Архивация FASTQ файлов

Подгузов Никита
Кравченко Дмитрий
Бойкий Дмитрий

Руководитель: Пржибельский Андрей Дмитриевич

Санкт-Петербургский Академический университет

25 мая 2015 года

Цели и задачи

- Задача: максимально сжать входной FASTQ-файл

Цели и задачи

- Задача: максимально сжать входной FASTQ-файл
- Цели: изучить уже известные алгоритмы сжатия, придумать какие-то эвристики к ним

Формат файла

- Биологические данные, набор ридов — маленьких фрагментов длиной от 25 до 20000 символов, кусочков генома

Формат файла

- Биологические данные, набор ридов — маленьких фрагментов длиной от 25 до 20000 символов, кусочков генома
- При считывании возможны ошибки

Формат файла

- Биологические данные, набор ридов — маленьких фрагментов длиной от 25 до 20000 символов, кусочков генома
- При считывании возможны ошибки
- Состоит из блоков данных

Блок данных

```
@EAS54_6_R1_2_1_540_792
TTGGCAGGCCAAGGCCGATGGATCA
+
;;;;;;;;;;7;;;;;-;;;3;83
```

Формат файла

- Биологические данные, набор ридов — маленьких фрагментов длиной от 25 до 20000 символов, кусочков генома
- При считывании возможны ошибки
- Состоит из блоков данных

Блок данных

```
@EAS54_6_R1_2_1_540_792
TTGGCAGGCCAAGGCCGATGGATCA
+
;;;;;;;;;;7;;;;;-;;;3;83
```

- Большой размер входного файла (10 Mb – 100 Gb)

Пути решения

Два типа алгоритмов сжатия:

Пути решения

Два типа алгоритмов сжатия:

1. Словарный метод

Пути решения

Два типа алгоритмов сжатия:

1. Словарный метод
2. Метод энтропийного сжатия

Алгоритм LZW

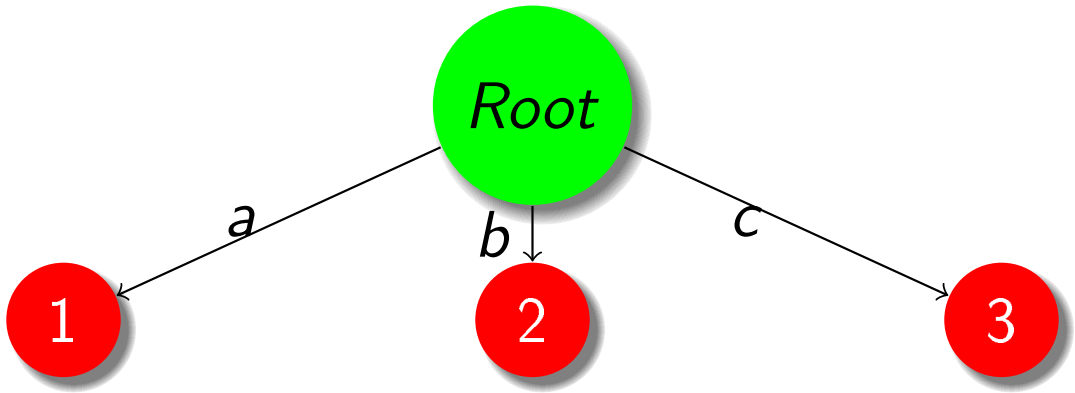
- Тип: словарный

Алгоритм LZW

- Тип: словарный
- Для быстрого поиска ранее встреченных подстрок используется бор

Алгоритм LZW

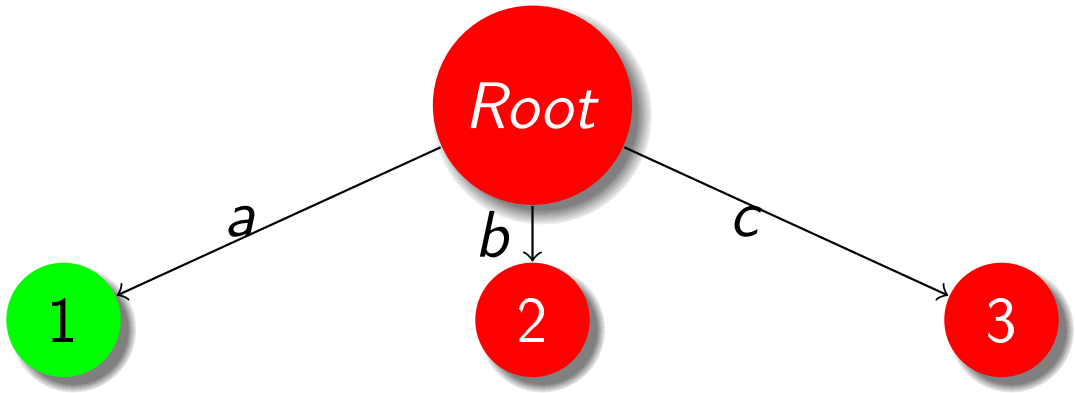
s = abacaba



вывод

Алгоритм LZW

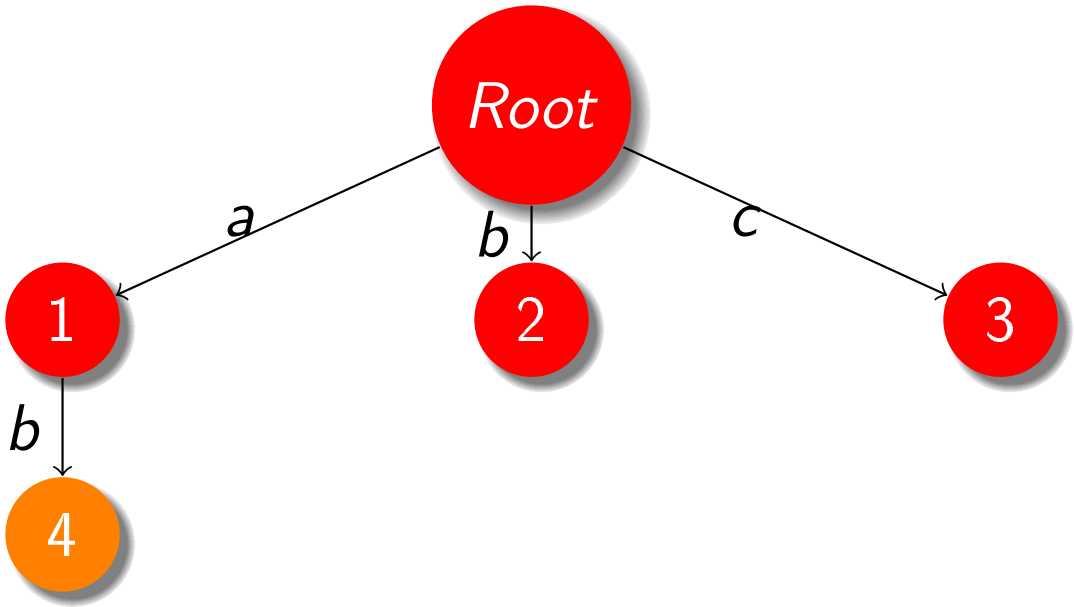
s = abacaba



вывод

Алгоритм LZW

s = abacaba

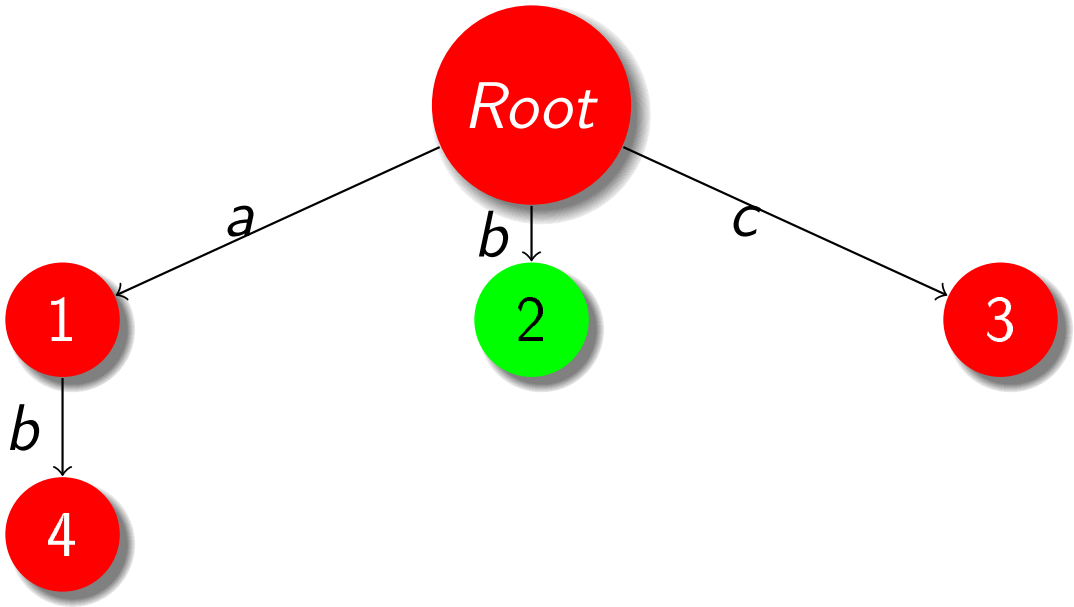


вывод

1

Алгоритм LZW

s = abacaba

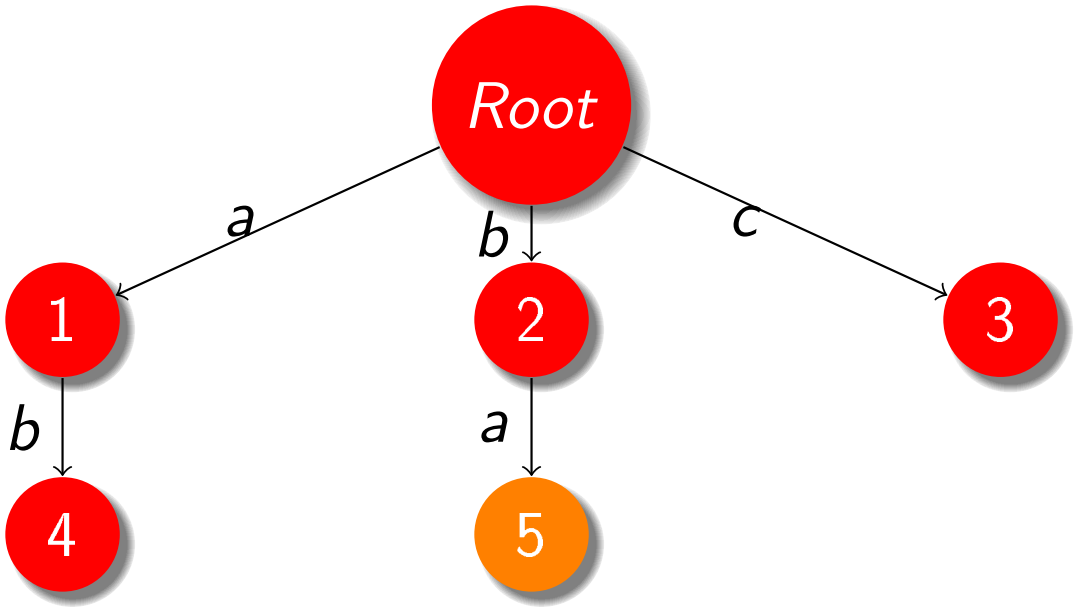


вывод

1

Алгоритм LZW

s = abacaba

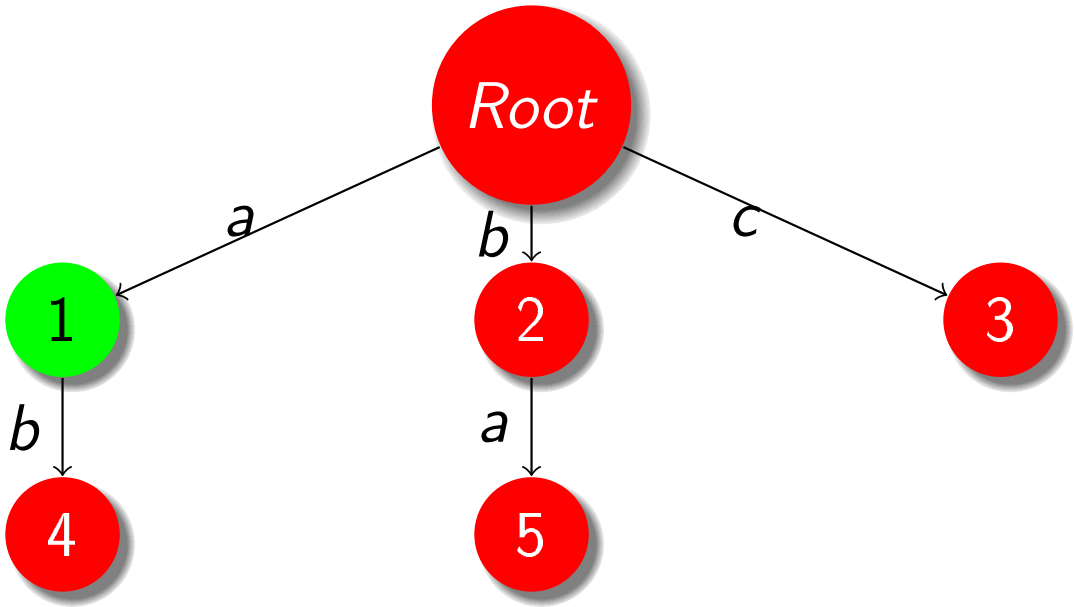


вывод

1 2

Алгоритм LZW

s = abacaba

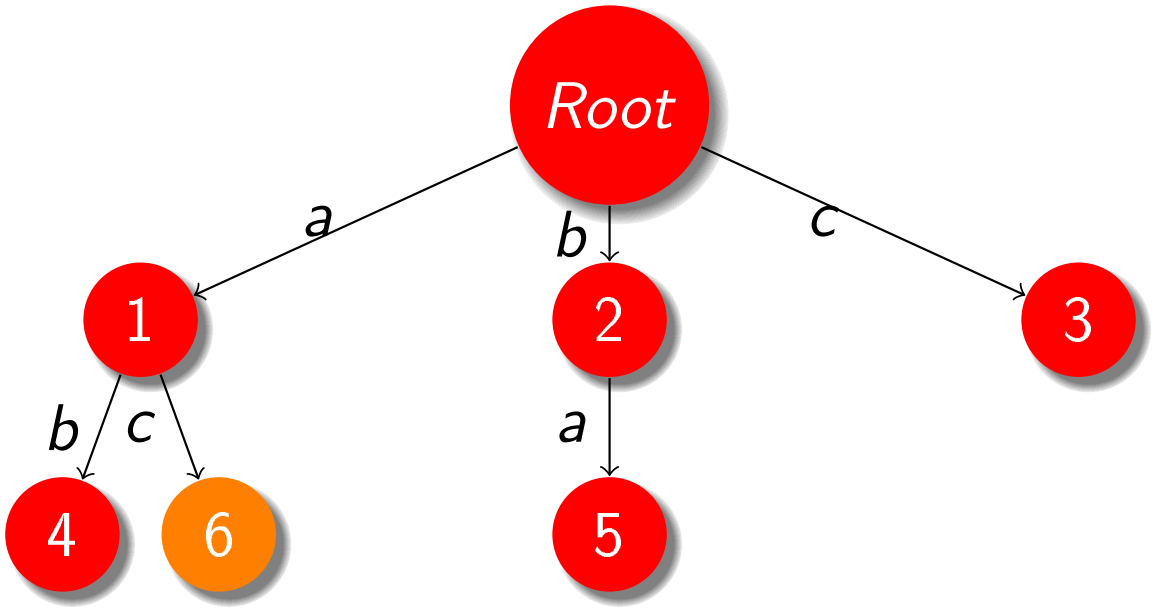


вывод

1 2

Алгоритм LZW

s = abacaba

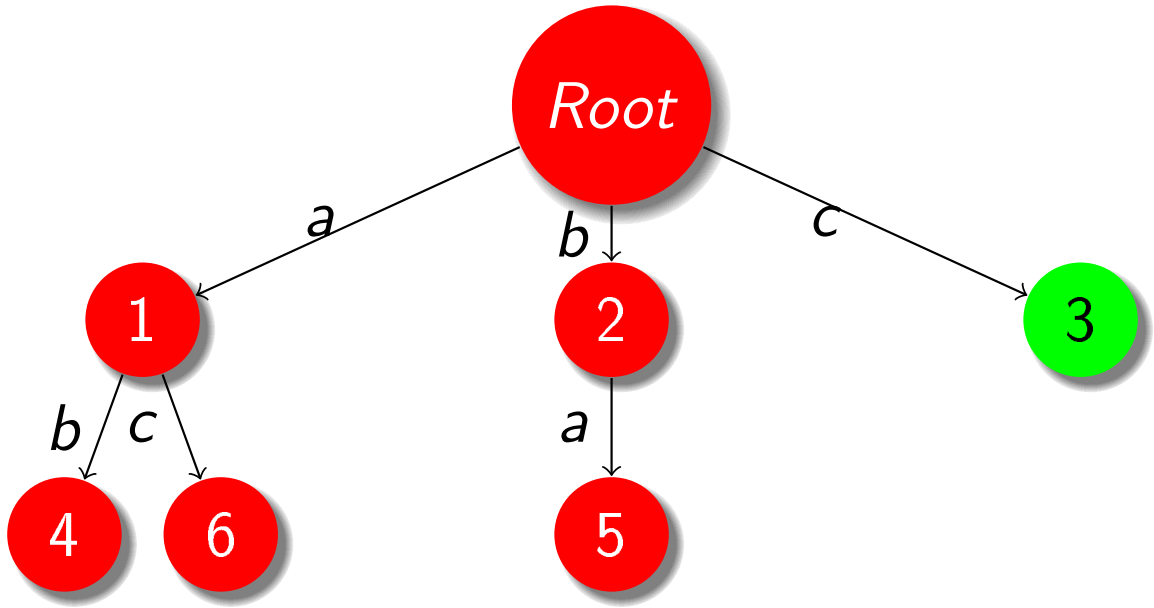


вывод

1 2 1

Алгоритм LZW

s = abacaba

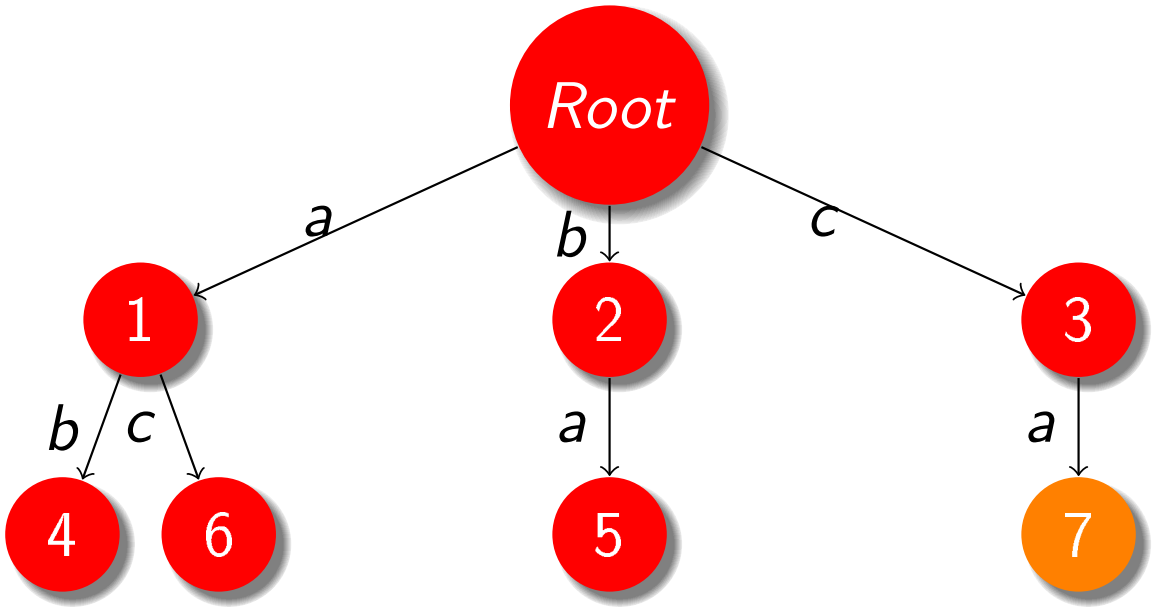


вывод

1 2 1

Алгоритм LZW

s = abacaba

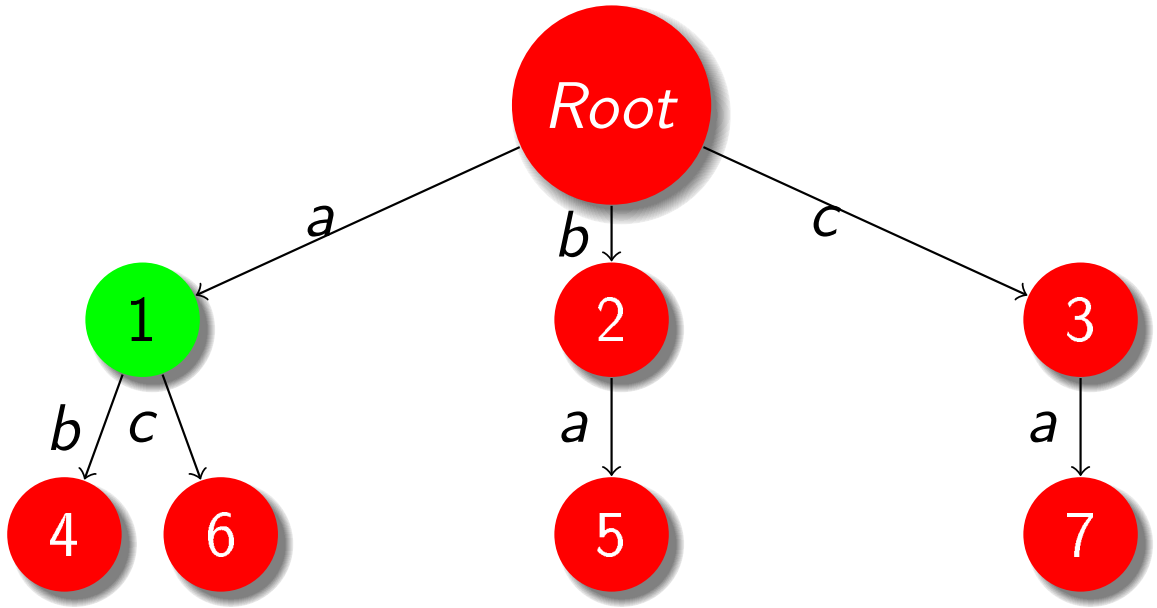


вывод

1 2 1 3

Алгоритм LZW

s = abacaba

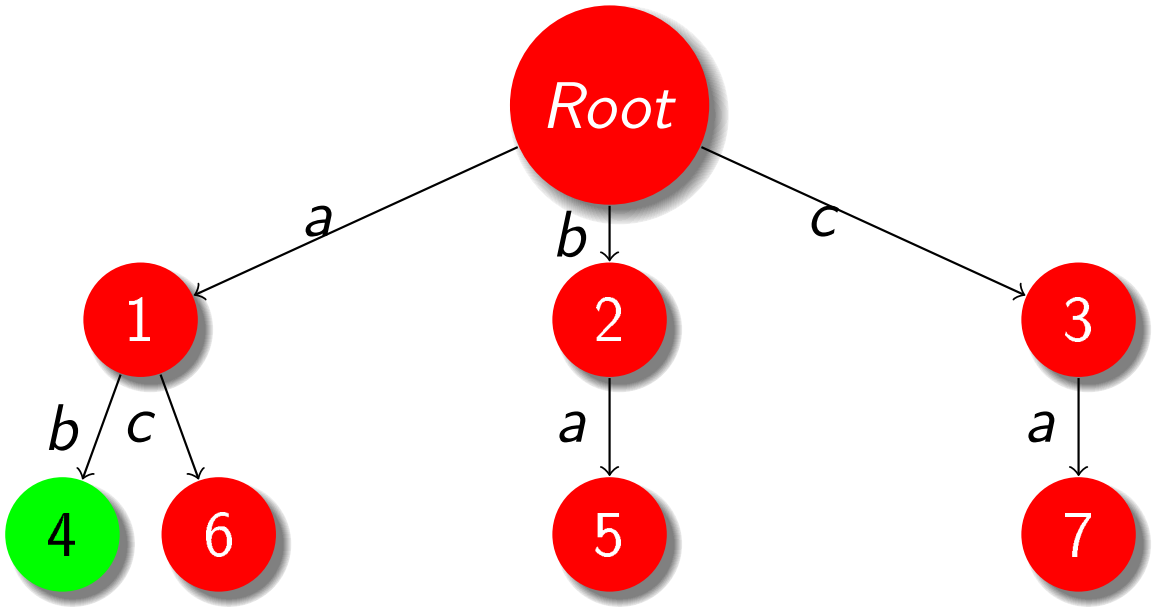


вывод

1 2 1 3

Алгоритм LZW

s = abacaba

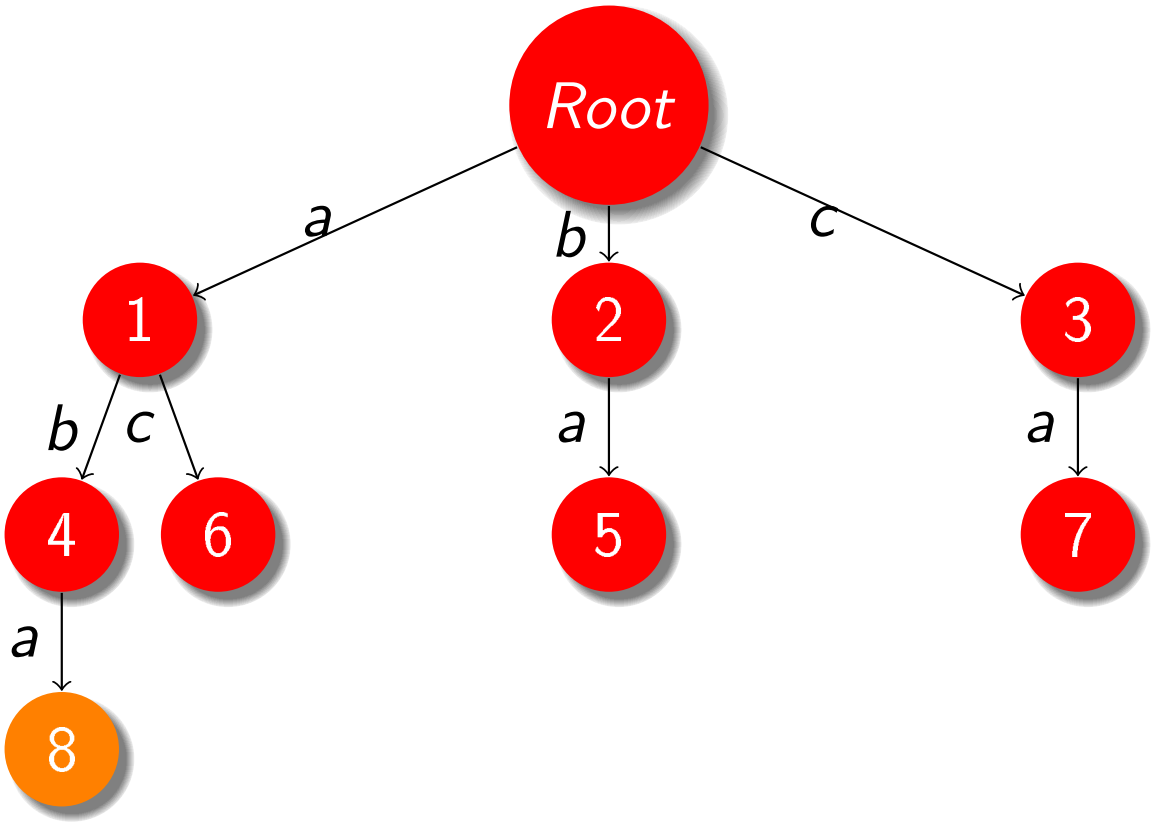


вывод

1 2 1 3

Алгоритм LZW

s = abacaba

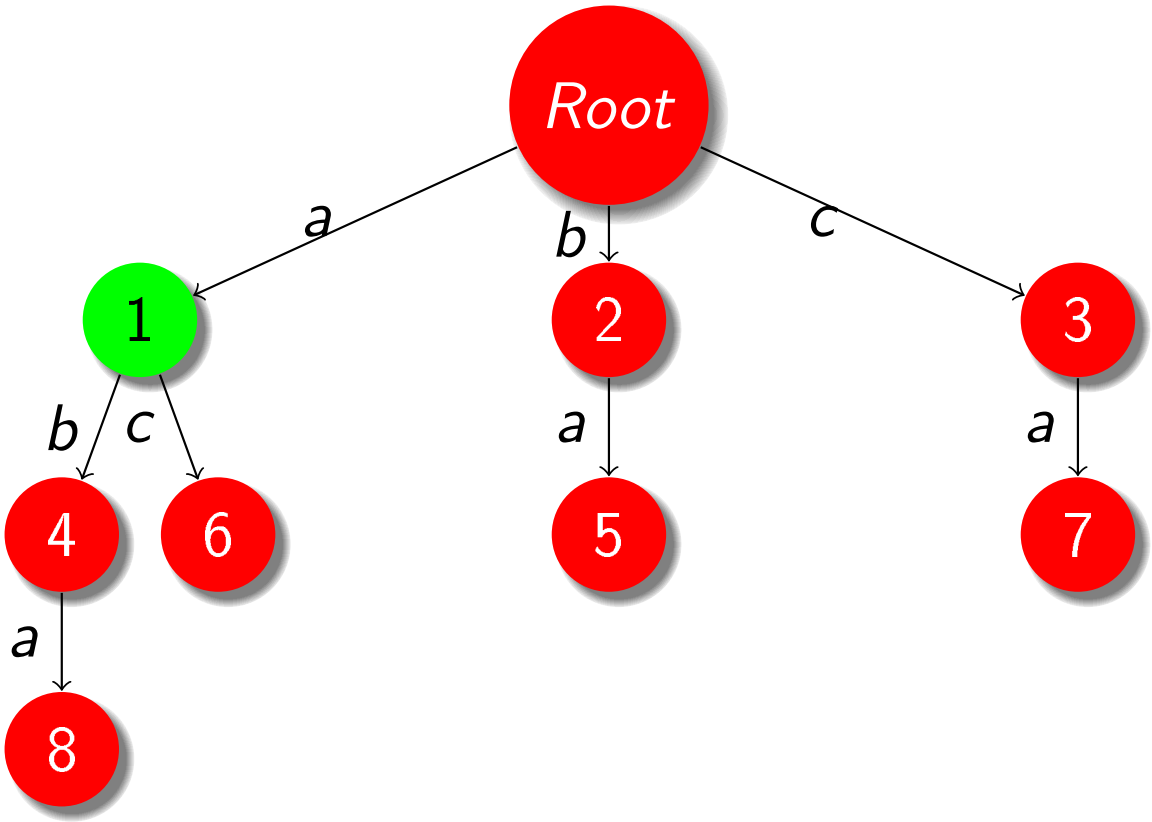


вывод

1 2 1 3 4

Алгоритм LZW

s = abacaba

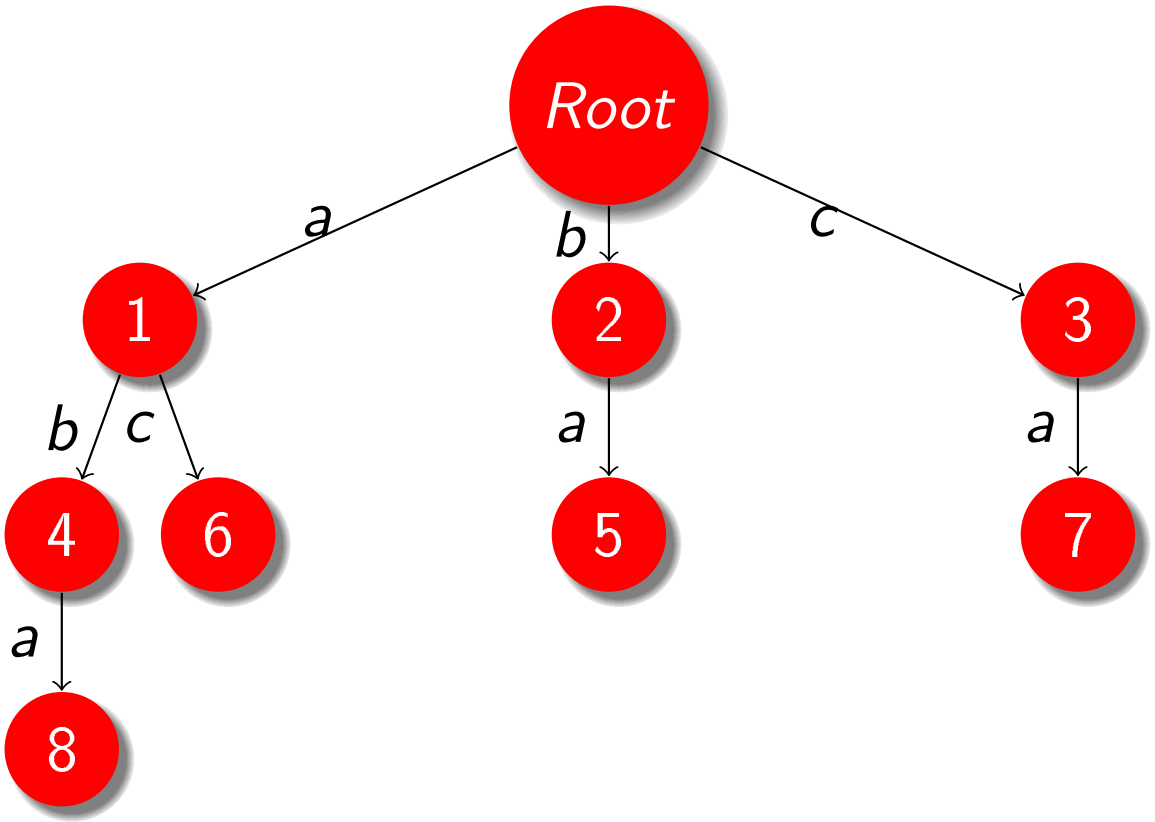


вывод

1 2 1 3 4

Алгоритм LZW

s = abacaba



вывод

1 2 1 3 4 1

Алгоритм LZW

Проблема: из-за большого объема данных надо чистить бор
Способы:

Алгоритм LZW

Проблема: из-за большого объема данных надо чистить бор
Способы:

1. Оставить бор неизменяемым, как только заполнится первый раз

Алгоритм LZW

Проблема: из-за большого объема данных надо чистить бор
Способы:

1. Оставить бор неизменяемым, как только заполнится первый раз
2. Полное очищение бора при его заполнении

Алгоритм LZW

Проблема: из-за большого объема данных надо чистить бор
Способы:

1. Оставить бор неизменяемым, как только заполнится первый раз
2. Полное очищение бора при его заполнении
3. Множество, в котором хранятся встречаемости вершин.

Алгоритм LZW

Проблема: из-за большого объема данных надо чистить бор
Способы:

1. Оставить бор неизменяемым, как только заполнится первый раз
2. Полное очищение бора при его заполнении
3. Множество, в котором хранятся встречаемости вершин.
4. Очищение какой-то части бора (например, 50%) при его заполнении

Алгоритм LZW

Проблема: из-за большого объема данных надо чистить бор
Способы:

1. Оставить бор неизменяемым, как только заполнится первый раз
2. Полное очищение бора при его заполнении
3. Множество, в котором хранятся встречаемости вершин.
4. Очищение какой-то части бора (например, 50%) при его заполнении
5. Каждое константное количество шагов очищать бор на константное количество вершин

Алгоритм Хаффмана

- Тип: частотный

Алгоритм Хаффмана

- Тип: частотный
- Алгоритм

Алгоритм Хаффмана

- Тип: частотный
- Алгоритм
  - Подсчёт встречаемости каждого символа

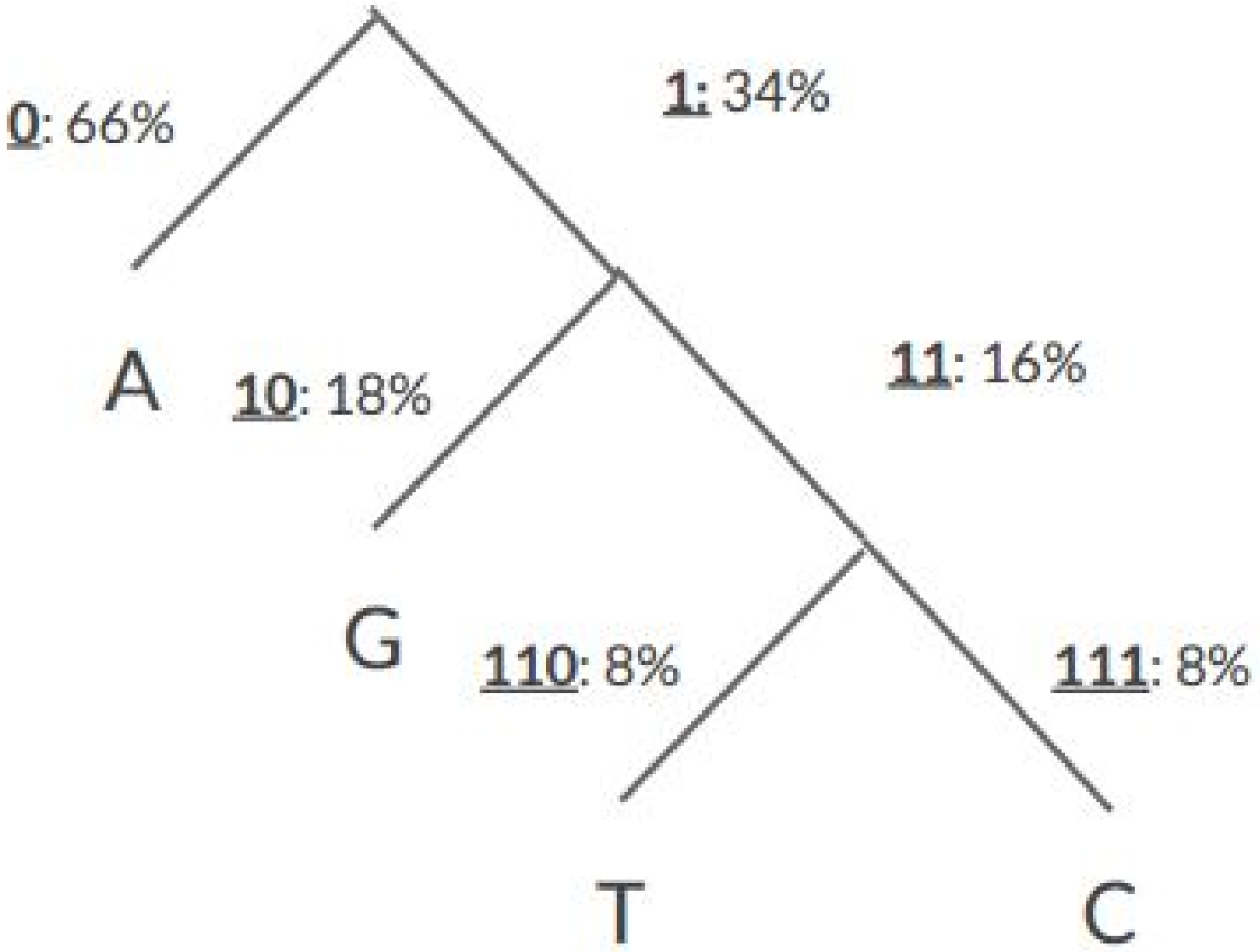
Алгоритм Хаффмана

- Тип: частотный
- Алгоритм
  - Подсчёт встречаемости каждого символа
  - Построение дерева

Алгоритм Хаффмана

- Тип: частотный
- Алгоритм
  - Подсчёт встречаемости каждого символа
  - Построение дерева
  - Сопоставление символов и кодов

Алгоритм Хаффмана



Оптимизации

- Разделение файла на 4 части

Оптимизации

- Разделение файла на 4 части
- Применение разных алгоритмов к разным частям

Оптимизации

- Разделение файла на 4 части
- Применение разных алгоритмов к разным частям
- Удаление ненужных веток в боре (алгоритм LZW)

Оптимизации

- Разделение файла на 4 части
- Применение разных алгоритмов к разным частям
- Удаление ненужных веток в боре (алгоритм LZW)
- Кодирование разностей между Quality

Технические подробности

- Проект написан на C++ (Google C++ Style Guide)

Технические подробности

- Проект написан на C++ (Google C++ Style Guide)
- Для удобства работы в команде использовался git

Технические подробности

- Проект написан на C++ (Google C++ Style Guide)
- Для удобства работы в команде использовался git
- Тестирование на реальных данных (см. результаты ниже)

Технические подробности

- Проект написан на C++ (Google C++ Style Guide)
- Для удобства работы в команде использовался git
- Тестирование на реальных данных (см. результаты ниже)
- Ссылка на исходники: https://github.com/SPbAU-BI1/FASTQ

Результаты

Archiver	input file size (MB)	output file size (MB)	compression rate
our	63.50	20.65	0.31
gz	63.50	24.29	0.38
zip	63.50	24.29	0.38
rar	63.50	21.53	0.34
bzip2	63.50	19.10	0.30

Вопросы?