для упрощения его развертывания, зафиксировав версии инструментов, используемых в контейнерах системных библиотек и биоинформатических баз данных.
8. Сконфигурировать программный конвейер для непрерывной работы на вычислительном кластере. При необходимости, скорректировать

конфигурацию запуска на основе данных производительности программного конвейера и отдельных инструментов.

Целью исследования является создание масштабируемого программного конвейера, принимающего на вход выходные данные секвенатора (файл в формате FASTQ) и выдающего на выходе файл с набором генетических вариантов человека в формате VCF.

Библиотека minimap2_index_modifier для создания модифицированного индекса инструмента minimap2 реализована на основе разработанного метода создания модифицированного индекса референсной генетической последовательности с добавлением данных об известных генетических вариантах (глава 2).

Также для исследования масштабирования программного конвейера разработан инструментарий развертывания программного конвейера на SLURM кластере в облачной среде Asperitas.

Дальнейшие пункты третьей главы соответствуют нумерации шагов последовательности действий построения масштабируемого программного конвейера анализа NGS данных полногеномного секвенирования человека.

3.1 Определение решаемой задачи и цели биоинформатического анализа, выбор наборов данных для тестирования программного конвейера, выбор методики оценки качества программного конвейера и допустимых порогов значений метрик качества на тестовых данных

Для задачи анализа данных NGS полногеномного секвенирования (при котором из выходных данных секвенатора в формате FASTQ необходимо получить геном человека в формате VCF для его последующего анализа врачом-генетиком) могут быть использованы реальные данные проекта "The genome in a bottle". В качестве метрик могут выступать такие метрики как точность, полнота и F-мера. Для промежуточной оценки качества выравнивания можно использовать анализ распределения метрики качества выравнивания ридов MAPQ. Для того чтобы

оценить метрики качества для любых данных проекта "The genome in a bottle", необходимо заложить в реализацию программного конвейера возможность работы на двух версиях референсного генома: hg19 и hg38.

В качестве данных для тестирования возьмем данные проекта "The genome in a bottle", на которых проводилось первое соревнование "FDA Truth Challenge", а именно, данные генома HG002:

1. Эталонные FASTQ файлы:
    - HG002-NA24385-50x_1.fastq.gz
    - HG002-NA24385-50x_2.fastq.gz
2. Эталонный VCF-файл и вспомогательный файл:
    - HG002_GIAB_highconf_IllFB-IllGATKHC-CG-Ion-Solid_CHROM1-22_v3.2.2_highconf.vcf.gz
3. Файл геномных регионов, в пределах которого происходит анализ и сравнение генетических вариантов:
    - HG002_GIAB_highconf_IllFB-IllGATKHC-CG-Ion-Solid_CHROM1-22_v3.2.2_highconf.bed

В качестве данных для дополнительного анализа также будут выступать HG002 со сниженным покрытием и данные генома HG001:

1. Эталонные FASTQ файлы:
    - HG001-NA12878-35x_1.fastq.gz
    - HG001-NA12878-35x_2.fastq.gz
2. Эталонный VCF-файл и вспомогательный файл:
    - HG001_GRCh38_1_22_v4.2.1_benchmark.vcf.gz
3. Файл геномных регионов, в пределах которого происходит анализ и сравнение генетических вариантов:
    - HG001_GRCh38_1_22_v4.2.1_benchmark.bed

К FASTQ файлам применена стандартная операция тримминга адаптерных последовательностей с помощью инструмента cutadapt[86], ниже приведена строка запуска cutadapt (Листинг 3.1).

Листинг 3.1 – Команда запуска инструмента cutadapt

```
cutadapt --minimum-length 60 -j 250 -a
AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A
AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --pair-filter=any  -o
HG002-NA24385-50x_cutadapt0_1.fastq.gz -p
HG002-NA24385-50x_cutadapt0_2.fastq.gz HG002-NA24385-50x_1.fastq.gz
HG002-NA24385-50x_2.fastq.gz
```

Согласно таблице лучших метрик[87] среди всех программных конвейеров, участвовавших в соревнованиях "FDA Truth Challenge" (Таблица 1), метрика *Recall* имеет максимальное значение 0.999673 для программного конвейера "bgallagher-sentieon".

Таблица 1 – Лучшие значения метрик среди всех программных конвейеров "FDA Truth Challenge"

Тип варианта	**Recall**	**Precision**	**F-score**
SNP	0.999673 (bgallagher-sentieon)	0.999807 (astatham-gatk)	0.999587 (rpoplin-dv42)

Поставим целью максимизировать полноту (метрика Recall) так, чтобы превзойти показатель программного конвейера "bgallagher-sentieon".

3.2 Определение требований к вычислительным мощностям, к производительности программного конвейера и возможности его масштабирования. Настройка физических или виртуальных вычислительных узлов

В первой главе отмечено, что использование облачных технологий позволяет гибко масштабировать вычислительные ресурсы. Использование виртуальных кластеров в облаке совместно с планировщиком SLURM дает дополнительные преимущества по эффективному управлению пользовательскими задачами и предоставлению доступа к ресурсам кластера.

Для тестового стенда для разработки программного конвейера в облачной среде используем следующую конфигурацию виртуальных машин:

- мастер-узел (VCPU = 4, RAM = 8Gb, disk = 100Gb)
- четыре рабочих узла (VCPU = 64, RAM = 250Gb, disk = 1000Gb)
- для работы с данными – SSD размером 6Tb, примонтированный к рабочим узлам и мастер-узлу.

Настроим на выбранной конфигурации SLURM-кластер.

SLURM представляет собой интеграцию нескольких фоновых процессов в единую систему, каждый из которых выполняет свою функцию на кластере. На каждом рабочем хосте установлен slurmd, который выполняет отведенный ему этап работы и возвращает результат расчетов. Slurmdbd предоставляет пользовательский интерфейс для работы с базой данных (это может быть MariaDB или MySQL). Slurmrestd – фоновый процесс, обеспечивающий возможность взаимодействия пользователя с распределенным сервисом при помощи REST API интерфейса.

Решение по развертыванию SLURM в облачной инфраструктуре базируется на архитектуре распределенного сервиса и реализовано при помощи проекта ИСП РАН Michman[88] и инструмента управления конфигурациями Ansible. Michman – программный комплекс, позволяющий автоматически развертывать

распределенные сервисы, используя ресурсы IaaS[89] провайдера. Система оркестрации предоставляет REST API[90] интерфейс для взаимодействия с пользователем. Она позволяет создать кластер из виртуальных машин и настроить на нем указанные сервисы при условии, что они присутствуют в системе. Настройка фонового процесса slurmd на рабочие узлы кластера осуществляется путем установки deb/rpm-пакетов с помощью менеджера репозиториев Nexus из локальной сети оркестратора Michman. Также в Michman реализована ansible[91]-роль для автоматизации развертывания реестра докер-контейнеров в облачной инфраструктуре.

Ниже приведен пример конфигурации для REST API запроса (Листинг 3.2), который инициирует создание виртуального SLURM-кластера, состоящего из двух рабочих узлов с операционной системой CentOS-8-Stream на каждом.

Листинг 3.2 – Конфигурация REST API запроса, который инициирует создание виртуального SLURM-кластера

```
{
        "DisplayName": "slurm-test",
        "Services":
        [
                {
                        "Name": "Slurm service",
                        "Type": "slurm",
                        "Version": "Slurm",
                        "Config":
                        {
                                "use_gpu": "false",
                                "cons_res": "Core",
                        }
                }
        ],
        "Image": "centos8",
        "NHosts": 2
}
```

Для запуска SLURM-кластера на базе облачных виртуальных машин требуется:

1. Создать облачные виртуальные машины с необходимой операционной системой
2. Развернуть SLURM-кластер с помощью процедуры, описанной выше
3. Создать конфигурационные файлы (Листинг 3.3) для управления ресурсами и задачами на облачных виртуальных машинах
4. Настроить вычислительные узлы для работы с Cromwell (установить при необходимости docker, получить доступ к образам docker-контейнеров, java 11)
5. Запустить SLURM и начать отправлять задачи на облачные виртуальные машины для выполнения

Листинг 3.3 – Конфигурация SLURM-кластера

```
#Configuration file for slurm
#Authentification
AuthType=auth/munge

#The accounting storage mechanism type.
AccountingStorageType= accounting_storage/none

#The name by which this Slurm managed cluster is known in the accounting
database.
ClusterName= bio-test-slurm

#Hostname of the machine where Slurm control daemon is executed
ControlMachine= bio-test-slurm-master

SlurmctldDebug=debug5
SelectType=select/cons_res
SelectTypeParameters=CR_CPU_Memory
DefMemPerCPU=3000
MinJobAge=100000

#The job accounting mechanism type.
JobAcctGatherType= jobacct_gather/linux

#If set to "ALL" then jobs which exceed a partition's size and/or time limits will be
rejected at submission time.
EnforcePartLimits= ALL
```

```
27
28 #Identifies the plugin to be used for process tracking on a job step basis.
29 #proctrack/cgroup. Uses linux cgroups to constrain and track processes, and is the
30 default for systems with cgroup support.
31 ProctrackType= proctrack/cgroup
32
33 #Controls when a DOWN node will be returned to service.
34 # 1  means,  DOWN node will become available for use upon registration with a valid
35 configuration only if it was set DOWN due to being non-responsive.
36 #For any other reason, its state will not automatically be changed.
37 ReturnToService= 1
38
39 SchedulerType= sched/backfill
40
41 #Pathname of a file into which the slurmctld daemon's logs are written.
42 SlurmctldLogFile= /var/log/slurm-llnl/slurmctld.log
43
44 #Pathname of a file into which the slurmctld daemon may write its process id.
45 SlurmctldPidFile= /var/run/slurm-llnl/slurmctld.pid
46
47 #Pathname of a file into which the slurmd daemon's logs are written.
48 SlurmdLogFile= /var/log/slurm-llnl/slurmd.log
49
50 #Pathname of a file into which the slurmd daemon may write its process id.
51 SlurmdPidFile= /var/run/slurm-llnl/slurmd.pid
52
53 #Pathname of a directory into which the slurmd daemon's state information
54 #and batch job script information are written.
55 SlurmdSpoolDir= /var/spool/slurmd
56
57 #Pathname of a directory into which the Slurm controller, slurmctld, saves its state.
58 #Slurm state will be saved here to recover from system failures.
59 StateSaveLocation= /var/spool/slurmctld
60
61 #Optional parameters for the task plugin.
62 TaskPluginParam= None
63
64 TaskPlugin = task/cgroup
65
66 #Nodes
67 #CPUs - number of logical processors on the node.
68 #RealMemory - size of real memory on the node in megabytes.
```

```
NodeName = bio-test-slurm-slave-1 CPUs=64 RealMemory=240000
NodeName = bio-test-slurm-slave-2 CPUs=64 RealMemory=240000
NodeName = bio-test-slurm-slave-3 CPUs=64 RealMemory=240000
NodeName = bio-test-slurm-slave-4 CPUs=64 RealMemory=240000

#Partition
PartitionName=main Nodes  =  bio-test-slurm-slave-[1-4]  Default=YES
OverSubscribe=yes
```

3.3 Выбор набора прикладных инструментов для создания программного конвейера и реализация алгоритма модификации геномного индекса на основе минимизаторов для выравнивания ридов

3.3.1 Выбор набора прикладных инструментов для создания программного конвейера

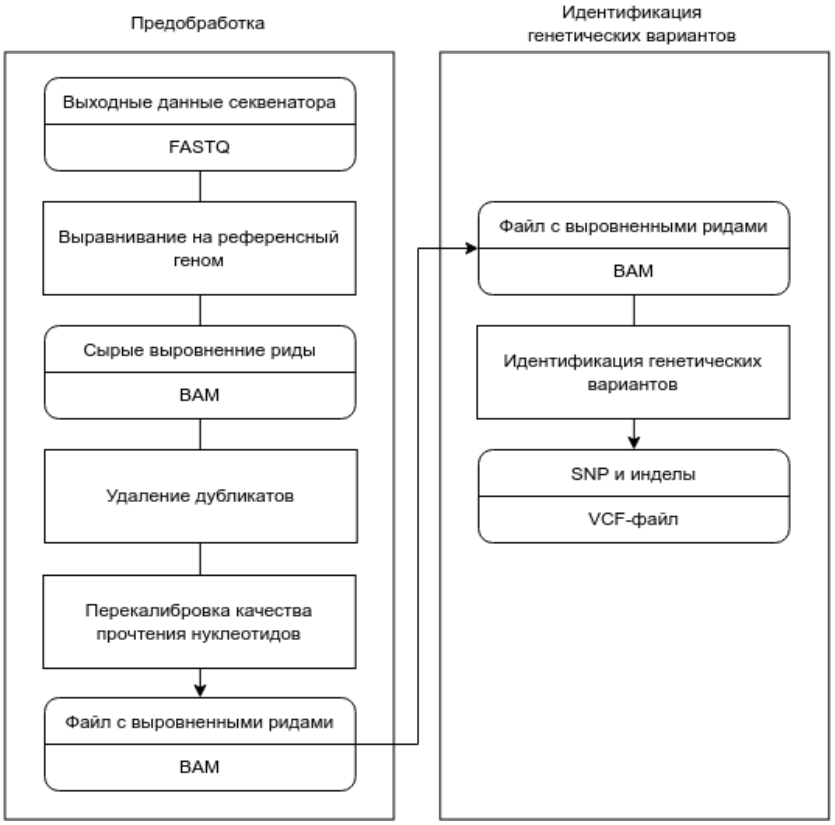

Рисунок 3.2 – Схема программного конвейера для анализа данных полногеномного секвенирования из набора конвейеров "Best Practices Workflows"

Программный конвейер построен согласно методике создания программных конвейеров секвенирования ДНК, разработанной в Институте Броуда (Broad Institute) (Рисунок 3.2).

В качестве инструментов для выполнения отдельных шагов программного конвейера согласно гайду по построению конвейеров анализа полногеномного секвенирования использовались следующие инструменты:

- Выравнивание на референсный геном – minimap2 (версия 2.24)
- Удаление дубликатов – samtools (версия 1.11)
- Перекалибровка качества прочтения нуклеотидов и идентификация генетических вариантов – фреймворк gatk (версия 4.1.7.0, инструменты BaseRecalibrator, HaplotypeCaller и др.)
- Работа с интервалами и определение качества выравнивания – фреймворк picard (версия 2.25.0, инструменты CollectWgsMetrics, ScatterIntervalList, MergeVCF и др.)
- тримминг (за рамками программного конвейера) – cutadapt (версия 4.4)
- сравнение с эталоном и получение итоговых метрик (за рамками программного конвейера) – hap.py (версия 0.3.15)

В качестве инструмента идентификации генетических вариантов выбран HaplotypeCaller, так как на вычислительных узлах отсутствуют графические ускорители.

Выбор инструмента для выравнивания (minimap2) обусловлен тем, что внутреннее устройство инструмента (в качестве якорей алгоритма выравнивания служат минимизаторы, внутреннее представление индекса представляет из себя хэш-таблицу) позволяет реализовать метод из главы 2.

Алгоритм программного конвейера приведен ниже (Листинг 3.4).

Листинг 3.4 – Алгоритм программного конвейера полногеномного секвенирования

```
1 minimap2 -Y -ax sr 1.fq.gz 2.fq.gz index.mni > aligned.bam
2 samtools sort aligned.bam > aligned.sorted.bam
```

```
samtools markdup aligned.sorted.bam > aligned.sorted.nodups.bam
samtools index aligned.sorted.nodups.bam > aligned.sorted.nodups.bam.bai
gatk BaseRecalibrator aligned.sorted.nodups.bam reference.fa --known-sites
dbsnp.vcf.gz --known-sites 1000G_phase3.indels.vcf.gz > BQSR_report.txt
gatk ApplyBQSR BQSR_report.txt aligned.sorted.nodups.bam reference.fa >
aligned.sorted.nodups.recalibrated.bam
gatk HaplotypeCaller aligned.sorted.nodups.recalibrated.bam reference.fa dbsnp.vcf
interval_list > result.vcf
```

Для различных инструментов программного конвейера используются следующие референсные файлы и базы данных:

- dbSNP (версия 138, последняя версия доступная для hg19)
- indels (1000 genomes phase 3)
- interval_list (из проекта GIAB)
- reference.fa (референсный геном hg19)

Программный конвейер реализован на языке WDL и сконфигурирован для запуска на SLURM-кластере в облачной среде ИСП РАН Asperitas. Все инструменты контейнеризированы и размещены в реестре контейнеров в gitlab.

3.3.2 Алгоритм работы инструмента minimap2

Ниже описан алгоритм выравнивания ридов на основе минимизаторов:

1. Алгоритм выравнивания ридов инструмента minimap2 первым шагом выполняет поиск позиций минимизаторов в референсном геноме и ридах
2. После нахождения позиций минимизаторов короткого прочтения алгоритм minimap2 выполняет этап построения цепочек якорей, чтобы попытаться связать соседние минимизаторы в референсном геноме в том же порядке, что и в исходном риде. Этот процесс позволяет быстро идентифицировать области потенциального выравнивания (цепочки минимизаторов) в эталонном геноме.
3. Последним этапом цепочки минимизаторов достраиваются до рида, который выравнивается на эталонный геном с помощью алгоримов точного

выравнивания, что позволяет учесть замены, пропуски и вставки в окончательном выравнивании.

Таким образом, minimap2 для того, чтобы обеспечить эффективное нахождение якорей, последующее объединение якорей в цепочки и выравнивание ридов, использует минимизаторы в качестве якорей алгоритма seed-chain-align и представление структуры индекса в виде хэш-таблицы, что позволяет применить метод модификации индекса добавлением информации о генетических вариантах, предложенный во второй главе.

3.3.3 Реализация метода выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах

Метод выравнивания генетических последовательностей на референсный геном реализован в виде библиотеки minimap2_index_modifier которая представляет из себя инструмент minimap2 версии 2.24 с доработанных функционалом построения модифицированного индекса по файлу с генетическими вариантами согласно методу, описанному в главе 2. Для работы с файлом генетических вариантов задействована библиотека htslib версии 1.17.

В качестве базы генетических вариантов взяты данные третьей фазы проекта "1000 геномов". В данной работе в индекс добавлялись только SNP с частотой встречаемости в популяции (MAF, minor allele frequency[92]) более 0,05 (стандартное пороговое значение для исключения редких генетических вариантов).

Результатом работы библиотеки minimap2_index_modifier является модифицированный индекс референсной последовательности для заданных параметров минимизаторов k и w.

3.4 Выбор способа реализации программного конвейера и определение необходимости использования фреймворков управления программными конвейерами и домен-специфичного языка описания программного конвейера

Для запуска программных конвейеров, написанных на языке WDL, используется программное средство Cromwell, настроенное для запуска на SLURM кластере.

Основные шаги для запуска задач на SLURM-кластере с помощью Cromwell:

1. Установка Cromwell на мастер-узле, с которого планируется управление вычислительными задачами.
2. Настройка конфигурации Cromwell, в которой указаны параметры для взаимодействия со SLURM-кластером. Параметры содержат информацию о распределенных ресурсах, очередях задач, параметрах запуска и т. д. Cromwell будет использовать настройки, указанные в конфигурационном файле, чтобы отправить задачи на SLURM-кластер для выполнения, а также будет следить за состоянием задач и собирать результаты после их завершения.
3. Создание задачи. Необходимо описать программный конвейер с использованием языка описания рабочих процессов, поддерживаемого Cromwell (например, WDL – Workflow Description Language).
4. Запуск задачи с помощью Cromwell.

Ниже приведен листинг части конфигурационного файла Cromwell (Листинг 3.5), который конфигурирует Cromwell для запуска на SLURM-кластере.

Листинг 3.5 – Конфигурация Cromwell в части запуска задач на SLURM-кластере

```
backend {
  # Override the default backend.
  default = "SLURM"
```

```
4
5   SLURM {
6     actor-factory =
7  "cromwell.backend.impl.sfs.config.ConfigBackendLifecycleActorFactory"
8     config {
9       script-epilogue = "sync"
10
11      runtime-attributes = """
12      Int runtime_minutes = 6000
13      String? docker
14      String? docker_user
15      Int cpu = 8
16      Int memory = 15000
17      String? ctat_dir
18      String? sdf_dir
19      """
20
21      sudo submit = """
22         sudo sbatch \
23           --wait \
24           -J ${job_name} \
25           -D ${cwd} \
26           -c ${cpu} \
27           -t ${runtime_minutes} \
28           --mem ${memory}
29           --wrap "/bin/bash ${script}"
30      """
31
32      submit-docker = """
33        # Pull the image using the head node, in case our workers don't have network
34  access
35
36        sudo sbatch \
37          -J ${job_name} \
38          -D ${cwd} \
39          -c ${cpu} \
40          --mem ${memory} \
41          --wrap "sudo docker run --cpus=${cpu} --memory=${memory}m --security-opt
42  seccomp:unconfined ${if defined(ctat_dir) then "-v " + ctat_dir +
43  ":/ctat_genome_lib_build_dir" else ""} ${if defined(sdf_dir) then "-v " + sdf_dir +
44  ":/${basename(sdf_dir)}" else ""} -v ${cwd}:${docker_cwd} ${docker} ${job_shell}
45  ${docker_script}"
```

```
46        """
47
48        kill = "scancel ${job_id}"
49        check-alive = "squeue -j ${job_id}"
50        job-id-regex = "Submitted batch job (\\d+).*"
51
52
53
54        # File system configuration.
55        filesystems {
56
57          # For SFS backends, the "local" configuration specifies how files are handled.
58          local {
59
60            # Try to hard link (ln), then soft-link (ln -s), and if both fail, then copy the files.
61            localization: [
62              "hard-link", "cached-copy", "copy"
63            ]
64            # An experimental localization strategy called "cached-copy" is also available
65  for SFS backends.
66            # This will copy a file to a cache and then hard-link from the cache. It will copy
67  the file to the cache again
68            # when the maximum number of hardlinks for a file is reached. The maximum
69  number of hardlinks can be set with:
70            # max-hardlinks: 950
71          }
72        }
73        default-runtime-attributes {
74          failOnStderr: false
75          continueOnReturnCode: 0
76  }}}}
```

3.5 Реализация программного конвейера и исследование его работы на тестовых данных

В данном разделе описаны детали реализации программного конвейера, в частности эксперименты по подбору параметров инструментов программного конвейера, выбор версий инструментов и биоинформатических баз данных, а также приводятся результаты экспериментов по исследованию работы пайплайна с использованием модифицированного индекса.

3.5.1 Подбор параметров инструмента minimap2

Для того чтобы подобрать параметры k и w для генерации индекса инструмента minimap2 и последующего выравнивания, проведен ряд экспериментов на данных проекта "Genome in a bottle" (Приложение А1. Таблица 9). В результате экспериментов показано, что наиболее высокий результат по полноте (метрика *Recall*) получен при k=27 и w=7.

В процессе проведения экспериментов возникло предположение, что качество выравнивания на краях интервалов может падать из-за того, что часть алгоритмов, например, алгоритм работы HaplotypeCaller, могут использовать информацию о ридах, которые не полностью попали в уверенный интервал. Экспериментально (Приложение А2. Таблица 10) показано, что увеличение границ интервалов на 1000 нуклеотидов увеличивает качество работы программного конвейера на эталонном геноме.

3.5.2 Метрики качества для оценки качества выравнивания на модифицированный индекс и всего программного конвейера

Для сравнения качества работы исходного и модифицированного программного конвейера реализован модуль подсчета метрик качества, который позволяет анализировать распределение метрики MAPQ в SAM файле и значения метрик качества идентификации генетических вариантов по сравнению с эталонным файлом генетических вариантов (VCF-файлом) образца HG002 из данных консорциума "The Genome in a Bottle". Для сравнения полученных VCF-файлов, содержащих генетические варианты, использовался инструмент hap.py. Модуль позволяет вычислять метрики $Recall = TP / (TP + FN)$, $Precision = TP / (TP + FP)$ и $F\text{-}score = 1 / (Recall^{-1} + Precision^{-1})$, где *TP* (True Positives) — идентифицированные генетические варианты из эталонного VCF-файла, *FP* (False Positives) — идентифицированные генетические варианты, отсутствующие в

эталонном VCF-файле, *FN* (False Negatives) – не идентифицированные генетические варианты из эталонного VCF-файла.

3.5.3 Результаты вычисления метрик работы программного конвейера до и после модификации индекса

Оценка качества программного конвейера для обработки данных секвенирования ДНК человека с использованием модифицированного и стандартного индексов проводилась на данных соревнования “The precisionFDA Truth Challenge”. Для сравнения качества идентификации однонуклеотидных полиморфизмов использовалась метрика Recall. Результаты оценки качества идентификации однонуклеотидных полиморфизмов приведены в таблице 2. По метрике Recall программный конвейер с модифицированным индексом превосходит программный конвейер bgallagher-sentieon, который занял первое место в соревновании “The precisionFDA Truth Challenge” (в соревновании использовались данные образца HG002 из проекта "The genome in a bottle").

Таблица 2 – Оценка качества идентификации однонуклеотидных полиморфизмов на покрытии 60X (SNP, образец HG002)

Программный конвейер	**TP**	**FN**	**Recall**
Со стандартным индексом	3054647	1217	0.999602
С модифицированным индексом	3054647	943	**0.999691**
bgallagher-sentieon	3054647	998	0.999673

Для дополнительного исследования работы программного конвейера для обработки данных секвенирования ДНК человека с использованием модифицированного индекса были проведены эксперименты по оценке качества разработанного программного конвейера с использованием модифицированного индекса референсного генома по сравнению с программным конвейером, который использует стандартный индекс инструмента minimap2 для разной глубины покрытия (покрытием называется среднее количество раз, которое каждый нуклеотид в геноме считывается в процессе секвенирования).

Для оценки этапа предобработки программного конвейера (Рисунок 5) оценим распределение значений метрики качества выравнивания (Mapping quality, MAPQ). В качестве метрик оценки конечного результата работы программного конвейера будем использовать такие метрики, как точность (Precision), полнота

(Recall) и F-мера. В качестве данных для тестирования возьмем данные секвенирования ДНК биологического образца HG002 из проекта "The genome in a bottle" (в рамках проекта собраны и охарактеризованы эталонные справочные образцы данных секвенирования ДНК человека).

В таблицах 3 и 4 приведены различия в количестве ридов с разными значениями метрики MAPQ после выполнения этапа предобработки программного конвейера при использовании стандартного и модифицированного индекса референсного генома. Значения MAPQ разбиты по группам: 60 (выравнивание ридов без замен и разрывов), 30-60 (выравнивание с высоким качеством), 2-60 (однозначное выравнивание любого качества). Результаты приведены для двух вариантов глубины покрытия: 17X и 60X.

Таблица 3 – Количество ридов с разными значениями MAPQ (HG002 60X)

MAPQ	**Стандартный**	**Модифицированный**	**Разница**
60	949840269	949271649	+568620
30-60	979056628	978898034	+158594
2-60	1007563422	1007539696	+23726

Таблица 4 – Количество ридов с разными значениями MAPQ (HG002 17X)

MAPQ	**Стандартный**	**Модифицированный**	**Разница**
60	250311191	250160776	+150415
30-60	257949672	257906764	+42908
2-60	265409566	265402391	+7175

Анализ распределения значений MAPQ показывает, что в результате модификации индекса увеличивается как количество ридов выровненных без замен и разрывов, так и общее количество качественно выровненных ридов.

Значения метрик для идентификации SNP, полученные с помощью инструмента hap.py при сравнении результатов работы программного конвейера с эталонным VCF-файлом биологического образца HG002 приведены в таблицах 5 и 6.

Разработанный программный конвейер показал снижение ложноотрицательных срабатываний на 25% (274 SNP) на данных с покрытием

60X, а также снижение ложноотрицательных срабатываний на 3% (2945 SNP) на данных с покрытием 17X.

Таблица 5 – Результаты сравнения полученного VCF-файла с эталоном для всего генома на покрытии 60X (SNP, образец HG002)

Тип индекса	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
Стандартный	3054647	1217	8417	0.999602	0.997251	0.998425
Модифицированный	3054647	943	8382	0.999691	0.997262	0.998475

Таблица 6 – Результаты сравнения полученного VCF-файла с эталоном для всего генома на покрытии 17X (SNP, образец HG002)

Тип индекса	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
Стандартный	3053446	35828	9786	0.988266	0.996767	0.992499
Модифицированный	3053446	33949	9793	0.988882	0.996767	0.992809

Таким образом показано что разработанный программный конвейер для обработки данных секвенирования ДНК человека с использованием модифицированного индекса превосходит существующие реализации по качеству (по полноте, Recall) идентификации однонуклеотидных полиморфизмов, кроме того разработанный программный конвейер с использованием модифицированного индекса превосходит аналогичный с использованием стандартного индекса по метрике Recall и количеству качественно выровненных ридов, при этом значения остальных метрик (Precision и F–score) не уменьшаются.

3.6 Исследование производительности программного конвейера путем запуска анализа на тестовых и/или реальных данных, определение узких мест и возможностей распараллеливания программного конвейера

Для определения потребления ресурсов и выявления узких мест программный конвейер профилирован с помощью утилиты SAR[93]. Результаты профилирования по загрузке процессора и оперативной памяти на тестовых

данных представлены на Рисунке 3.3 и Рисунке 3.4.

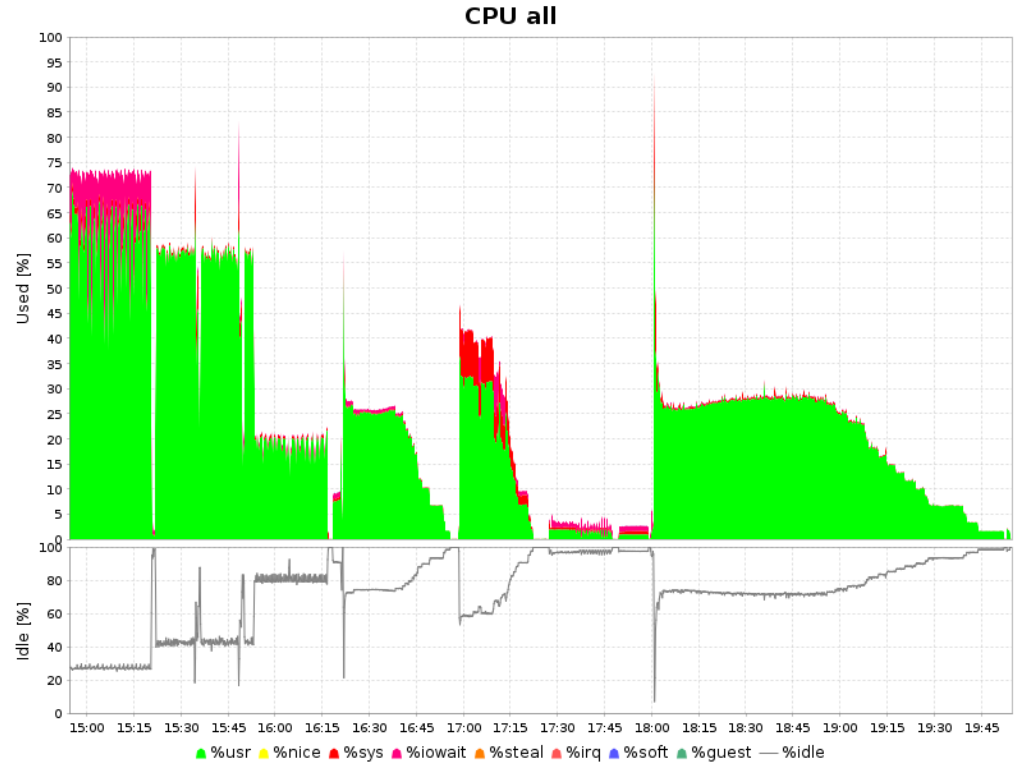

Рисунок 3.3 – Профилировка NGS программного конвейера. Процессорное время

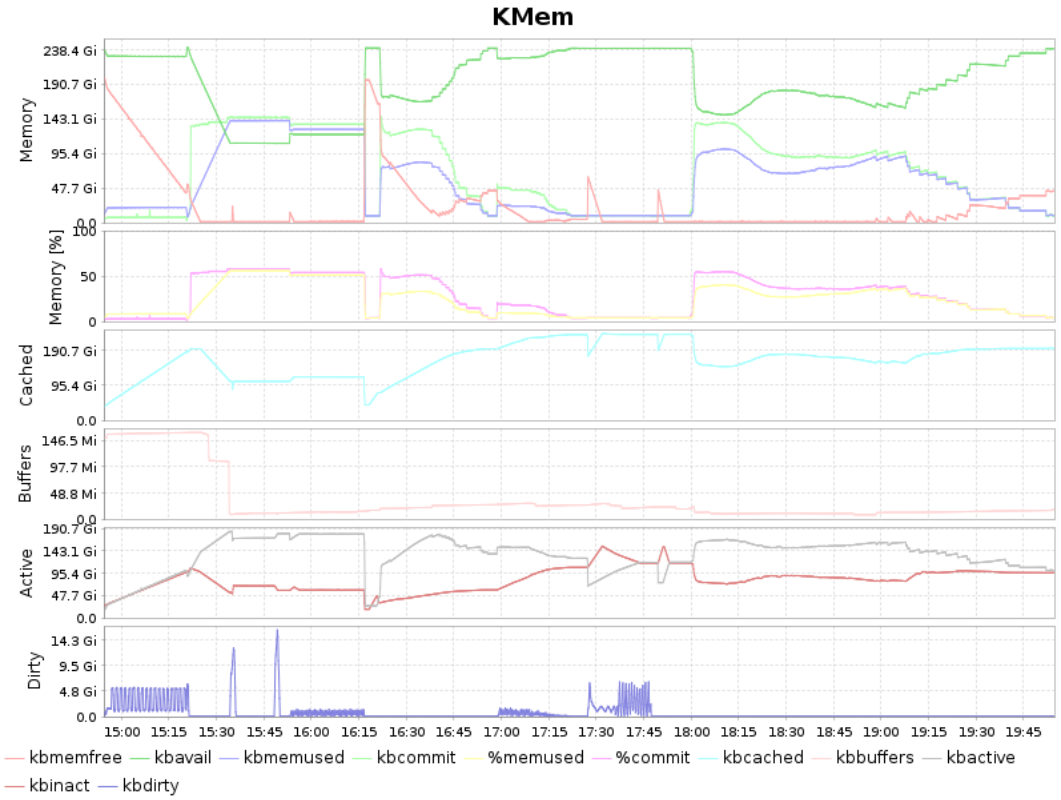

Рисунок 3.4 – Профилировка NGS программного конвейера. Потребление памяти

Исходя из анализа результатов профилировки и ряда дополнительных экспериментов по определению граничных значений, определены следующие требования по памяти к контейнеризированным инструментам программного конвейера (Таблица 7):

Таблица 7 – Ограничения инструментов программного конвейера по памяти

Задача WDL (инструмент)	Ограничение по памяти (Mb)
Align (minimap2)	220000
SortSam (picard)	4000
BaseRecalibrator	10000 на каждый поток
ApplyBQSR	4000 на каждый поток
HaplotypeCaller_GATK4_VCF (HaplotypeCaller)	24000 на каждый поток
MergeVCFs	2500

Данные ограничения прописаны в конфигурации запуска программного конвейера.

3.7 Реализация контейнеризации отдельных инструментов программного конвейера для упрощения его развертывания, с фиксацией версий инструментов, используемых в контейнерах системных библиотек и биоинформатических баз данных

Для обеспечения воспроизводимости программного конвейера все инструменты контейнеризированы. В качестве базового образа для всех контейнеров выбран контейнер phusion/baseimage.

Основными преимуществами docker-контейнера phusion/baseimage[94] являются:

1. Корректный процесс инициализации: baseimage поставляется с процессом инициализации /sbin/my_init, который корректно завершает дочерние

процессы и правильно реагирует на SIGTERM. Таким образом, контейнер не будет заполнен процессами-зомби, и команда docker stop будет работать правильно.

2. Исправлена несовместимость[95] пакетного менеджера apt[96] с docker.
3. syslog-ng – baseimage запускает syslog, чтобы важные системные сообщения не терялись.
4. cron – запускает утилиту cron, чтобы задания cron работали корректно.
5. SSH server – позволяет легко войти в контейнер для проверки или администрирования.

В качестве примера приведем листинг Dockerfile инструмента picard (Листинг 3.6).

Листинг 3.6 – Пример Dockerfile для инструмента picard

```
FROM phusion/baseimage:master
ARG PICARD_PUBLIC_VERSION=2.25.0
ENV TERM=xterm-256color \
PICARD_URL=https://github.com/broadinstitute/picard/releases/download/${PICARD
_PUBLIC_VERSION}/picard.jar

WORKDIR /usr/gitc
# Install dependencies
RUN set -eux; \
      apt-get update; \
      apt-get install -y \
        openjdk-8-jre \
        wget \
   && rm -rf /var/lib/apt/lists/*; \
# Install picard
wget ${PICARD_URL};
```

После того как контейнер инструмента собран, необходимо загрузить его в реестр контейнеров. Загруженные контейнеры доступны для загрузки при запуске программного конвейера при условии, что выполнен вход в реестр контейнеров (Рисунок 3.5), например, по токену с доступом на чтение.

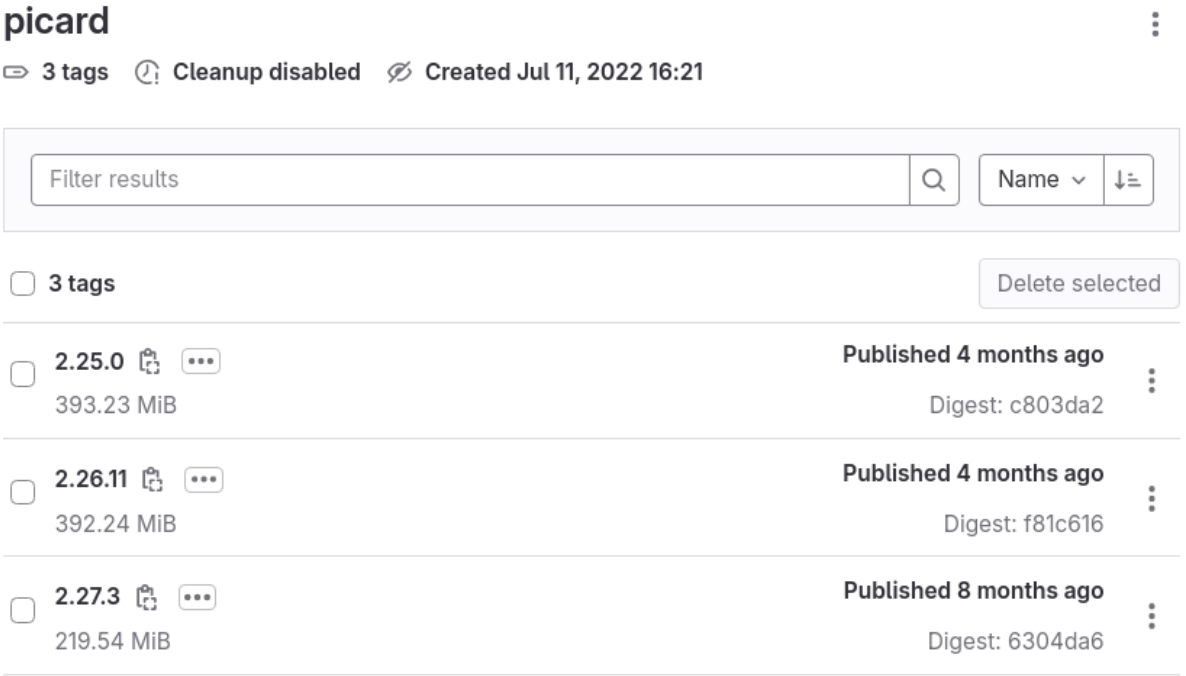

Рисунок 3.5 – Реестр контейнеров, версии контейнера с инструментом picard

Команда запуска контейнера прописана в конфигурации бэкенда фреймворка Cromwell (submit-docker). В конфигурации, описанной в разделе 3.4, контейнеры загружаются из реестра на рабочие узлы и запускаются с заранее заданными ограничениями по памяти и количеству вычислительных ядер. Параметры, рассчитанные в разделе 3.7, позволяют избежать ошибок, связанных с выделением памяти.

3.8 Конфигурация программного конвейера для непрерывной работы на вычислительном кластере. При необходимости, корректировка конфигурации запуска на основе данных производительности программного конвейера и отдельных инструментов

Для запуска сервера используется команда (Листинг 3.7).

Листинг 3.7 – Команда запуска сервера Cromwell

```
java -Dconfig.file=slurm.conf -jar /var/shared_dir/cromwell-80.jar server
```

Для постановки задачи в очередь используется команда (Листинг 3.8):

Листинг 3.8 – Команда постановки задачи для выполнения на сервере Cromwell

```
java -jar /var/ssd/cromwell-85.jar submit WholeGenomeGermlineSingleSample.wdl
--inputs hg19.hg002.k27w7.snps.maf0.05.json --imports warp.zip --options
options.json
```

Результаты запуска программного конвейера на разном количестве вычислительных узлов в кластере приведены в Таблице 8.

Таблица 8 – Результаты выполнения программного конвейера на SLURM-кластере, восемь одновременно запущенных конвейеров, покрытие 60X

Количество узлов	2	3	4
Время работы, минут	4300	3927	2505

Результаты профилировки показывают, что использование SLURM-кластера для анализа данных полногеномного секвенирования позволяет увеличить эффективность использования вычислительных ресурсов. Использование SLURM-кластера по запросу позволяет решить проблему масштабируемости и воспроизводимости анализа при возникновении потребности в анализе большого количества данных в сжатые сроки. Так добавление 2 узлов позволило прозрачно для пользователя увеличить производительность кластера в 1,7 раза.

3.9 Выводы к третьей главе

В третьей главе разработана и реализована архитектура системы анализа данных NGS на базе программного конвейера, реализующего предложенный метод выравнивания генетических последовательностей, описанный во второй главе. На основе инструмента minimap2 разработана библиотека minimap2_index_modifier для модификации индекса добавлением в него информации о генетических вариантах. Создан модифицированный индекс с помощью генетических вариантов третьей фазы проекта "1000 геномов".

Реализован масштабируемый программный конвейер анализа NGS данных полногеномного секвенирования человека согласно последовательности действий

описанной в начале главы 3. Определена задача, определены тестовые данные, выбрана методика качества оценки программного конвейера и допустимые пороги значений метрики Recall. За основу взят программный конвейер "Germline short variant discovery" от Broad Institute.

Для каждого шага программного конвейера выбраны инструменты и их версии, программный конвейер реализован на языке WDL для обеспечения возможности работы на SLURM кластере с помощью фреймворка Cromwell.

Проведен ряд экспериментов для подбора параметров инструмента minimap2, а также эксперимент, в результате которого показано, что увеличение границ интервалов на 1000 нуклеотидов увеличивает качество работы программного конвейера на эталонном геноме.

Для итогового программного конвейера с использованием модифицированного индекса проведены эксперименты по анализу данных проекта "The genome in a bottle". В результате достигнута поставленная цель превзойти лучший программный конвейер из участвовавших в соревновании "FDA Truth Challenge" по метрике *Recall* для задачи идентификации SNP. Также проведен анализ остальных метрик (*Precision*, *F-score*). Значения данных метрик несколько ниже, чем у лучших пайплайнов соревнования, однако в текущей реализации программного конвейера отсутствует шаг фильтрации, который должен уменьшить количество ложных срабатываний.

Для исследования масштабирования разработанного программного конвейера разработана методика и инструментарий развертывания программного конвейера на SLURM-кластере в облачной среде ИСП РАН Asperitas.

Показано, что программный конвейер увеличивает производительность при последовательном увеличении количества вычислительных узлов. Так, добавление двух узлов позволило прозрачно для пользователя увеличить производительность кластера в 1,7 раза и ускорить время анализа с 4300 минут до 2500 минут.

Заключение

Основные результаты работы заключаются в следующем:

1. Разработан метод выравнивания генетических последовательностей на референсный геном с использованием данных об известных генетических вариантах.
2. Разработаны алгоритмы в составе метода выравнивания генетических последовательностей и аналитические оценки их вычислительной и пространственной сложности через доказательство соответствующих теорем
3. Дана оценка потребления памяти для алгоритма модификации индекса на модельном примере показывает, что для создания модифицированного индекса достаточно мощностей сервера с 500гб оперативной памяти.
4. Разработана архитектура системы анализа данных NGS на базе программного конвейера, реализующего предложенный метод выравнивания генетических последовательностей
5. В рамках архитектуры реализован программный конвейер на основе популярного инструмента выравнивания ридов minimap2. Программный конвейер апробирован на данных проекта "The Genome in a Bottle".
6. Разработанный программный конвейер для обработки данных секвенирования ДНК человека с использованием модифицированного индекса превосходит существующие реализации по качеству (по полноте, Recall) идентификации однонуклеотидных полиморфизмов, кроме того разработанный программный конвейер с использованием модифицированного индекса превосходит аналогичный с использованием стандартного индекса по метрике Recall и количеству качественно выровненных ридов, при этом значения остальных метрик (Precision и F–score) не уменьшаются.

Благодарности

Хотелось бы выразить благодарность Турдакову Денису за научное руководство, Егору Гугучкину за активное содействие в получении результатов. А также моей жене Ольге за всестороннюю поддержку и коллегам из отдела "Информационные системы" ИСП РАН, принимавшим участие в обсуждениях в процессе работы.

Список сокращений и условных обозначений

Конвейер	программный конвейер, пайплайн, pipeline
База	нуклеотид, буква, обозначающая нуклеотид, base, nucleotide
Вариант	генетический вариант, мутация, variant, genetic variant
Вызов вариантов	определение или идентификация генетических вариантов, variant calling
Рид	короткое прочтение, read, short read
Якорь	короткая подпоследовательность, seed
1000 Genomes Project	Проект “1000 геномов”
alignment	выравнивание, картирование
BAM	Binary Alignment/Map, формат файла для выравнивания последовательностей (в бинарном виде)
BWA	Burrows-Wheeler Aligner, выравниватель Барроуза-Уиллера
DNA	дезоксирибонуклеиновая кислота, ДНК, deoxyribonucleic acid
Docker	платформа для контейнеризации приложений
FASTQ	формат файла для хранения последовательностей и их качества
F1	F-score, F-мера
GATK	genome analysis toolkit, геномный аналитический инструментарий
GIAB	The Genome in a Bottle, проект “Геном в бутылке”
GT	genotype, генотип
Indel	insertion/deletion, вставка или удаление нуклеотида

k-мер	нуклеотидная последовательность длины *k*
mapping	выравнивание, картирование
MAPQ	mapping quality, качество выравнивания
NGS	next generation sequencing, секвенирование нового поколения
RNA	ribonucleic acid, рибонуклеиновая кислота, РНК
SAM	Sequence Alignment/Map, формат файла для выравнивания последовательностей
SLURM	Simple Linux Utility for Resource Management, Простая Linux Утилита для Управления Ресурсами
SNP	single nucleotide polymorphism, однонуклеотидный полиморфизм, замена
SV	structural variations, структурные вариации
T2T	консорциум Telomere-to-Telomere
VCF	Variant Call Format, формат записи вариантов последовательностей
WGS	whole-genome sequencing, полногеномное секвенирование
WDL	Workflow Description Language, язык описания рабочих процессов или программных конвейеров

Список литературы

1. Lee, H., & Buckley, K. M. (1999). ECG data compression using cut and align beats approach and 2-D transforms. IEEE Transactions on Biomedical Engineering, 46(5), 556-564.
2. Church, K. (1993, June). Char_align: A program for aligning parallel texts at the character level. In 31st Annual Meeting of the Association for Computational Linguistics (pp. 1-8).
3. Melamed, I. D. (1999). Bitext maps and alignment via pattern recognition. Computational Linguistics, 25(1), 107-130.
4. Wikimedia Foundation. Sequence alignment. Wikipedia. https://en.wikipedia.org/wiki/Sequence_alignment
5. Levenshtein, V. (1965). Leveinshtein distance.
6. Likic, V. (2008). The Needleman-Wunsch algorithm for sequence alignment. Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne, 1-46.
7. Xia, Z., Cui, Y., Zhang, A., Tang, T., Peng, L., Huang, C., ... & Liao, X. (2021). A review of parallel implementations for the Smith–Waterman algorithm. Interdisciplinary Sciences: Computational Life Sciences, 1-14.
8. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. Journal of molecular biology, 215(3), 403-410.
9. Lee, C., Grasso, C., & Sharlow, M. F. (2002). Multiple sequence alignment using partial order graphs. Bioinformatics, 18(3), 452-464.
10. Makigaki, S., & Ishida, T. (2020). Sequence alignment using machine learning for accurate template-based protein structure prediction. Bioinformatics, 36(1), 104-111.
11. Shendure, J., & Ji, H. (2008). Next-generation DNA sequencing. Nature biotechnology, 26(10), 1135-1145.

12. Wadapurkar, R. M., & Vyas, R. (2018). Computational analysis of next generation sequencing data and its applications in clinical oncology. Informatics in Medicine Unlocked, 11, 75-82.

13. Lee, H. C., Lai, K., Lorenc, M. T., Imelfort, M., Duran, C., & Edwards, D. (2012). Bioinformatics tools and databases for analysis of next-generation sequence data. Briefings in functional genomics, 11(1), 12-24.

14. Nowotny, P., Kwon, J. M., & Goate, A. M. (2001). SNP analysis to dissect human traits. Current Opinion in Neurobiology, 11(5), 637-641.

15. Mullaney, J. M., Mills, R. E., Pittard, W. S., & Devine, S. E. (2010). Small insertions and deletions (INDELs) in human genomes. Human molecular genetics, 19(R2), R131-R136.

16. Qian, L., Luo, Z., Du, Y., & Guo, L. (2009). Cloud computing: An overview. In Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1 (pp. 626-631). Springer Berlin Heidelberg.

17. Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics, 34(18), 3094-3100.

18. US Food and Drug Administration. (2016). PrecisionFDA Truth Challenge.

19. Yoo, A. B., Jette, M. A., & Grondona, M. (2003, June). Slurm: Simple linux utility for resource management. In Workshop on job scheduling strategies for parallel processing (pp. 44-60). Berlin, Heidelberg: Springer Berlin Heidelberg.

20. Sanger, F., Nicklen, S., & Coulson, A. R. (1977). DNA sequencing with chain-terminating inhibitors. Proceedings of the national academy of sciences, 74(12), 5463-5467.

21. Meera Krishna, B., Khan, M. A., & Khan, S. T. (2019). Next-generation sequencing (NGS) platforms: an exciting era of genome sequence analysis. Microbial Genomics in Sustainable Agroecosystems: Volume 2, 89-109.

22. Rhoads, A., & Au, K. F. (2015). PacBio sequencing and its applications. Genomics, proteomics & bioinformatics, 13(5), 278-289.

23. Roberts, R. J., Carneiro, M. O., & Schatz, M. C. (2013). The advantages of SMRT sequencing. Genome biology, 14(6), 1-4.

24. Jain, M., Olsen, H. E., Paten, B., & Akeson, M. (2016). The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. Genome biology, 17, 1-11.

25. Ma, X., Shao, Y., Tian, L., Flasch, D. A., Mulder, H. L., Edmonson, M. N., ... & Zhang, J. (2019). Analysis of error profiles in deep next-generation sequencing data. Genome biology, 20, 1-15.

26. Paszkiewicz, K., & Studholme, D. J. (2010). De novo assembly of short sequence reads. Briefings in bioinformatics, 11(5), 457-472.

27. Koboldt, D. C. (2020). Best practices for variant calling in clinical sequencing. Genome Medicine, 12(1), 1-13.

28. Subramanian, I., Verma, S., Kumar, S., Jere, A., & Anamika, K. (2020). Multi-omics data integration, interpretation, and its application. Bioinformatics and biology insights, 14, 1177932219899051.

29. Cho, W. C. (2007). Proteomics technologies and challenges. Genomics, proteomics & bioinformatics, 5(2), 77-85.

30. Gibney, E. R., & Nolan, C. M. (2010). Epigenetics and gene expression. Heredity, 105(1), 4-13.

31. Genome reference consortium National Center for Biotechnology Information. Available at: https://www.ncbi.nlm.nih.gov/grc

32. FASTA Format for Nucleotide Sequences https://www.ncbi.nlm.nih.gov/genbank/fastaformat/

33. Pan, B., Kusko, R., Xiao, W., Zheng, Y., Liu, Z., Xiao, C., ... & Hong, H. (2019). Similarities and differences between variants called with human reference genome HG19 or HG38. BMC bioinformatics, 20(2), 17-29

34. Pan, B., Kusko, R., Xiao, W., Zheng, Y., Liu, Z., Xiao, C., ... & Hong, H. (2019). Similarities and differences between variants called with human reference genome HG19 or HG38. BMC bioinformatics, 20(2), 17-29.

35. Koboldt, D. C., Ding, L., Mardis, E. R., & Wilson, R. K. (2010). Challenges of sequencing human genomes. Briefings in bioinformatics, 11(5), 484-498.

36. Cock, P. J., Fields, C. J., Goto, N., Heuer, M. L., & Rice, P. M. (2010). The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic acids research, 38(6), 1767-1771.

37. Ewing, B., & Green, P. (1998). Base-calling of automated sequencer traces using phred. II. Error probabilities. Genome research, 8(3), 186-194.

38. Ewing, B., Hillier, L., Wendl, M. C., & Green, P. (1998). Base-calling of automated sequencer traces usingPhred. I. Accuracy assessment. Genome research, 8(3), 175-185.

39. Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., ... & 1000 Genomes Project Analysis Group. (2011). The variant call format and VCFtools. Bioinformatics, 27(15), 2156-2158.

40. Beier, S., Fiebig, A., Pommier, C., Liyanage, I., Lange, M., Kersey, P. J., ... & Scholz, U. (2022). Recommendations for the formatting of Variant Call Format (VCF) files to make plant genotyping data FAIR. F1000Research, 11.

41. Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv preprint arXiv:1303.3997.

42. Langmead, B., & Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. Nature methods, 9(4), 357-359.

43. Ahmed, N., Bertels, K., & Al-Ars, Z. (2016, December). A comparison of seed-and-extend techniques in modern DNA read alignment algorithms. In 2016 IEEE international conference on bioinformatics and biomedicine (BIBM) (pp. 1421-1428). IEEE.

44. Trapnell, C., & Salzberg, S. L. (2009). How to map billions of short reads onto genomes. Nature biotechnology, 27(5), 455-457

45. Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with Burrows–Wheeler transform. bioinformatics, 25(14), 1754-1760.

46. Grabowski, S., & Raniszewski, M. (2017). Sampled suffix array with minimizers. Software: Practice and Experience, 47(11), 1755-1771.

47. Eizenga, J. M., Novak, A. M., Sibbesen, J. A., Heumos, S., Ghaffaari, A., Hickey, G., ... & Garrison, E. (2020). Pangenome graphs. Annual review of genomics and human genetics, 21, 139-162.

48. Grytten, I., Rand, K. D., Nederbragt, A. J., & Sandve, G. K. (2020). Assessing graph-based read mappers against a baseline approach highlights strengths and weaknesses of current methods. BMC genomics, 21, 1-9.

49. Li, H., Ruan, J., & Durbin, R. (2008). Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome research, 18(11), 1851-1858.

50. Sequence Alignment/Map Format Specification - Github Pages, samtools.github.io/hts-specs/SAMv1.pdf. Accessed 3 Sept. 2023.

51. Cornish, A., & Guda, C. (2015). A comparison of variant calling pipelines using genome in a bottle as a reference. BioMed research international, 2015.

52. Guo, Y., He, J., Zhao, S., Wu, H., Zhong, X., Sheng, Q., ... & Long, J. (2014). Illumina human exome genotyping array clustering and quality control. Nature protocols, 9(11), 2643-2662.

53. PrecisionFDA Truth Challenge V2: Calling variants from short and long reads in difficult-to-map regions

54. Goutte, C., & Gaussier, E. (2005, March). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In European conference on information retrieval (pp. 345-359). Berlin, Heidelberg: Springer Berlin Heidelberg.

55. Overview – precisionFDA, https://precision.fda.gov/

56. Niu, J., Denisko, D., & Hoffman, M. M. (2022). The browser extensible data (BED) format. File Format Standard, 1, 8.

57. Lo, C. C., & Chain, P. S. (2014). Rapid evaluation and quality control of next generation sequencing data with FaQCs. BMC bioinformatics, 15(1), 1-8.

58. Cline, E., Wisittipanit, N., Boongoen, T., Chukeatirote, E., Struss, D., & Eungwanichayapant, A. (2020). Recalibration of mapping quality scores in Illumina short-read alignments improves SNP detection results in low-coverage sequencing data. PeerJ, 8, e10501.

59. Hwang, S., Kim, E., Lee, I., & Marcotte, E. M. (2015). Systematic comparison of variant calling pipelines using gold standard personal exome variants. Scientific reports, 5(1), 17875.

60. Sims, D., Sudbery, I., Ilott, N. E., Heger, A., & Ponting, C. P. (2014). Sequencing depth and coverage: key considerations in genomic analyses. Nature Reviews Genetics, 15(2), 121-132.

61. Sherry, S. T., Ward, M. H., Kholodov, M., Baker, J., Phan, L., Smigielski, E. M., & Sirotkin, K. (2001). dbSNP: the NCBI database of genetic variation. Nucleic acids research, 29(1), 308-311.

62. Landrum, M. J., Lee, J. M., Benson, M., Brown, G., Chao, C., Chitipiralla, S., ... & Maglott, D. R. (2016). ClinVar: public archive of interpretations of clinically relevant variants. Nucleic acids research, 44(D1), D862-D868.

63. Ng, P. C., & Henikoff, S. (2003). SIFT: Predicting amino acid changes that affect protein function. Nucleic acids research, 31(13), 3812-3814.

64. Adzhubei, I., Jordan, D. M., & Sunyaev, S. R. (2013). Predicting functional effect of human missense mutations using PolyPhen‐2. Current protocols in human genetics, 76(1), 7-20.

65. Koenig, D. (2007). Groovy in action.

66. Van Rossum, G. (2007, June). Python Programming Language. In USENIX annual technical conference (Vol. 41, No. 1, pp. 1-36).

67. Birger, C., Hanna, M., Salinas, E., Neff, J., Saksena, G., Livitz, D., ... & Getz, G. (2017). FireCloud, a scalable cloud-based platform for collaborative genome analysis: Strategies for reducing and controlling costs. bioRxiv, 209494.

68. Rehm, H. L., Page, A. J., Smith, L., Adams, J. B., Alterovitz, G., Babb, L. J., ... & Rodarmer, K. W. (2021). GA4GH: International policies and standards for data sharing across genomic research and healthcare. Cell genomics, 1(2).

69. Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. Nature biotechnology, 35(4), 316-319.

70. Van der Auwera, G. A., & O'Connor, B. D. (2020). Genomics in the cloud: using Docker, GATK, and WDL in Terra. O'Reilly Media.

71. Zhang, J., Baran, J., Cros, A., Guberman, J. M., Haider, S., Hsu, J., ... & Kasprzyk, A. (2011). International Cancer Genome Consortium Data Portal—a one-stop shop for cancer genomics data. Database, 2011, bar026.

72. Pan-cancer analysis of whole genomes //Nature. – 2020. – Т. 578. – №. 7793. – С. 82-93.

73. Lau, J. W., Lehnert, E., Sethi, A., Malhotra, R., Kaushik, G., Onder, Z., ... & Davis-Dusenbery, B. (2017). The Cancer Genomics Cloud: collaborative, reproducible, and democratized—a new paradigm in large-scale computational research. Cancer research, 77(21), e3-e6.

74. Feingold, E. A., Good, P. J., Guyer, M. S., Kamholz, S., Liefer, L., Wetterstrand, K., ... & Young, A. C. (2004). The ENCODE (ENCyclopedia of DNA elements) project. Science, 306(5696), 636-640.

75. Netto, M. A., Calheiros, R. N., Rodrigues, E. R., Cunha, R. L., & Buyya, R. (2018). HPC cloud for scientific and business applications: taxonomy, vision, and research challenges. ACM Computing Surveys (CSUR), 51(1), 1-29.

76. Sagiroglu, S., & Sinanc, D. (2013, May). Big data: A review. In 2013 international conference on collaboration technologies and systems (CTS) (pp. 42-47). IEEE.

77. Hassan, M., Awan, F. M., Naz, A., deAndrés-Galiana, E. J., Alvarez, O., Cernea, A., ... & Kloczkowski, A. (2022). Innovations in genomics and big data analytics for personalized medicine and health care: A review. International journal of molecular Sciences, 23(9), 4645.

78. Ware, J. S., Roberts, A. M., & Cook, S. A. (2012). Next generation sequencing for clinical diagnostics and personalised medicine: implications for the next generation cardiologist. Heart, 98(4), 276-281.

79. Shang, J., Zhu, F., Vongsangnak, W., Tang, Y., Zhang, W., & Shen, B. (2014). Evaluation and comparison of multiple aligners for next-generation sequencing data analysis. BioMed research international, 2014.

80. Endrullat, C., Glökler, J., Franke, P., & Frohme, M. (2016). Standardization and quality management in next-generation sequencing. Applied & translational genomics, 10, 2-9.

81. Day‐Williams, A. G., & Zeggini, E. (2011). The effect of next‐generation sequencing technology on complex trait research. European journal of clinical investigation, 41(5), 561-567.

82. Jain, M. (2012). Next-generation sequencing technologies for gene expression profiling in plants. Briefings in functional genomics, 11(1), 63-70.

83. Nekrutenko, A., & Taylor, J. (2012). Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. Nature Reviews Genetics, 13(9), 667-672.

84. Siu, L. L., Conley, B. A., Boerner, S., & LoRusso, P. M. (2015). Next-generation sequencing to guide clinical trials. Clinical Cancer Research, 21(20), 4536-4544.

85. Mernik, M., Heering, J., & Sloane, A. M. (2005). When and how to develop domain-specific languages. ACM computing surveys (CSUR), 37(4), 316-344.

86. Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet. journal, 17(1), 10-12.

87. Precisionfda Truth Challenge. PrecisionFDA Truth Challenge – precisionFDA. https://precision.fda.gov/challenges/truth/results

88. Aksenova, E., Lazarev, N., Badalyan, D., Borisenko, O., & Pastukhov, R. (2020, December). Michman: an Orchestrator to deploy distributed services in cloud environments. In 2020 Ivannikov Ispras Open Conference (ISPRAS) (pp. 57-63). IEEE.

89. Manvi, S. S., & Shyam, G. K. (2014). Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. Journal of network and computer applications, 41, 424-440.

90. Masse, M. (2011). REST API design rulebook: designing consistent RESTful web service interfaces. " O'Reilly Media, Inc."

91. Hochstein, L., & Moser, R. (2017). Ansible: Up and Running: Automating configuration management and deployment the easy way. " O'Reilly Media, Inc.".

92. Linck, E., & Battey, C. J. (2019). Minor allele frequency thresholds strongly affect population structure inference with genomic data sets. Molecular Ecology Resources, 19(3), 639-647.

93. Thomas-Krenn.AG. (2014, June 27). Linux Performance Analysis using Ksar. Visit the main page. www.thomas-krenn.com/en/wiki/Linux_Performance_Analysis_using_kSar

94. Your Docker Imagemight be broken without you knowing it. Baseimage-docker: A minimal Ubuntu base image modified for Docker-friendliness. https://phusion.github.io/baseimage-docker/

95. Container cannot connect to upstart · ISSUE #1024 · Moby/Moby. https://github.com/dotcloud/docker/issues/1024

96. Silva, G. N. (2001). APT howto. http://www.debian.org/doc/manuals/apt-howto/index.en.html.

Приложение А

Эксперименты выполненные в процессе подбора параметров программного конвейера анализа данных полногеномного секвенирования человека

А.1 Подбор параметров k и w для инструмента minimap2

Таблица 9 – Результаты экспериментов по подбору параметров k и w для инструмента minimap2

k 12, w 8 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3181648	72738	14509	0.977649	0.99546	0.986474
INDEL	467702	460551	7151	2153	0.98471	0.995349	0.99
k 14, w 10 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3181648	54391	16368	0.983287	0.994911	0.989065
INDEL	467702	462627	5075	2063	0.989149	0.995563	0.992345
k 15, w 10 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3204631	49755	15996	0.984711	0.995033	0.989845
INDEL	467702	462986	4716	1976	0.989917	0.995752	0.992826
k 16, w 10 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**

SNP	3254386	3206915	47471	15864	0.985413	0.995077	0.990222
INDEL	467702	463167	4535	1967	0.990304	0.995773	0.993031
k 18, w 12 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3209060	45326	14975	0.986072	0.995355	0.990692
INDEL	467702	463285	4417	1812	0.990556	0.996106	0.993323
k 21, w 10 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3213759	40627	15676	0.987516	0.995145	0.991316
INDEL	467702	463473	4229	1848	0.990958	0.99603	0.993488
k 21, w 11 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3213034	41352	15288	0.987293	0.995264	0.991263
INDEL	467702	463435	4267	1813	0.990877	0.996105	0.993484
k 21, w 12 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3212363	42023	14593	0.987087	0.995477	0.991265
INDEL	467702	463447	4255	1773	0.990902	0.996191	0.993539
k 23, w 11 (HG001)							

Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3214140	40246	15042	0.987633	0.995341	0.991472
INDEL	467702	463497	4205	1787	0.991009	0.996161	0.993578
k 25, w 11 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3214579	39807	15327	0.987768	0.995254	0.991497
INDEL	467702	463506	4196	1782	0.991028	0.996172	0.993593
k 25, w 12 (HG001)							
Тип варианта варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3213816	40570	14603	0.987534	0.995476	0.991489
INDEL	467702	463495	4207	1741	0.991005	0.996259	0.993625
k 26, w 10 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3215358	39028	15617	0.988008	0.995166	0.991574
INDEL	467702	463546	4156	1778	0.991114	0.996181	0.993641
k 26, w 11 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3214736	39650	15187	0.987816	0.995298	0.991543
INDEL	467702	463523	4179	1739	0.991065	0.996264	**0.993658**

k 26, w 12 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3213992	40394	14746	0.987588	0.995433	0.991495
INDEL	467702	463486	4216	1728	0.990986	0.996287	0.993629
k 27, w 8 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3216531	37855	16225	0.988368	0.994981	0.991663
INDEL	467702	463577	4125	1819	0.99118	0.996093	0.993631
k 27, w 9 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3216099	38287	15678	0.988235	0.995148	0.99168
INDEL	467702	463564	4138	1798	0.991152	0.996138	0.993639
k 27, w 10 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3215522	38864	15204	0.988058	0.995294	0.991663
INDEL	467702	463540	4162	1756	0.991101	0.996228	0.993658
k 27, w 11 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**

SNP	3254386	3215064	39322	15127	0.987917	0.995317	0.991603
INDEL	467702	463449	4253	1748	0.990907	0.996244	0.993568
k 27, w 12 (HG001)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3214206	40180	14575	0.987654	0.995486	0.991554
INDEL	467702	463470	4232	1730	0.990952	0.996283	0.99361
k 27, w 7 (extended 1k, f2000, modified index maf 0.05) (HG002)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F1**
SNP	3054647	3053636	1011	8611	0.999669	0.997188	0.998427
INDEL	344566	343305	1261	1574	0.99634	0.995435	0.995887
k 27, w 7 (extended 1k, f2250,6250, modified index maf 0.05) (HG002)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F1**
SNP	3054647	3053680	967	8390	0.999683	0.99726	0.99847
INDEL	344566	343299	1267	1558	0.996323	0.995481	0.995902
k 27, w 7 (extended 1k, f2500,6500, modified index maf 0.05) (HG002)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F1**
SNP	3054647	3053695	952	8370	0.999688	0.997266	0.998476
INDEL	344566	343295	1271	1548	0.996311	0.995509	0.99591
k 27, w 7 (extended 1k, f2500,6500, modified index NO INDELS IN COMBO maf 0.05, cutadapt0) (HG002)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F1**
SNP	3054647	3053707	**943**	8382	**0.999691**	0.997262	0.995907

INDEL	344566	343297	1269	1542	0.996317	0.995527	0.995922
k 27, w 7 (extended 1k, f2750,6750, modified index maf 0.05, cutadapt0) (HG002)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F1**
SNP	3054647	3053679	968	8342	0.999683	0.997276	0.998478
INDEL	344566	343302	1264	1544	0.996332	0.995521	0.995926
k 27, w 7 (extended 1k, f3000, modified index maf 0.05, cutadapt0) (HG002)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F1**
SNP	3054647	3053671	976	8336	0.99968	0.997277	0.998478
INDEL	344566	343301	1265	1542	0.996329	0.995527	0.995928
k 27, w 7 (extended 1k, f6000,10000 modified index maf 0.05, cutadapt0) (HG002)							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F1**
SNP	3054647	3053691	956	8324	0.999687	0.997281	0.998483
INDEL	344566	343290	1276	1542	0.996297	0.995527	0.995912

А.2 Исследование влияния ширины интервала для идентификации вариантов

Таблица 10 – Результаты работы инструмента hap.py сравнения полученных VCF с эталонным для всего генома на покрытии 30X HG001

Стандартный minimap2:							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3206982	47404	15186	0.985434	0.995287	0.990336
INDEL	467702	461358	6344	2905	0.986436	0.993746	0.990077
Minimap2 с измененным индексом, включающим все SNP и индели из 1000 genomes:							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3207820	46566	27047	0.985691	0.991638	0.988656
INDEL	467702	460952	6750	3948	0.985568	0.991512	0.988531
Minimap2 с измененным индексом, включающим все SNP для HG001, фаста с расширенным interval_list (+1000):							
Тип варианта	**Всего вариантов**	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F-score**
SNP	3254386	3220746	**33640**	**13042**	**0.989663**	0.995967	0.992805
INDEL	467702	461804	5898	2587	0.987389	0.994432	0.990898