7. *Leach D., Walsh M.* A characterization of lattice-ordered graphs // Proc. Integers Conf. 2005. N. Y. : Gruyter, 2007. P. 327–332.

8. *Салий В. Н.* Система абстрактных связных подграфов линейного графа // Прикладная дискретная математика. 2012. № 2(16). С. 90–94.

The Ordered Set of Connected Parts of a Polygonal Graph

V. N. Salii

Saratov State University, Russia, 410012, Saratov, Astrahanskaya st., 83, SaliiVN@info.sgu.ru

Under a polygonal graph is meant an oriented graph obtained from a cycle by some orientation of its edges. The set of all abstract (i. e. pairwise non-isomorphic) connected parts of a polygonal graph is ordered by graph embedding. Polygonal graphs are characterized for which this ordered set is a lattice.

Key words: polygonal graph, linear graph, binary vector, duality, ordered set, lattice.

References

1. Salii V. N. Minimal primitive extensions of oriented graphs. *Prikladnaya diskretnaya matematika*, 2008, no. 1(1), pp. 116–119 (in Russian).

2. Trotter W. T., Moore J. I. Some theorems on graphs and posets. *Discrete Math.*, 1976, vol. 15, no. 1, pp. 79–84.

3. Jacobson M. S., Kézdy F. E., Seif S. The poset of connected induced subgraphs of a graph need not be Sperner. *Order*, 1995, vol. 12, no. 3, pp. 315–318.

4. Kézdy A. E., Seif S. When is a poset isomorphic to the poset of connected induced subgraphs of a graph? *Southwest J. Pure Appl. Math.*, 1996, vol. 1, pp. 42–50. Available at: http://rattler.cameron.edu/swjpam.html (Accessed 28, September, 2012).

5. Nieminen J. The lattice of connected subgraphs of a connected graph. *Comment. Math. Prace Mat.*, 1980, vol. 21, no. 1, pp. 187–193.

6. Adams P., Eggleton R. B., MacDougall J. A. Degree sequences and poset structure of order 9 graphs. *Proc. XXXV Southeast Conf. Comb., Graph Theory and Computing*. Boca Raton, FL, USA, 2004, vol. 166, pp. 83–95.

7. Leach D., Walsh M. A characterization of lattice-ordered graphs. *Proc. Integers Conf.*, 2005. New York, Gruyter, 2007, pp. 327–332.

8. Salii V. N. The system of abstract connected subgraphs of a linear graph. *Prikladnaya diskretnaya matematika*, 2012, no. 2(16), pp. 90–94 (in Russian).

УДК 004.021

СОВМЕСТНОЕ ПРИМЕНЕНИЕ ГРАФА ДЕ БРЁЙНА, ГРАФА ПЕРЕКРЫТИЙ И МИКРОСБОРКИ ДЛЯ *DE NOVO* СБОРКИ ГЕНОМА

А. А. Сергушичев[1], А. В. Александров[2], С. В. Казаков[3], Ф. Н. Царев[4], А. А. Шалыто[5]

[1]Магистрант кафедры компьютерных технологий, Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, alserg@rain.ifmo.ru

[2]Магистрант кафедры компьютерных технологий, Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, alexanrdov@rain.ifmo.ru

[3]Магистрант кафедры компьютерных технологий, Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, svkazakov@rain.ifmo.ru

[4]Кандидат технических наук, ассистент кафедры программной инженерии и верификации программ, Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, tsarev@rain.ifmo.ru

[5]Доктор технических наук, заведующий кафедрой технологий программирования, профессор, Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, shalyto@mail.ifmo.ru

В работе предлагается метод сборки контигов геномных последовательностей из парных чтений. Особенностью этого метода является разбиение процесса сборки контигов на три этапа: сборка квазиконтигов из чтений, сборка контигов из квазиконтигов и микросборка. На первом из этапов используется граф де Брёйна, на втором — граф перекрытий. Описываются результаты экспериментального исследования разработанного метода на чтениях геномов бактерии *E. Coli* (размер генома — 4.5 миллиона нуклеотидов) и рыбы *Maylandia zebra* (размер генома — миллиард нуклеотидов). Преимущество разработанного метода состоит в том, что для его работы требуется существенно меньше оперативной памяти по сравнению с существующими программными средствами для сборки генома.

Ключевые слова: сборка генома, контиги, граф де Брёйна, граф перекрытий, микросборка.

ВВЕДЕНИЕ

Многие современные задачи биологии и медицины требуют знания геномов живых организмов, который состоит из нескольких нуклеотидных последовательностей молекул дезоксирибонуклеиновой кислоты (ДНК). Поэтому возникает необходимость в дешевом и быстром методе секвенирования — определения последовательности нуклеотидов в образце ДНК. Существующие секвенаторы — устройства для чтения ДНК — не позволяют считать за один раз всю молекулу. Вместо этого они позволяют читать фрагменты генома небольшой длины. В настоящее время получил распространение следующий подход: сначала большое число копий ДНК разбивается (например, с помощью ультразвука) на много маленьких фрагментов длиной около 500 нуклеотидов, а затем у фрагментов считываются их префикс и суффикс (длиной до 80–120 нуклеотидов каждый). Эти префикс и суффикс называются парными чтениями. Описанный процесс заканчивается, когда суммарное число прочитанных нуклеотидов превосходит размер генома в несколько десятков раз (50–200). 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, который откуда.

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.

Процесс сборки делится, как правило, на следующие этапы:

1. Исправление ошибок в данных секвенирования.
2. Сборка контигов — максимальных непрерывных последовательностей нуклеотидов, которые удалось восстановить.
3. Построение скэффолдов — упорядоченных множеств контигов, разделенных промежутками, на размеры которых есть какие-то оценки.

Секвенаторы первого поколения выдавали на выходе длинные чтения. Это позволяло при небольшом покрытии (около 10-кратного) получать достаточно длинные и качественные контиги. Вследствие этого было разработано много сборщиков, основанных на подходе *overlap-layout-consensus* (*OLC*) [2]. Этот подход использует поиск перекрытий, не меньших некоторой заданной длины, между всеми чтениями и построение графа перекрытий — графа, вершинами которого являются чтения, а ребрами — найденные перекрытия.

Секвенаторы второго поколения позволили существенно удешевить процесс секвенирования, но длина чтений уменьшилась. При использовании такой технологии большое число чтений и их небольшая длина не позволяет эффективно использовать метод *OLC*. Одной из наиболее часто используемых математических моделей для сборки генома из таких данных является граф де Брёйна [3]. Вершинами этого графа являются строки из нуклеотидов длины k — k-меры; ребрами являются $(k+1)$-меры; ребро соединяет вершины, соответствующие его префиксу и суффиксу. На использовании этого графа основано множество соверменных средств для сборки генома: *Velvet* [4], *Allpaths* [5], *AbySS* [6], *SOAPdenovo* [7], *EULER* [8]. Одним из недостатков, которым обладают перечисленные программные средства, является большой объем оперативной памяти, необходимый им для сборки генома, сходного по размерам с геномом человека (2–3 миллиарда нуклеотидов). Так, например, *SOAPdenovo* необходимо около 140 ГБ оперативной памяти, а *ABySS* — 21 компьютер с 16 ГБ каждый (всего — 336 ГБ). Такие затраты памяти обусловлены наличием ошибок секвенирования в исходных данных (такие ошибки ведут к увеличению размера графа де Брёйна), а также неэкономным методом хранения этого графа.

В настоящей работе предлагается метод, ориентированный на низкое использование памяти. Исправление ошибок в небольшом объеме памяти было рассмотрено в работе [9]. Построение скэффолдов в настоящей работе не рассматривается.

1. ПРЕДЛАГАЕМЫЙ МЕТОД

Этап сборки контигов в предлагаемом методе состоит из трех подэтапов.

1. Сборка квазиконтигов из чтений геномной последовательности. Квазиконтигами называются последовательности нуклоетидов, которые, с одной стороны, длиннее чтений, но, с другой стороны, все еще являются достаточно короткими. Этот подэтап выполняется с использованием графа де Брёйна.

2. Сборка контигов из квазиконтигов. Выполняется с использованием графа перекрытий и метода *overlap-layout-consensus*.
3. Построение контигов с помощью микросборки. Использует граф де Брёйна и граф, сходный с графом перекрытий.

Сборка квазиконтигов. В первом подэтапе используется граф де Брёйна, 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 достаточно большой вероятностью считать входящими в геном (как правило, это значение составляет от трех до шести). Множество вершин состоит из тех вершин графа де Брёйна, которым инцидентно хотя бы одно из выбранных ребер. Если участок нуклеотидной последовательности покрылся достаточно хорошо, то все входящие в него $(k+1)$-меры по много раз входят в исходные данные, а тогда в этом графе существует путь между первым и последним k-мерами участка. Предлагаемый метод основан на поиске такого пути для фрагмента, соответствующего парным чтениям. Чтобы оба чтения находились на одной цепочке, второе чтение заменяется комплементарной ему последовательностью. Чтения при этом направленны внутрь, поэтому исходному фрагменту соответствует путь между первым k-мером первого чтения и обращенным первым k-мером второго чтения. В дальнейшем считаем, что операция комплементирования чтения уже произведена и оба чтения находятся на одной нити.

Известно, что длины фрагментов, из которых были получены парные чтения, имеют распределение, близкое к нормальному, поэтому для длин путей существуют верхняя и нижняя границы, и, следовательно, слишком короткие и слишком длинные пути можно отбросить. Если нашелся единственный путь, то можно с очень большой уверенностью сказать, что он соответствует реальной подстроке геномной последовательности, поэтому этот фрагмент считается восстановленным, а найденный путь выводится. Возможен также вариант, когда нашлось несколько путей одинаковой длины. В случае, если эти пути достаточно похожи, в неоднозначных позициях можно выводить любой из нуклеотидов, так как на следующем подэтапе ищутся в том числе и неточные перекрытия.

Для поиска путей применяется подход *meet-in-the-middle* — пути ищутся одновременно с двух сторон с помощью двух одновременных обходов в ширину. Первый обход идет по прямым ребрам графа де Брёйна начиная от первого чтения в паре, а второй — по обратным, начиная от второго чтения. Таким образом, после l_1 шагов в первом обходе получается слой вершин L_1, удаленных от начальной вершины на l_1, а после l_2 шагов во втором обходе — слой вершин L_2, удаленных от конечной вершины на l_2. Если в какой-то момент слои L_1 и L_2 вершинно пересекаются, то существуют путь между начальной и конечной вершиной длины l_1+l_2, которому соответствует последовательность длины l_1+l_2+k. Используя деревья обходов в ширину, этот путь можно восстановить.

В обходах в ширину в качестве начальной и конечной вершин выбираются соответственно первый k-мер первого чтения и обращенный первый k-мер второго чтения. При этом, чтобы не потерять возможность собрать квазиконтиг даже при ошибках в этих k-мерах, на шаге l_1 первого обхода в множество L_1 добавляется l_1-й k-мер первого чтения, а на шаге l_2 второго обхода в множество L_2 добавляется обращенный l_2-й k-мер второго чтения. Из-за этого пересечение множеств L_1 и L_2 может на самом деле давать путь короче l_1+l_2.

Можно также заметить, что путей длины l_1 от начальной вершины может быть много, про часть из которых можно сказать, что маловероятно, что они являются подпутями искомого пути. Для этого можно сравнивать каждый путь длины l_1 с префиксом первого чтения длины l_1+k (пока такой префикс существует). Если путь отличается от префикса чтения в большом числе позиций с маленькой вероятностью ошибки, то такой путь можно не рассматривать — удалить соответствующую вершину из L_1. Аналогично удаляются некоторые вершины из L_2.

Важным моментом является то, в каком порядке делать шаги в обходах в ширину. Имеет смысл их каким-то образом чередовать, так как число посещенных вершин в каждом обходе в ширину асимптотически экспоненциально зависит от числа шагов в этом обходе. Два одновременных обхода в ширину навстречу друг другу позволяют сократить число посещенных вершины, а, стало быть, и операций, с примерно d^l до $2d^{l/2}$, где d — средняя степень вершины в графе де Брёйна, а l — примерная длина искомого пути. На практике можно делать попеременное выполнение шагов, либо, что еще лучше, делать шаг в том обходе, в котором в текущем слое меньше вершин.

На рис. 1 показан пример работы алгоритма поиска путей. В этом примере были найдены пути разной длины, что означает, что из этой пары чтений собрать квазиконтиг не получается.

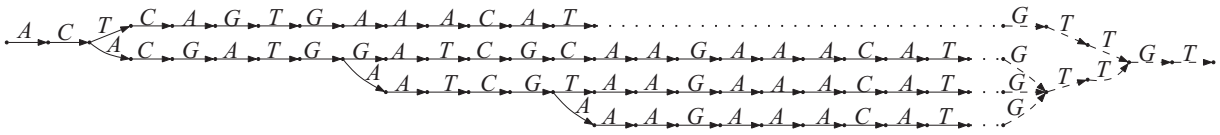

Рис. 1. Пример фрагмента графа, получившегося при поиске путей. Сплошные и штриховые пунктирные ребра — ребра, добавленные в ходе первого и второго обходов в ширину соответственно. Точечные пунктирные ребра соединяют вершины, соответствующие двум одинаковым k-мерам

Выполнение обходов в ширину заканчивается в одном из трех случаев:

1) сумма $l_1 + l_2 + k$ достигла верхней границы на длину пути;

2) было найдено два соединяющих начальную и конечную вершины пути разной длины;

3) число посещенных вершин стало слишком большим.

Для того чтобы потребление памяти при применении предлагаемого метода было не очень большим, необходимо иметь компактное представление используемого графа де Брёйна. Для этого достаточно хранить только множество его ребер, что можно эффективно делать, используя, например, хеш-таблицу с открытой адресацией. Преимуществами такого подхода хранения перед другими являются его простота, быстродействие и возможность балансировки между используемой памятью и скоростью. Более эффективными с точки зрения потребляемой памяти являются *rank/select* словари [10]. Необходимый для них объем оперативной памяти близок к теоретическому пределу, однако время доступа к данным выше (в том числе асимптотически), чем у хеш-таблиц. Другим вариантом, предложенным в статье [11], является применение фильтров Блума с хранением небольшого числа ложноположительных ребер в хеш-таблице, что позволяет сохранить асимптотическое время доступа $O(1)$, при близком к оптимальному использовании памяти. Этот подход может быть применен и в предлагаемом алгоритме сборки квазиконтигов.

Стоит отметить, что для работы предложенного метода не требуется, чтобы парные чтения перекрывались, в отличие от некоторых других подобных подходов.

Сборка контигов из квазиконтигов. Как уже было сказано выше, сборка контигов из квазиконтигов основана на подходе *overlap-layout-consensus*. На этом подэтапе сначала производится поиск перекрытий между квазиконтигами. Для этого строится строка вида «$C_1\$C_2\$C_3\$...C_n\$$», где C_i — i-й квазиконтиг, а n — число квазиконтигов. Затем для этой строки строится суффиксный массив [12] — массив суффиксов строки, упорядоченных лексикографически. С его помощью можно найти все квазиконтиги, заканчивающиеся на заданную строку s. Это можно сделать, например, с помощью бинарного поиска суффиксов в суффиксном массиве, которые начинаются с s. Так как из-за сортировки они будут располагаться рядом, то получится интервал суффиксов в суффиксном массиве. В начале этого интервала будут располагаться те суффиксы, в которых после s идет «\$». Каждый из этих суффиксов соответствует квазиконтигу, который заканчивается на s. Если в качестве строки s брать префиксы квазиконтига, то описанным способом можно найти квазиконтиги, с которым они перекрываются. Если к ним добавить префиксы с небольшим изменением, то можно также находить неточные перекрытия.

Для хранения суффиксного массива в памяти используется примерно 3 бита на нуклеотид при хранении исходной строки и 5 байт на элемент суффиксного массива — позицию, с которой начинается суффикс. Чтобы сократить использование памяти, применяется разбиение суффиксов на корзины в зависимости от их префикса небольшой фиксированной длины.

Для построения суффиксного массива применяется корзинная сортировка с последующим применением быстрой сортировки для достаточно небольших участков массива. Так как все суффиксы в одной корзине начинаются с одинакового префикса, каждую корзину можно сортировать независимо, загружая в память только исходную строку (полностью) и массив суффиксов из текущей корзины, заданных их позициями в строке.

Для того чтобы при поиске перекрытий не выполнять бинарного поиска многократно для одних и тех же параметров, поиск перекрытий осуществляется одновременно для группы квазиконтигов. Будем поддерживать два интервала: интервал $Q = (q_l, q_r)$, соответствующий квазиконтигам с одинаковым префиксом некоторой длины l, для которых в данный момент ищутся перекрытия, и интервал $S = (s_l, s_r)$, соответствующий суффиксам с одинаковым префиксом той же длины l. При этом префиксы квазиконтигов из Q и суффиксов из S различаются лишь в небольшом числе позиций. На

каждом шаге сначала проверяется, есть ли среди суффиксов из S такие, что на $(l+1)$-й позиции в них находится «\$». Такие суффикксы соответствуют перекрытиям квазиконтигов из Q с некоторыми другими квазиконтигами. Затем интервал квазиконтигов Q и интервал суффиксов S делятся на четыре интервала Q_A, Q_G, Q_C, Q_T и S_A, S_G, S_C, S_T соответственно в зависимости от $(l+1)$-го нуклеотида. Функция поиска перекрытий рекурсивно вызывается для всех 16 пар интервалов с учетом того, что несовпадение $(l+1)$-х символов может дать различие префиксов, превыщающее установленный порог на число возможных несовпадений в перекрытии.

После того как все перекрытия были найдены, получается граф, в котором вершинами являются квазиконтиги, а ребрами — перекрытия. Этот граф упрощается: из него удаляются транзитивные перекрытия — такие перекрытия между контигами A и C, что существует квазиконтиг B, который перекрывается и с квазиконтигом A, и с квазиконтигом C (рис. 2).

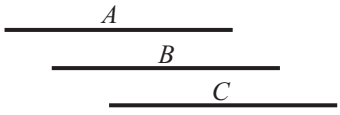

Рис. 2. Транзитивное перекрытие квазиконтигов A, B и C

В конце подэтапа полученный граф перекрытий модифицируется с помощью набора эвристик: удаление квазиконтигов, не имеющих перекрытий с другими, удаление небольших «отростков» в графе, объединение путей одинаковой длины и др. Эти модификации применяются для того, чтобы избавиться от ошибок, которые могли быть в квазиконтигах. Затем в этом графе выделяются пути без ветвлений, которые соответствуют контигам. Для того чтобы из последовательности перекрывающихся квазиконтигов получить последовательность контигов, для каждой позиции контига выбирается нуклеотид, который наиболее часто встречается в соответсвующих позициях квазиконтигов, покрывающих эту позицию контига.

Микросборка. В подэтапе микросборки сначала определяются пары контигов, которые могут быть расположены в геноме рядом. Для этого все чтения картируются на контиги с помощью программного средства *Bowtie*. Находятся все парные чтения, картированные на разные контиги. Такие парные чтения называются соединяющими. Для каждой пары соединенных таким образом контигов определяется, в каком порядке они идут в геноме, и выбираются все пары чтений, хотя бы одно из которых картируется на эти контиги. Из этих чтений строится граф де Брёйна (рис. 3). Его размер существенно меньше графа, используемого на этапе сборки квазиконтигов (поэтому этап называется микросборкой). Путем поиска путей в этом графе промежуток между контигами заполняется, тем самым они объединяются.

На рис. 3 контиги A и B предположительно расположены в геноме рядом, так как существует пара чтений (a_1, a_2), в которой a_1 входит в первый контиг, а a_2 — во второй. Пары чтений (b_1, b_2) и (c_1, c_2) не соединяют эти контиги, но используются для построения графа де Брёйна. Этот граф имеет меньший размер, чем в подэтапе сборки квазиконтигов, — при сборке квазиконтигов граф имеет размер порядка длины генома, который может состоять из миллиардов нуклеотидов, а при микросборке — порядка размера контигов или даже размера фрагмента (при некоторой оптимизации), то есть порядка тысяч нуклеотидов. За счет этого удается восстановить больше пар чтений.

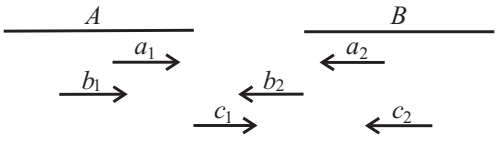

Рис. 3. Пары чтений, соединяющие контиги и использующиеся для построения графа де Брёйна

После того как некоторые пары контигов получилось соединить, может оказаться, что после некоторого контига A есть несколько вариантов контигов, которые могут идти после него. В некоторых случаях это соответствует тому, что контиг A является повтором и встречается в геноме несколько раз. Возможен и другой вариант, когда после контига A могут идти контиги B и C, притом что после контига B может идти контиг C. Этот случай может соответствовать последовательному расположению контигов A, B и C в геноме. Чтобы учесть такие случаи, строится граф, похожий на граф перекрытий, в котором вершинами являются контиги, а ребрам соответствуют соединения между контигами — длина промежутка (в частности, она может быть отрицательной, если контиги перекрываются) и последовательность нуклеотидов, заполняющая этот промежуток. В этом графе пути одинаковой длины объединяются, после чего неветвящиеся пути в этом графе выводятся как новая версия контигов.

2. ЭКСПЕРИМЕНТЫ

Предложенные алгоритмы были реализованы на языке программирования *Java*. Экспериментальные исследования разработанного метода проводились для генома бактерии *E. Coli* (размер генома — примерно 4,5 миллиона нуклеотидов) и рыбы *Maylandia zebra* (размер генома — примерно миллиард нуклеотидов). Для сборки генома *E. Coli* использовалась библиотека парных чтений *SRR001665* со средним размером фрагмента около 200 нуклеотидов, с размером чтения 36 нуклеотидов и суммарным 160-кратным покрытием.

Сборка генома *E. Coli* проводилась на компьютере с 16 ГБ оперативной памяти и 6-ядерным процессором *AMD Phenom™ II X6 1090T*. Размер памяти, доступный виртуальной машине *JVM*, составлял 1 ГБ. При сборке контигов после первого подэтапа было получено 10 миллионов квазиконтигов — из большинства пар чтений было получено по квазиконтигу. Перед вторым подэтапом часть самых коротких квазиконтигов была отброшена, суммарный размер оставшихся составил 175 миллионов нуклеотидов. После второго подэтапа было получено 525 контигов со значением метрики $N50$, равным 17804 и максимальным размером, равным 73908. После подэтапа микросборки было получено 247 контигов со значением метрики $N50$, равным 53720, и максимальным размером контига равным 167319. Покрытие генома контигами составило 98%. Эти результаты примерно соответствуют результатам других сборщиков, что говорит о применимости предложенного метода.

Сборка генома *Maylandia zebra* проводилась в рамках проекта *Assemblathon 2* [13], организованного Калифорнийским университетом в Дэвисе (*University of California, Davis*). Для сборки контигов использовался набор чтений со средним размером фрагмента 180 и 60-кратным покрытием. Для запуска программ использовался компьютер с 32 ГБ оперативной памяти и двумя 4-ядерными процессорами. Суммарное время работы составило пять суток. Перед исправлением ошибок чтения были обрезаны, 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 часов. В результате было найдено 150 миллионов исправлений. Всего чтений было 600 миллионов, поэтому было исправлено в среднем каждое четвертое чтение. Сборка квазиконтигов заняла 38 часов. Квазиконтиги были получены из 60% чтений. Сборка контигов выполнилась за 26 часов. В результате было получено 734165 контигов, суммарный размер которых составляет 680321319 нуклеотидов. Длина максимального составляет 23514 нуклеотидов, средняя длина — 927, значение метрики $N50$ — 1799. Микросборка не повлияла на окончательный результат. Возможной причиной этого может являться то, что при покрытии генома контигами около 80–85% (примерно такая часть чтений была найдена в контигах) и среднем размере контига около 1000, среднее расстояние между «соседними» контигами составляет около 250, а парные чтения со средним размером фрагмента около 180 позволяют соединить контиги с промежутком примерно 120.

Исследования выполняются в рамках соглашения № 14.B37.21.0562 и государственного контракта № 16.740.11.0495 (заключены в рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России на 2009–2013 годы»).

Библиографический список

1. Illumina, Inc. URL: http://www.illumina.com/ (дата обращения : 18.05.2012).

2. *Böckenhauer H.-J., Bongrratz D.* Algorithmic Aspects of Bioinformatics. Berlin : Springer, 2007. 396 p.

3. *Pevzner P. A.* 1-Tuple DNA sequencing : computer analysis // J. Biomol. Struct. Dyn. 1989. Vol. 7. P. 63–73.

4. *Zerbino D. R., Birney E.* Velvet : Algorithms for de novo short read assembly using de Bruijn graphs // Genome Research. 2008. Vol. 18. P. 821–829.

5. *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. Vol. 18. P. 810–820.

6. *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. Vol. 19. P. 1117–1123.

7. *Li R., Zhu H., Ruan J., Qian W., Fang X., Shi Z., Li Y., Li S., Shan G., Kristiansen K., Li S., Yang H., Wang J., Wanget J.* De novo assembly of human genomes with massively parallel short read sequencing // Genome Research. 2010. Vol. 20. P. 265–272.

8. *Pevzner P. A., Tang H., Waterman M. S.* EULER : An Eulerian path approach to DNA fragment assembly // Proc. Natl. Acad. Sci. 2001. № 98. P. 9748–9753.

9. *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 // Науч.-техн. вестн. С.-Петерб. гос. ун-та информационных технологий, механики и оптики. 2011. № 5. С. 81–84.

10. *Okanohara D., Sadakane K.* Practical entropy-compressed rank/select dictionary // Computing Research Repository. 2006. URL: http://arxiv.org/abs/cs/0610001 (дата обращения : 18.05.2012).

11. *Chikhi R., Rizk G.* Space-efficient and exact de Bruijn graph representation based on a Bloom filter // Algorithms in Bioinformatics. 2012. P. 236–248.

12. *Гасфилд Д.* Строки, деревья и последовательности в алгоритмах. Информатика и вычислительная биология. СПб. : Невский диалект, 2003. 656 с.

13. The Assemblathon. URL: http://www.assemblathon.org (дата обращения : 18.05.2012).

Combining De Bruijn Graphs, Overlap Graphs and Microassembly for *De Novo* Genome Assembly

A. A. Sergushichev, A. V. Alexandrov, S. V. Kazakov, F. N. Tsarev, A. A. Shalyto

Saint-Petersburg National Research University of Information Technologies, Mechanics and Optics, Russia, 197101, St. Petersburg, Kronverkskiy pr., 49, alserg@rain.ifmo.ru, alexanrdov@rain.ifmo.ru, svkazakov@rain.ifmo.ru, tsarev@rain.ifmo.ru, shalyto@mail.ifmo.ru

In this paper we present a method for *de novo* genome assembly that splits the process into three stages: quasicontig assembly; contig assembly from quasicontigs; contig postprocessing with microassembly. The first stage uses de Bruijn graph, the second one uses overlap graph. We have carried out experiments of assembling the *E. Coli* genome (size ≈ 4.5 Mbp) and *Maylandia zebra* genome (size ≈ 1 Gbp). Advantage of proposed method is a low memory consumption.

Key words: genome assembly, contigs, de Bruijn graph, overlap graph, microassembly.

References

1. *Illumina, Inc.* Available at: http://www.illumina.com/ (Accessed 18, May, 2012).

2. Böckenhauer H.-J., Bongrratz D. *Algorithmic Aspects of Bioinformatics*. Springer, 2007, 396 p.

3. Pevzner P. A. 1-Tuple DNA sequencing: computer analysis. *J. Biomol. Struct. Dyn.* 1989. vol. 7, pp. 63–73.

4. Zerbino D. R., Birney E. Velvet : Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 2008, vol. 18, pp. 821–829.

5. 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, vol. 18, pp. 810–820.

6. 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, vol. 19, pp. 1117–1123.

7. Li R., Zhu H., Ruan J., Qian W., Fang X., Shi Z., Li Y., Li S., Shan G., Kristiansen K., Li S., Yang H., Wang J., Wanget J. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 2010, vol. 20, pp. 265–272.

8. Pevzner P. A., Tang H., Waterman M. S. EULER : An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci.*, 2001, no. 98, pp. 9748–9753.

9. Aleksandrov A. V., Kazakov S. V., Melnikov S. V., Sergushichev A. A., Tsarev F. N., Shalyto A. A. Errors Correction Method in the Readings Set of Nucleotide Sequence. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2011, no. 5, pp. 81–84 (in Russian).

10. Okanohara D., Sadakane K. Practical entropy-compressed rank/select dictionary. *Comput. Research Reposit.*, 2006. Available at: http://arxiv.org/abs/cs/0610001 (Accessed 18, May, 2012).

11. Chikhi R., Rizk G. Space-efficient and exact de Bruijn graph representation based on a Bloom filter. *Algorithms in Bioinformatics*, 2012, pp. 236–248.

12. Gusfield D. *Algorithms on String, Trees and Sequences. Computer Science and Computational Biology*. Cambridge Univ. Press, 1997, 554 p. (Rus. ed.: Gusfild D. *Stroki, derev'ia i posledovatel'nosti v algoritmakh. Informatika i vychislitel'naia biologiia*. St. Petersburg, Nevskii dialekt Publ., 2003, 656 p.).

13. *The Assemblathon*. Available at: http://www.assemblathon.org (Accessed 18, May, 2012).