УДК 004.021

МЕТОД СБОРКИ КОНТИГОВ ГЕНОМНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ НА ОСНОВЕ СОВМЕСТНОГО ПРИМЕНЕНИЯ ГРАФОВ ДЕ БРЮИНА И ГРАФОВ ПЕРЕКРЫТИЙ

А.В. Александров, С.В. Казаков, С.В. Мельников, А.А. Сергушичев, Ф.Н. Царев

Предлагается метод сборки контигов геномных последовательностей. Особенностью этого метода является разбиение процесса сборки контигов на два этапа – сборка квазиконтигов из чтений и сборка контигов из квазиконтигов. На первом из этапов используется граф де Брюина, на втором – граф перекрытий. Описываются результаты экспериментального исследования разработанного метода на чтениях генома рыбы Maylandia zebra, размер генома которой составляет примерно миллиард нуклеотидов. Преимущество разработанного метода состоит в том, что для его работы требуется существенного меньше оперативной памяти по сравнению с существующими программными средствами для сборки генома.

Ключевые слова: сборка генома, контиги, граф де Брюина, граф перекрытий.

Введение

Многие современные задачи биологии и медицины требуют знания геномов живых организмов, которые состоят из нескольких нуклеотидных последовательностей молекул дезоксирибонуклеиновой кислоты (ДНК). В связи с этим возникает необходимость в дешевом и быстром методе определения последовательности нуклеотидов в образце ДНК. Существующие устройства для чтения ДНК не позволяют считать за один раз всю молекулу ДНК. Вместо этого они позволяют читать фрагменты генома небольшой длины. Длина фрагмента может быть разной, она является важным параметром секвенирования – от нее напрямую зависит стоимость секвенирования и время, затрачиваемое на чтение одного фрагмента: чем больше длина считываемого фрагмента, тем выше стоимость чтения и тем дольше это чтение проис-

ходит. В связи с этим в настоящее время распространение получил следующий дешевый и эффективный подход: сначала выделяется случайно расположенный в геноме фрагмент длиной около 500 нуклеотидов, а затем считываются его префикс и суффикс (длиной порядка 80–120 нуклеотидов каждый). Эти префикс и суффикс называются *парными чтениями*. Описанный процесс повторяется такое число раз, чтобы обеспечить достаточно большое покрытие генома чтениями. Указанным образом работают, например, секвенаторы компании Illumina [1].

Отметим, что описанные выше префикс и суффикс читаются с разных нитей ДНК: один – с прямой, другой – с обратно-комплементарной, причем неизвестно, какой откуда. По этой причине удобно рассматривать геном и чтения, дополненные своими обратно-комплементарными копиями.

Задачей сборки генома является восстановление последовательности ДНК (ее длина составляет от миллионов до миллиардов нуклеотидов у разных живых существ) на основании информации, полученной в результате секвенирования. Этот процесс делится, как правило, на следующие этапы:

1. исправление ошибок в данных секвенирования;
2. сборка квазиконтигов – фрагментов, префиксы и суффиксы которых были получены на этапе секвенирования;
3. сборка контигов – максимальных непрерывных последовательностей нуклеотидов, которые удалось восстановить;
4. построение скэффолдов – последовательностей контигов, разделенных промежутками, для длин которых известны верхние и нижние оценки.

Одной из наиболее часто используемых при сборке генома математических моделей является граф де Брюина [2]. На его использовании основаны следующие программные средства сборки генома: Velvet [3], ALLPATHS [4], ABySS [5], SOAPdenovo [6], EULER [7].

Одним из недостатков, которым обладают перечисленные программные средства, является большой объем оперативной памяти, необходимый для сборки генома размером в миллиард нуклеотидов. Так, например, SOAPdenovo необходимо порядка 140 ГБ оперативной памяти, а ABySS – 21 компьютер с 16 ГБ оперативной памяти каждый (всего – 336 ГБ). Такие затраты памяти обусловлены наличием ошибок секвенирования в исходных данных, которые ведут к увеличению размера графа де Брюина, а также неоптимальным методом хранения этого графа.

Целью настоящей работы является разработка алгоритма сборки контигов геномной последовательности, использующего меньший объем оперативной памяти по сравнению с существующими. Входные данные для алгоритма представляют собой набор парных чтений, полученных на секвенаторе, а выходные данные – набор контигов.

Для исправления ошибок в данных секвенирования в настоящей работе используется алгоритм, описанный в работе [8]. Метод сборки контигов базируется на методе, предложенном в работе [9], для сборки бактериальных геномов размером в несколько миллионов нуклеотидов. Отличие предлагаемого метода от известного состоит в том, что предлагаемый метод рассчитан на сборку геномов в несколько миллиардов нуклеотидов. Построение скэффолдов в настоящей работе не рассматривается.

Архитектура метода сборки контигов

Сборка контигов в предлагаемом методе выполняется в два этапа.

1. Сборка квазиконтигов из чтений геномной последовательности.
2. Сборка контигов из квазиконтигов. Осуществляется с использованием графа перекрытий и метода Overlap–Layout–Consensus [10].

Сборка квазиконтигов из чтений геномной последовательности. Для сборки квазиконтигов осуществляется построение графа де Брюина, в котором множество ребер состоит только из «надежных» (k+1)-меров – тех, которые встречаются в чтениях достаточно большое число раз, не меньшее некоторого порогового значения, для того чтобы их можно было с очень большой вероятностью считать входящими в геном. Рассмотрим работу алгоритма на примере 10 пар чтений генома из 25 символов (рис. 1).

Если взять порог частоты для вхождения в граф равным единице, а k=3, то получится граф де Брюина, изображенный на рис. 2 (для простоты обратно-комплементарные ребра на рисунках не показаны).

Одним из важных свойств графа де Брюина является наличие в нем пути, соответствующего геному, при условии достаточного покрытия чтениями. В частности, это означает наличие пути для каждого из фрагментов, из которых были получены парные чтения. Предлагаемый метод сборки квазиконтигов основан на поиске таких путей.

Из всех путей, начало и конец которых совпадают с парой чтений, вызывают интерес только те, которые укладываются в априорные границы длин фрагментов, поэтому слишком короткие и слишком длинные пути можно отбросить. Оставшиеся пути – «хорошие» кандидаты на роль пути, соответствующего фрагменту в действительности. Если такой путь (рис. 3) – единственный, то можно с очень большой уверенностью сказать, что он соответствует реальной подстроке геномной последовательности, поэтому этот фрагмент считается восстановленным, а найденный путь выводится как квазиконтиг.

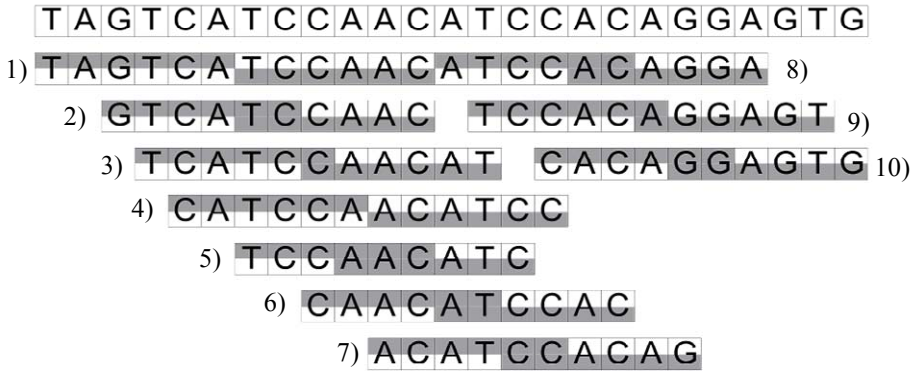

Рис. 1. Геном и его парные чтения

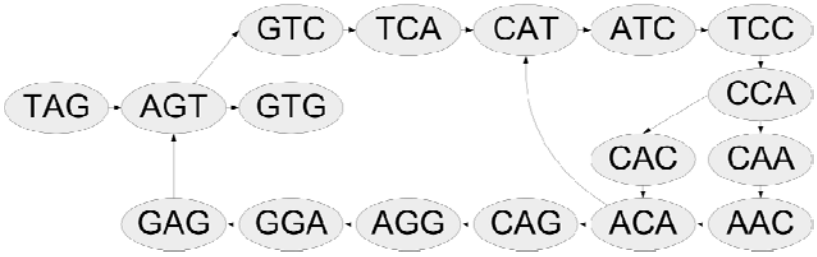

Рис. 2. Граф де Брюина при *k*=3

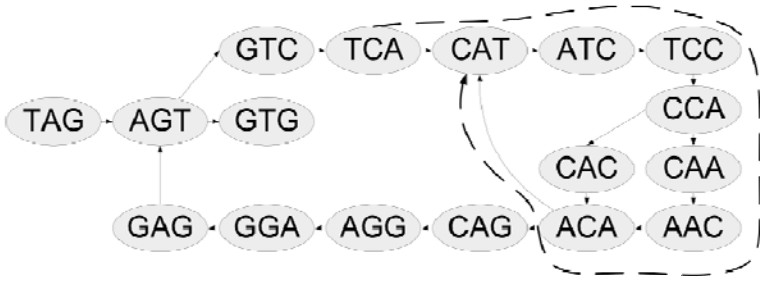

Рис. 3. Путь в графе де Брюина, соответствующий паре чтений номер 3

Если паре чтений в графе де Брюина соответствуют несколько путей (рис. 4), то из такой пары чтений квазиконтиг не генерируется.

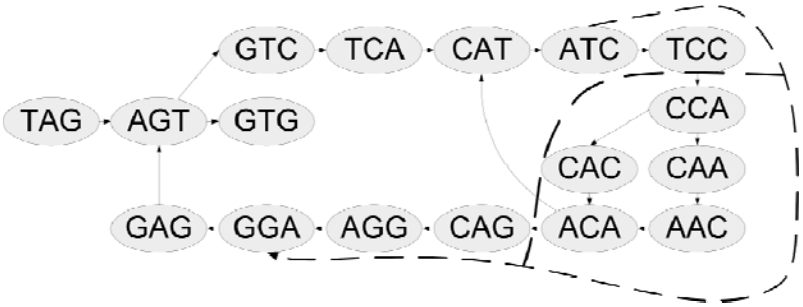

Рис. 4. Пути в графе де Брюина, соответствующие паре чтений номер 8

Приведем формальное описание алгоритма. Пусть путь p_1 длины l_1 ведет из вершины v_1 в вершину v_2, а путь p_2 длины l_2 ведет из вершины v_2 в вершину v_3. Будем обозначать конкатенацию этих путей, т.е. путь длины $l_1 + l_2$, соединяющий вершины v_1 и v_3, проходящий сначала по пути p_1, а затем – по p_2, как $p_1 \cdot p_2$. Рассмотрим множества путей P_1 и P_2.

С помощью $P_1 \cdot P_2$ будем обозначать все пути, которые можно получить конкатенацией путей p_1 и p_2 из P_1 и P_2 соответственно, т.е. $P_1 \cdot P_2 = \{p_1 \cdot p_2 \,|\, p_1 \in P_1, p_2 \in P_2\}$. Заметим, что конкатенировать можно только те пары путей p_1 и p_2, у которых последняя вершина p_1 совпадает с первой вершиной p_2. Например, если P_1 состоит из одного пути $v_1 \rightarrow v_2$, а множество P_2 состоит из путей $v_2 \rightarrow v_3$ и $v_4 \rightarrow v_5$, то множество $P_1 \cdot P_2$ будет состоять из одного пути $v_1 \rightarrow v_3$.

Задача, решаемая алгоритмом, состоит в поиске всех путей, соединяющих две заданные вершины v_1 и v_2, длины которых лежат в промежутке $[l_{\min};l_{\max}]$. Будем обозначать множество всех путей из v_1 в v_2 длины l как P^l, тогда искомое множество всех путей из v_1 в v_2 будет получаться объединением множеств P^l: $P = \bigcup_{l=l_{\min}}^{l_{\max}} P^l$.

Для выявления таких путей будем применять двунаправленный поиск [11] – одновременный поиск путей, ведущих из первой вершины, и путей, ведущих во вторую вершину. Это позволяет сократить время работы с $O(d^{l_{\max}})$ до $O(d^{l_{\max}/2})$, где d – средняя исходящая степень вершины графа (для графов де Брюина, встречающихся на практике, эта величина несущественно превышает единицу).

Применимость двунаправленного поиска объясняется тем, что любой путь p длины l из v_1 в v_2 можно разбить на два более коротких пути p_1 и p_2 длиной l_1 и l_2 так, чтобы выполнялись равенства $p = p_1 \cdot p_2$, $l = l_1 + l_2$.

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

На начальном шаге этого алгоритма l, l_1 и l_2 равны нулю, а P_1^0 и P_2^0 содержат по одному пути нулевой длины (эту пути состоят соответственно из вершин v_1 и v_2).

Если E – это множество всех ребер графа, то шаг в первом обходе осуществляется по формуле $P_1^{l_1+1} = P_1^{l_1} \cdot E$, а шаг во втором обходе по формуле $P_2^{l_2+1} = E \cdot P_2^{l_2}$.

Для того чтобы сократить потребление памяти при применении предлагаемого метода, необходимо иметь компактное представление используемого подграфа графа де Брюина. Для этого достаточно хранить только множество его ребер, что можно эффективно делать, используя, например, хеш-таблицу с открытой адресацией [12]. Преимуществами такого подхода хранения перед другими являются его простота в реализации и достаточно высокое быстродействие.

Также для уменьшения требуемого объема оперативной памяти используется то, что каждый (k+1)-мер входит в граф вместе с обратно-комплементарным. Тогда вместо пары (k+1)-меров s и s^{rc} можно хранить только один, определяемый по некоторому правилу – например, таким правилом может быть выбор лексикографически минимального (k+1)-мера. В этом случае необходимый объем памяти уменьшается примерно в 2 раза для четных k и ровно в 2 раза – для нечетных (только для четных k существуют обратно-комплементарные себе (k+1)-меры).

Сборка контигов из квазиконтигов. Сборка контигов из квазиконтигов основана на подходе Overlap-Layout-Consensus и состоит из нескольких этапов:

1. построение графа перекрытий между квазиконтигами;
2. уточнение графа перекрытий;
3. поиск контигов в графе перекрытий.

Для поиска перекрытий построим строку вида $C_1\$C_2\$C_3\$...C_n\$$, где C_i – i-й квазиконтиг, а n – число квазиконтигов. После этого необходимо построить суффиксный массив [13] для этой строки – отсортированный массив всех суффиксов строки. С его помощью можно найти все квазиконтиги, в которых встречается заданная подстрока. Это можно сделать, например, с помощью бинарного поиска суффиксов в суффиксном массиве, которые начинаются с заданной подстроки. Они будут располагаться рядом за счет сортировки.

Зафиксируем квазиконтиг, все перекрытия с которым требуется найти. Будем рассматривать его префиксы в порядке увеличения длины, начиная с минимального порога. Для каждого префикса будем проверять, входит ли такая подстрока с добавленным в конце \$ в суффиксный массив. Для того чтобы учесть еще и неточные перекрытия, проверяются не только сами префиксы, но и префиксы, в которые внесены небольшие изменения.

На следующем этапе происходят анализ найденных перекрытий, а также добавление и удаление перекрытий. Добавление происходит в случае, если квазиконтиг A перекрывается с квазиконтигами B и C, причем квазиконтиги B и C должны перекрываться, но такое перекрытие найдено не было. Удаление происходит, если квазиконтиг A перекрывается с квазиконтигом B, но B не похож на большинство квазиконтигов, с которыми перекрывается A.

Для поиска контигов выполняется поиск в ширину [12] в графе перекрытий. Он прерывается, если после текущего квазиконтига нет консенсуса – это означает, что квазиконтиги, перекрывающиеся с ним, различаются в большом числе позиций.

Экспериментальное исследование

Экспериментальные исследования разработанного метода проводились в рамках проекта Assemblathon 2, организованного университетом Калифорнии Санта-Круз [14]. Одним из наборов данных, который был подготовлен организаторами, являлся набор чтений рыбы Maylandia zebra. Размер генома этой рыбы оценивается примерно в 1 млрд нуклеотидов. Чтения геномной последовательности были разбиты на несколько библиотек, характеристика которых приведена в таблице.

Средний размер фрагмента в библиотеке	Покрытие
180	60
2500	62
5000	14
7000	16
9000	15
11000	12
40000	2,5

Таблица. Характеристика чтений генома рыбы Maylandia zebra

Общий объем исходных данных составлял 140 ГБ. Для сборки контигов использовалась только одна из библиотек чтений – со средним размером фрагмента в 180 нуклеотидов и 60-кратным покрытием. Алгоритмы сборки генома были реализованы на языке программирования Java. Для запуска программ использовался компьютер с 32 ГБ оперативной памяти и двумя 4-ядерными процессорами. Суммарное время работы всех трех этапов – исправления ошибок, сборки квазиконтигов и сборки контигов – составило 5 суток. Опишем подробнее результаты каждого из этапов.

Перед исправлением ошибок чтения были обрезаны таким образом, чтобы вероятность отдельной ошибки в каждом нуклеотиде не превышала 10%. После этого длина всех чтений в среднем уменьшилась на 20%. Исправление ошибок работало в течение 42 ч. В результате было найдено 150 млн исправлений. Всего было 600 млн. чтений, поэтому было исправлено в среднем каждое четвертое чтение.

Сборка квазиконтигов заняла 38 ч. Квазиконтиги были получены из 60% парных чтений.

Сборка контигов выполнилась за 26 ч. В результате было получено 734165 контигов, суммарный размер которых составляет 680×10^6 нуклеотидов. Длина максимального контига составляет 23514 нуклеотидов, средняя длина – 927, значение метрики N50 – 1799.

Отметим, что при использовании программного средства SOAPdenovo необходимый для сборки этого генома объем оперативной памяти составил 140 ГБ, а при использовании программного средства ABySS – 336 ГБ. Это позволяет говорить о том, что у разработанного метода требования к объему оперативной памяти существенного меньше.

Заключение

Предложен метод сборки контигов геномных последовательностей, основанный на совместном использовании графа де Брюина и графа перекрытий. Экспериментальное исследование этого метода проведено в рамках проекта Assemblathon 2. Это исследование показало, что разработанный метод обладает существенно меньшими требованиями к объему оперативной памяти, чем существующие.

Исследования выполнялись в рамках Федеральных целевых программ «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2013 годы» (государственный контракт № 07.514.11.4010) и «Научные и научно-педагогические кадры инновационной России на 2009–2013 годы» (государственный контракт № 16.740.11.0495).

Литература

1. Illumina, Inc. [Электронный ресурс]. – Режим доступа: http://www.illumina.com/, свободный. Яз. англ. (дата обращения 21.03.2012).
2. Pevzner P.A. 1-Tuple DNA sequencing: computer analysis // J. Biomol. Struct. Dyn. – 1989. – V. 7. – P. 63–73.
3. Zerbino D.R., Birney E. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs // Genome Research. – 2008. – V. 18. – P. 821–829.
4. Butler J., MacCallum I., Kleber M., Shlyakhter I.A., Belmonte M.K., Lander E.S., Nusbaum C., Jaffe D.B. ALLPATHS: De novo assembly of wholegenome shotgun microreads // Genome Research. – 2008. – V. 18. – P. 810–820.
5. Simpson J.T., Wong K., Jackman S.D., Schein J.E., Jones S.J., Birol I. ABySS: A parallel assembler for short read sequence data // Genome Research. – 2009. – V. 19. – P. 1117–1123.

6. Li R., Zhu H., Ruan J., Qian W., Fang X., Shi Z., Li Y., Li S., Shan G., Kristiansen K. et al. De novo assembly of human genomes with massively parallel short read sequencing // Genome Research. – 2010. – V. 20. – P. 265–272.
7. Pevzner P.A., Tang H., Waterman M.S. EULER: An Eulerian path approach to DNA fragment assembly // Proc. Natl. Acad. Sci. – 2001. – V. 98. – P. 9748–9753.
8. Александров А.В., Казаков С.В., Мельников С.В., Сергушичев А.А., Царев Ф.Н., Шалыто А.А. Метод исправления ошибок в наборе чтений нуклеотидной последовательности // Научно-технический вестник СПбГУ ИТМО. – 2011. – № 5 (75). – С. 81–84.
9. Исенбаев В.В. Разработка системы секвенирования ДНК с использованием paired-end данных. Бакалаврская работа. СПбГУ ИТМО, 2010 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/genom/_isenbaev_thesis.pdf, свободный. Яз. англ. (дата обращения 21.03.2012).
10. International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome // Nature. – 2001. – V. 409. – № 6822. – P. 860–921.
11. Рассел С., Норвиг П. Исскуственный интеллект: современный подход: Пер. с англ. – 2-е изд. – М.: Вильямс. 2006. – 1408 с.
12. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. – М.: Вильямс, 2011. – 1296 с.
13. Гасфилд Д. Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология. – СПб: Невский диалект, 2003. – 654 с.
14. Проект Assemblathon 2 [Электронный ресурс]. – Режим доступа: www.assemblathon.org, свободный. Яз. англ. (дата обращения 21.03.2012).

Александров Антон Вячеславович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, alantbox@gmail.com

Казаков Сергей Владимирович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, kazakov_sergey_v@mail.ru

Мельников Сергей Вячеславович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, melnikov@rain.ifmo.ru

Сергушичев Алексей Александрович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, alsergbox@gmail.com

Царев Федор Николаевич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, аспирант, fedor.tsarev@gmail.com