Extended Context-Free Grammars Parsing with Generalized LL

Author: Artem Gorokhov

Saint Petersburg State University
Programming Languages and Tools Lab, JetBrains

4/March/2017

Motivation

Биоинформатика

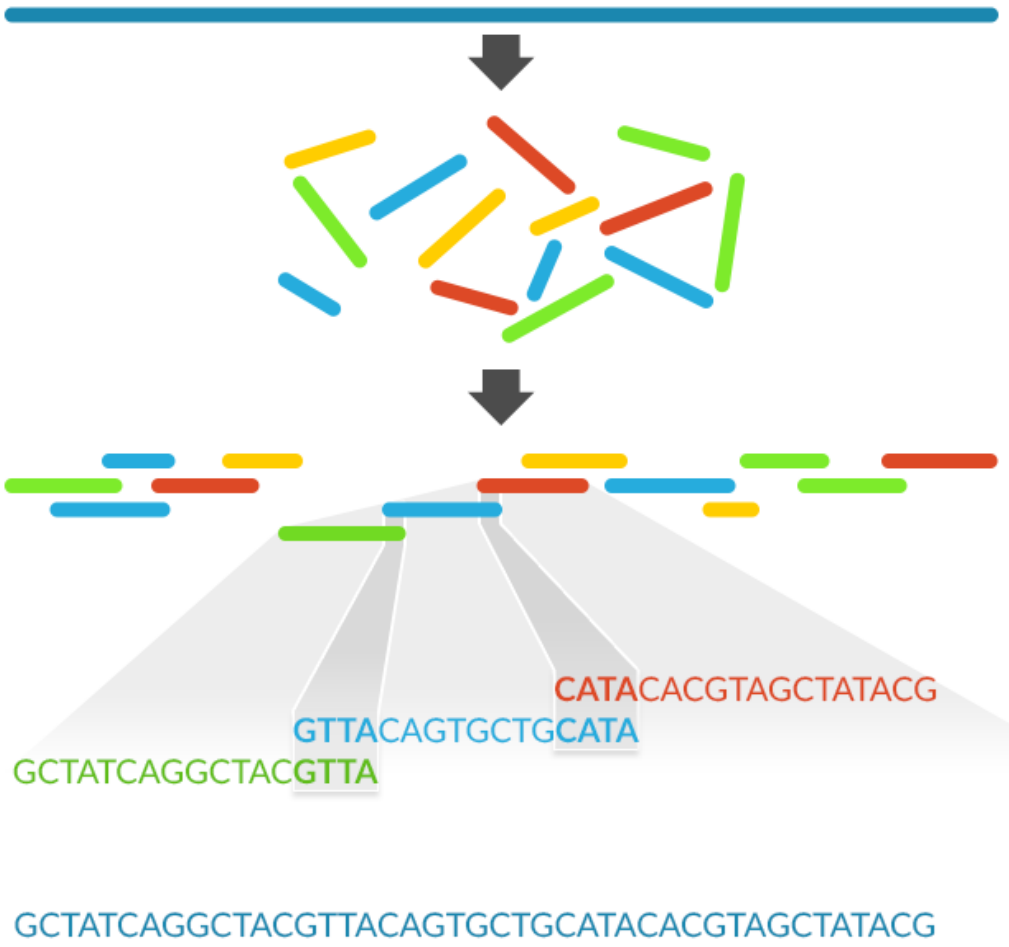
- Множество задач, связанных с обработкой и пониманием биологических данных
- Одна из задач — поиск организмов в метагеномных сборках

Геном

- Геном — длинная последовательность нуклеотидов
- На деле строка над алфавитом {A, C, G, U}

Получение данных

- Из биологического материала читаются короткие строчки
- Эти кусочки склеиваются в более длинные строки
- Множество строчек — сборка
- Данных очень много, поэтому строится граф, сожержащий множество полученных строк

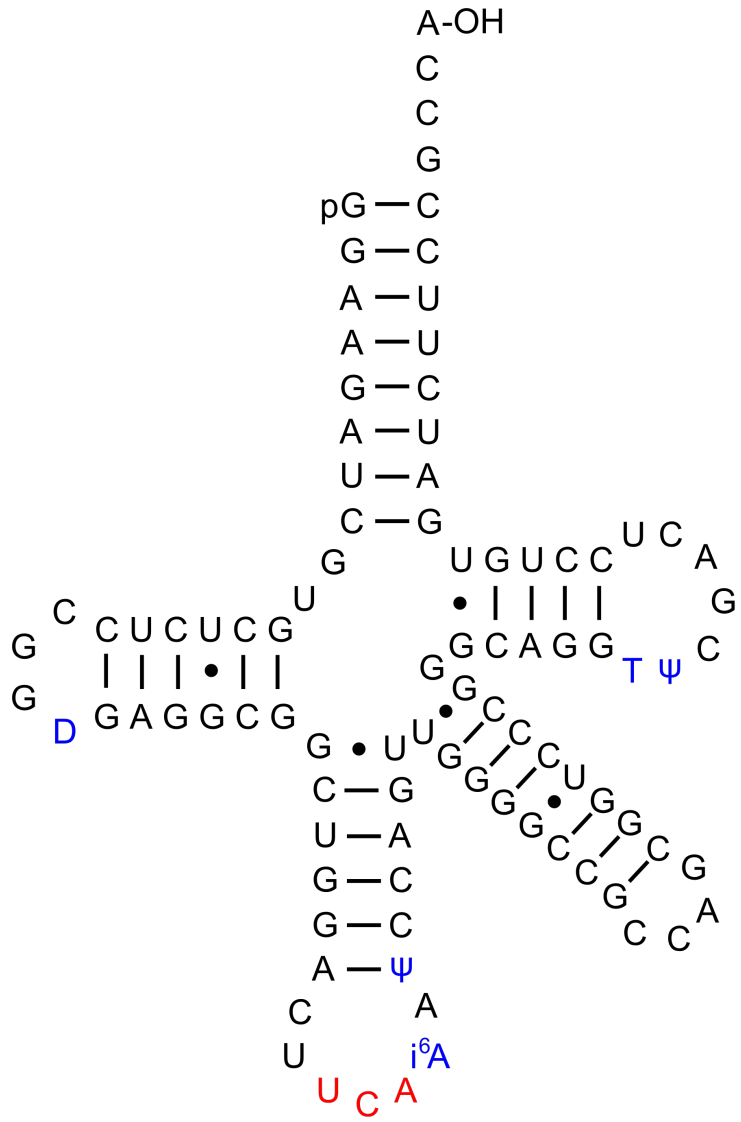


Метагеномная сборка

- Данные из окружающей среды
- Изучаем набор генов всех микроорганизмов в образце

Что ищем

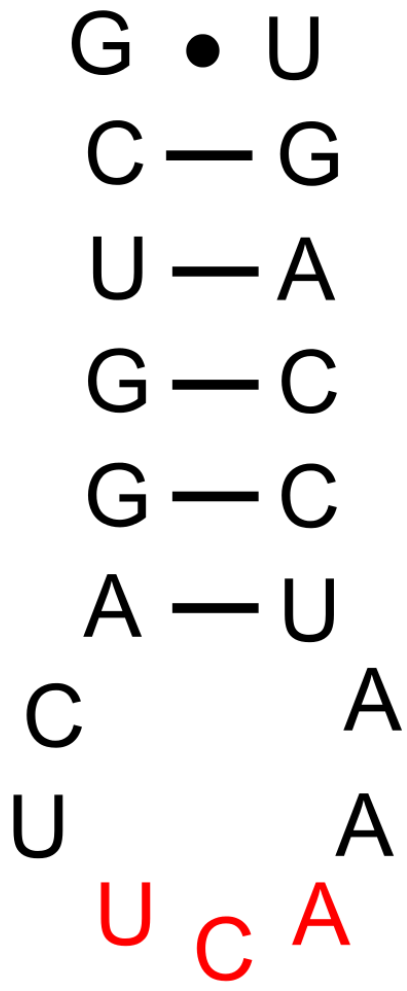
- Хочется понять что у нас в сборке
- Такие последовательности как тРНК, рРНК позволяют провести классификацию организма
- У этих последовательностей есть вторичная структура, которая может быть описана КС-грамматикой



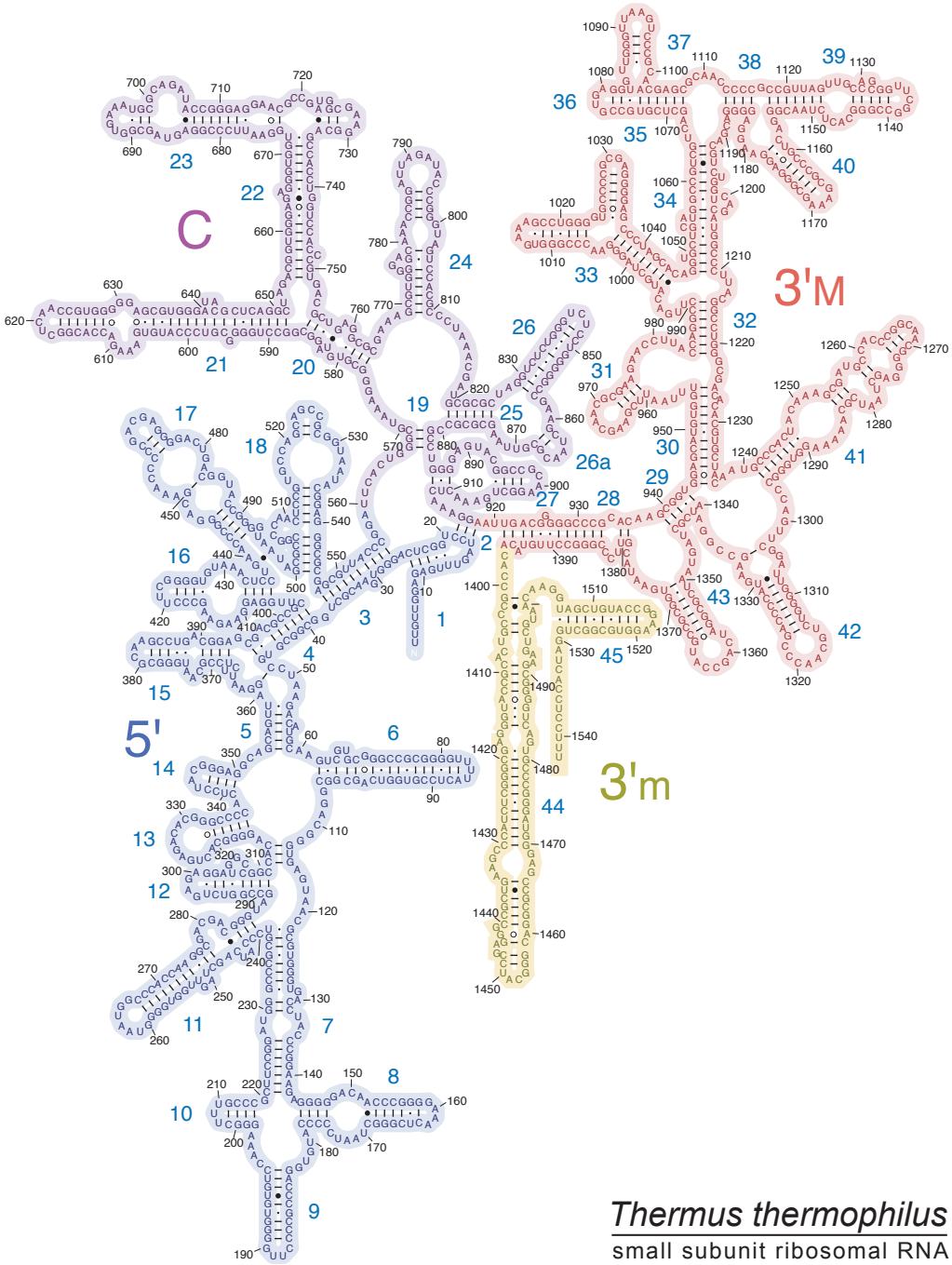
GGAAGAUCG...GCA... =>

Грамматика для кусочка тРНК

$$
\begin{aligned}
START &= STEM \\
STEM &= a\ STEM\ u \\
&\mid u\ STEM\ a \\
&\mid c\ STEM\ g \\
&\mid g\ STEM\ c \\
&\mid g\ STEM\ u \\
&\mid u\ STEM\ g \\
&\mid ANY^{*}[4..7] \\
ANY &= a \mid u \mid g \mid c
\end{aligned}
$$



Вторичная структура
16s pPHK



Graph parsing

- Задача поиска линейных цепочек, удовлетворяющих KC-грамматике, в графе

YaccConstructor

- В лаборатории созданы алгоритмы
- Реализован инструмент, основанный на алгоритме GLL
- Умеет решать задачу поиска линейных цепочек в графе, удовлетворяющих KC-грамматике

Цель и задачи

Цель работы — научиться классифицировать организмы в метагеномной сборке

Задачи:

- Адаптировать существующий алгоритм под специфику задачи
- Провести экспериментальные исследования работы алгоритма

GLL

- Разбор осуществляется при помощи дескрипторов
- Дескриптор — четвёрка (слот, позиция во входе, дерево, вершина стека)
- На каждом шаге достаём дескриптор из очереди и разобрав очередной символ создаём новые дескрипторы

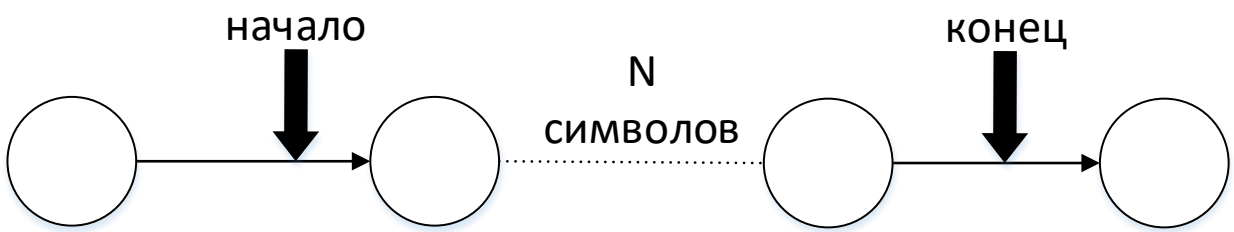
Подготовка сборки

Метагеномные сборки довольно большие, поэтому их необходимо предварительно обрабатывать

- Infernal позволяет распознавать структуры в линейном входе
- Рёбра, длиннее искомых структур можно делить на части и проверять с помощью Infernal
- После фильтрации рёбер граф распадается на компоненты связности, на которых алгоритм можно запускать анализатор независимо

Отказ от построения дерева

- Синтаксический анализатор возвращает лишь границы и длину найденных цепочек
- Восстановление цепочки идёт путём извлечения подграфа, состоящего из путей заданной длины
- Ложные фильтруются с помощью Infernal



Преобразование грамматики

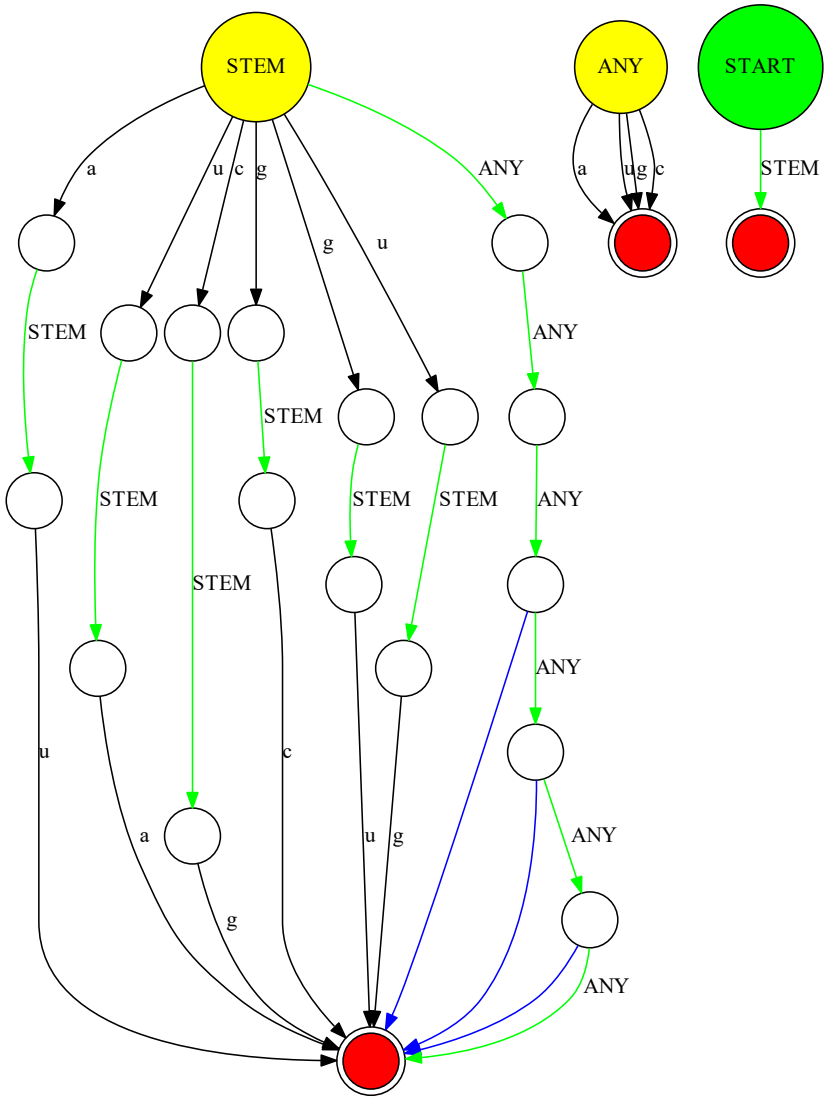
- Грамматика для 16s pPHK сильно неоднозначная и довольно большая
- Из-за этого количество слотов в грамматике очень много
- В процессе разбора создаётся огромное количество дескрипторов

Преобразование грамматики к автомату

Грамматика

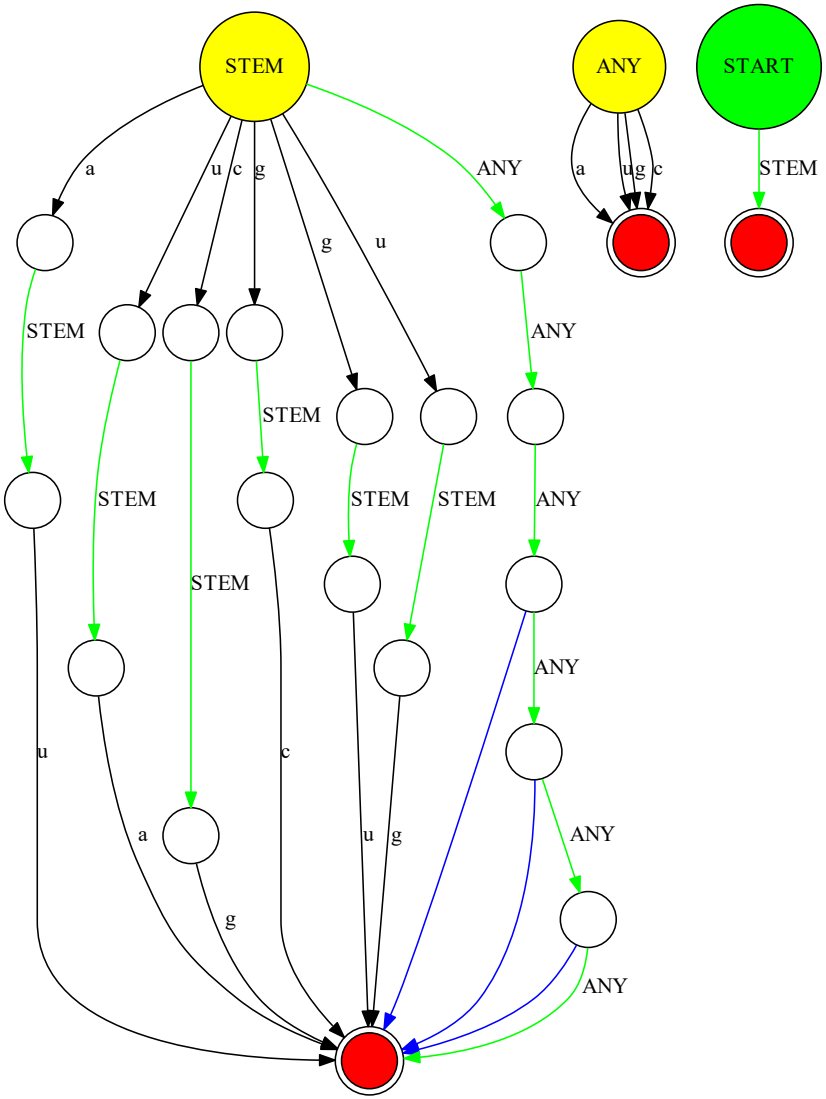
$$
\begin{aligned}
START &= STEM \\
STEM &= a\ STEM\ u \\
&\mid u\ STEM\ a \\
&\mid c\ STEM\ g \\
&\mid g\ STEM\ c \\
&\mid g\ STEM\ u \\
&\mid u\ STEM\ g \\
&\mid ANY^*[3..6] \\
ANY &= a \mid u \mid g \mid c
\end{aligned}
$$

Автомат

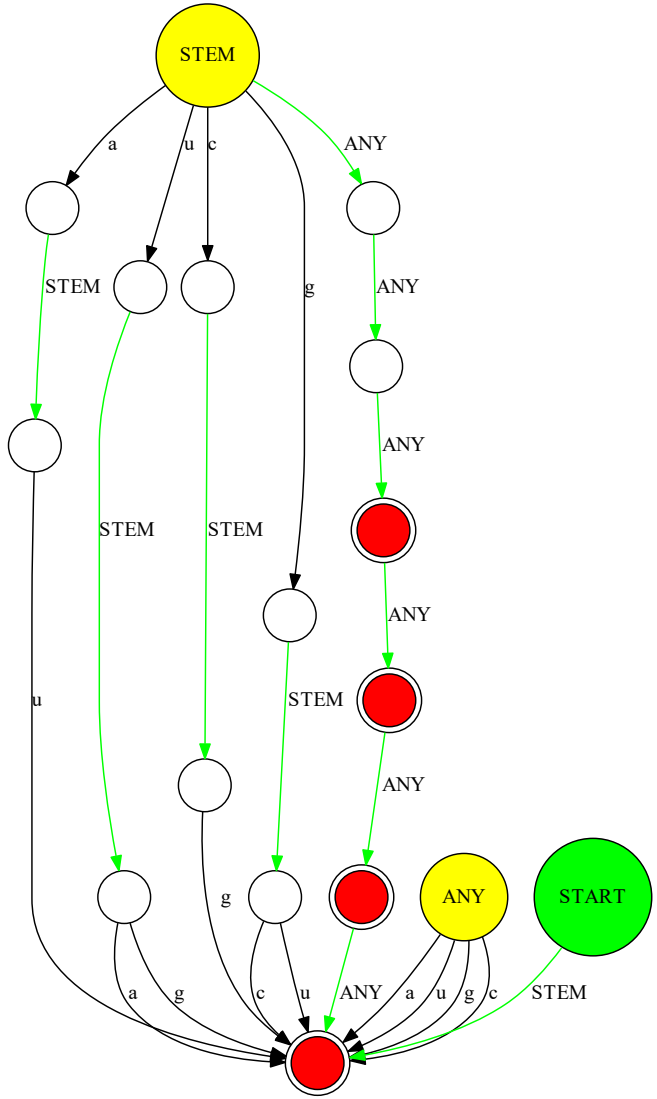


Минимизация автомата

Изначальный автомат



Минимизированый автомат



Эксперименты

Результаты работы на сборке, состоящей из 59000 вершин и 87000 рёбер и грамматике кусочка 16s pPHK, длиной около 300 символов

	начальная грамматика	мин. автомат
Время работы	10 ч.	3 ч. 40 мин.
Кол-во слотов /состояний	41	17

Эксперименты проводились на машине с 32 ГБ RAM и CPU core i7-4790

Результаты

- Разработан механизм подготовки сборок к синтаксическому анализу
- GLL адаптирован под распознавание по грамматике в форме EBNF
- Проведены эксперименты на части грамматики 16s pPHK

Направление работ

- Детальный анализ качества результата
- Исследовать возможность сильнее фильтровать граф, например применяя Infernal
- Поиск полноразмерных 16s pPHK