Genome Assembly

Студенты:

- Нгуен Тхань Тхиен
- Фам Куанг Ань
- Май Суан Бать
- Хоанг Хиеп

Группа: ИУ7и-66Б

Преподаватель: Строганов Ю. В.

Москва
2021

Введение

Определение последовательности нуклеотидов (As, Ts, Cs и Gs) в гене (также известном как ДНК) или секвенирование генома - одна из фундаментальных задач биоинформатики. Наш собственный геном составляет около 3 миллиардов нуклеотидов.

В фундаментальных исследованиях секвенирование может помочь в построении хромосомных карт, получении информации о структуре, функциях, активности фрагментов ДНК, идентификации видов и т. д.

Цель и задачи

Цели работы:

- Написать приложение для сборки генома из ридов с помощью графа Де Брёйна (рид или read — отдельная последовательность нуклеотидов, полученная в результате секвенирования);
- Улучшить знания функционального программирования.

Задачи проекта:

- Провести анализ предметной области;
- Выбрать инструменты и технологии для разработки;
- Спроектировать архитектуру для частей программного комплекса;
- Разработать и протестировать программный комплекс.

Секвенирование генома

История развития секвенирования генов ного поколения

1-е поколение : метод Фредерика Сангера

2-е поколение : метод пиросеквенирования, ионное полупроводниковое секвенирование, метод Illumina/Solexa

3-е поколение : одномолекулярное секвенирование в реальном времени (SMRT)

4-е поколение : нанопоровое секвенирование

Сборка генома

Сборка генома — процесс объединения большого числа ридов в одну (хромосома) или несколько (контиги, скаффолды) длинных последовательностей, в результате чего должна быть восстановлена исходная последовательность генома.

Реальные методы сборки

- OLC: сборка Overlap-Layout-Consensus
- DBG: сборка графа Де Брёйна

Оба обрабатывают неразрешимые повторы, по существу исключая их.

Фрагменты - это контиги (сокращение от contiguous).

Неразрешимые повторы разбивают сборку на фрагменты.

Графа Де Брёйна

- k-мер: последовательность из k нуклеотидов (мер: от греческого означает «часть»)

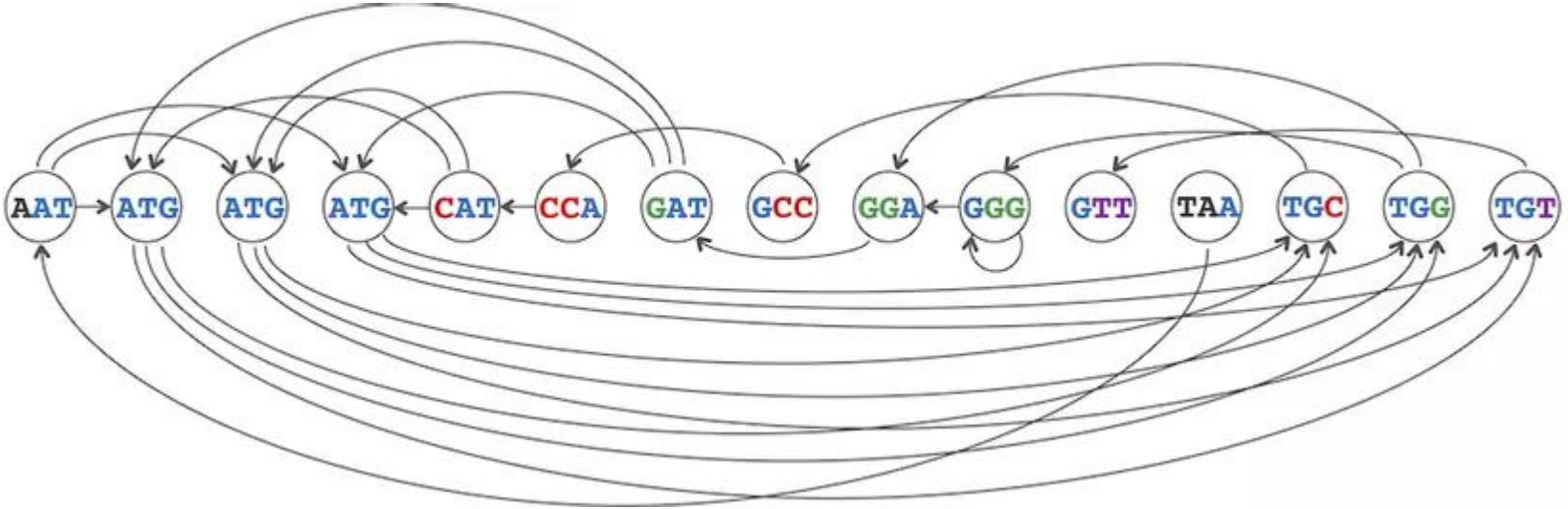
Пример : все k-меры рида
$(TATGGGGTGC) = ATG, GGG, GGG, GGT, GTG, TAT, TGC, TGG$

- Вершины графа де Брюйна: все k-меры
- Рёбра графа де Брюйна: все (k+1)-меры
- Ребро e соединяет префикс и суффикс e

Решить проблему нахождения набора k-мер из последовательности просто, но для сборки генома нам нужна его обратная задача: построение последовательности из k-мер.

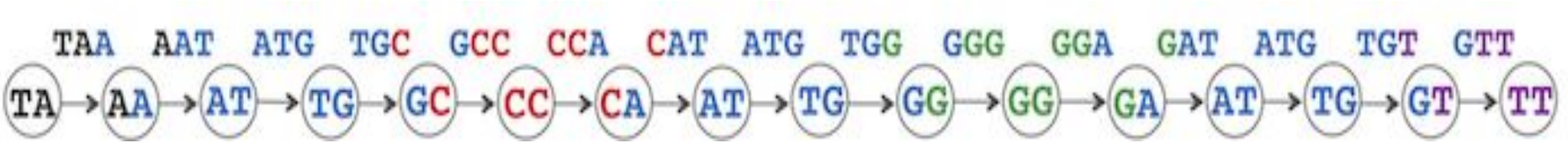
Графа Де Брёйна

Предположим, благодаря секвенированию Shotgun мы получаем все k-меры небольшого генома. Наша проблема заключается в нахождении пути Гамильтона для графа. Гамильтонов путь является простым путём (путём без петель), проходящим через каждую вершину графа ровно один раз. Решение этой задачи пока неизвестно.



Графа Де Брёйна

Путем преобразования каждой вершины графа в ребро с 2 новыми вершинами, являющимися (k-1) мерой старой вершины. Проблема заключалась в том, чтобы найти путь Эйлера.



Путь Эйлера

Ориентированный граф имеет эйлеров след тогда и только тогда, когда

- не более одной вершины имеет (исходная степень) - (внутренняя степень) = 1,
- не более одной вершины имеет (исходная степень) - (исходная степень) = 1,
- каждая другая вершина имеет одинаковую внутреннюю и исходящую степень.

Путь Эйлера

Алгоритм Флери

Алгоритм начинается с вершины нечетной степени или, если в графе ее нет, начинается с произвольно выбранной вершины.

На каждом шаге он выбирает следующее ребро в пути, удаление которого не разъединит граф, если только такого ребра нет, и в этом случае он выбирает оставшееся ребро, оставшееся в текущей вершине.

Затем он перемещается к другой конечной точке этого ребра и удаляет ребро.

В конце алгоритма не остается никаких ребер, а последовательность, из которой были выбраны ребра, образует эйлеров цикл, если в графе нет вершин нечетной степени, или эйлеров след, если есть ровно две вершины нечетной степени.

Декомпозиция задачи

Структура и технологии

Основные функции программы написаны на **Common Lisp**. В зависимости от того, имеет ли сгенерированный граф де Брёйна путь Эйлера, программа может вернуть либо nil, либо список ребер найденного пути Эйлера. Этот вывод передается в **Python**, который с помощью библиотеки **Graphviz** завершает визуализацию графа Бруджина с последовательной нумерацией ребер.

Основные функции

Выполнить разделение каждого рид из списка ридов разной длинной на k-меры.

```
(defun reads-kmers (reads k)
    (cond ((null reads) nil)
        (t (append (kmers (car reads) k) (reads-kmers (cdr reads) k)))))
```

Основные функции

Найти путь Эйлера.

```
(defun find-path (start graph path)
    (let ((edge (right-edge start graph)))
        (cond ((null graph) path)
            (T (find-path (cadr edge) (remove-edge edge graph 1) (add-edge edge path))))))
```

Основные функции

Выполнить сборку генома после получения пути Эйлера.

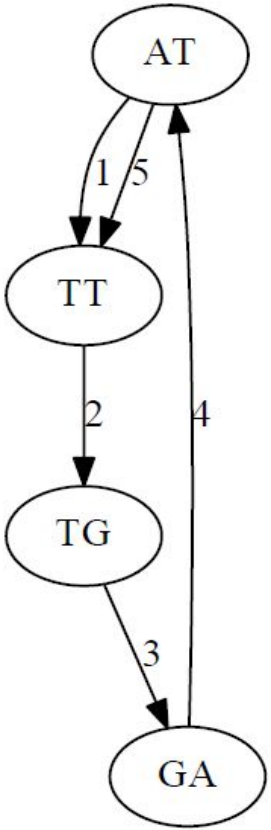
```
(defun genome-assembly (graph genome cnt)
    (cond ((null graph) genome)
        ((= cnt 0) (genome-assembly (cdr graph) (append (caar graph) (last (cadar graph))) (+ cnt 1)))
        (T (genome-assembly (cdr graph) (append genome (last (cadar graph))) (+ cnt 1)))))
```

Примеры работы

- Список ридов: ((T T G) (T G A) (A T T) (G A T) (A T T))
- k = 3
- Путь Эйлера: (((A T) (T T)) ((T T) (T G)) ((T G) (G A))

((G A) (A T)) ((A T) (T T)))

- Полученный геном: (A T T G A T T)

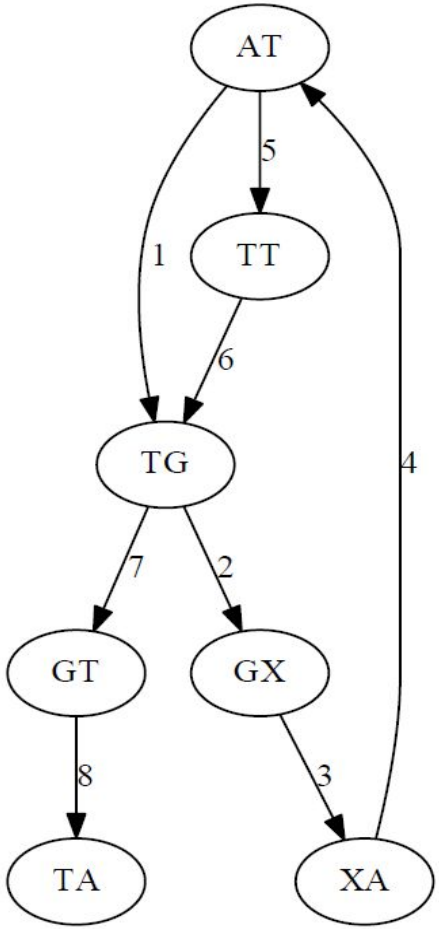


Примеры работы

- Список ридов: ((A T G X) (G X A T) (A T T G) (T G T A))
- k = 3
- Путь Эйлера: (((A T) (T T)) ((T T) (T G))

((T G) (G A)) ((G A) (A T)) ((A T) (T T)))

- Полученный геном: (A T T G A T T)



Анализ полученных результатов

Например, проницательный наблюдатель заметит, что метод де Брейна для сборки фрагментов основан на четырех скрытых предположениях, которые не верны для NGS. Мы считали само собой разумеющимся, что мы можем генерировать все k-меры, присутствующие в геноме, что все k-меры не содержат ошибок, что каждый k-мер появляется в геноме не более одного раза и что геном состоит из одной кольцевой хромосомы.