

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

09.03.04 - Программная инженерия
Профиль направления подготовки бакалавриата
«Системное и прикладное программное обеспечение»

ОТЧЁТ О ПРОХОЖДЕНИИ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ

Выполнил:

студент 2 курса группы 22207

Кириллов Иван Сергеевич

Место прохождения практики:

Кафедра информатики и математического
обеспечения

Сроки прохождения практики:

30.05-09.06

Руководитель практики:

к.т.н., доцент

Богоявленская Ольга Юрьевна

Оценка _____

Дата _____

Содержание

Введение	3
1 Структура (statistic_counter)	3
1.1 Префиксное дерево (бор)	3
1.2 Сжатие бора	4
1.3 Реализация структуры	5
2 Тестирование	7
3 Заключение	8
Список литературы	9

Введение

В наши дни задача генерации текста по математическим моделям становится все более и более актуальной в связи с тем, что она находит все больше применений в повседневной жизни. Данная задача изящно решается с помощью алгоритмов на основе Марковских цепей [1].

Цель: реализовать алгоритм построения статистики суффиксов и префиксов по заданным текстам.

Задачи:

1. Ознакомление с литературой по генерации текстов с помощью Марковских цепей;
2. Создание структуры данных для хранения префиксов/суффиксов текста на C++ и её последующая интеграция в Python;
3. Создание программного модуля по подсчёту и анализу префиксов и суффиксов в тексте;
4. Тестирование разработанного модуля и структуры данных.

Кооперация с другими разработчиками данной задачи (Павловым Максимом и Афанасьевым Артёмом), а также контроль версий программного кода и распределение подзадач осуществлялась с помощью репозитория на GitHub. В нашей группе моя задача заключалась в том, чтобы написать структуру для удобного хранения статистики префиксов/-суффиксов.

1 Структура (statistic_counter)

В основе структуры лежит бор (префиксное дерево) со сжатием построенное на указателях. Для начала обсудим его, а затем перейдем к реализации.

1.1 Префиксное дерево (бор)

Бор (англ. trie, луч, нагруженное дерево) — структура данных для хранения набора строк, представляющая из себя подвешенное дерево с символами на рёбрах. Строки получаются последовательной записью всех символов, хранящихся на рёбрах между корнем

бора и терминальной вершиной. Размер бора линейно зависит от суммы длин всех строк, а поиск в бору занимает время, пропорциональное длине образца.

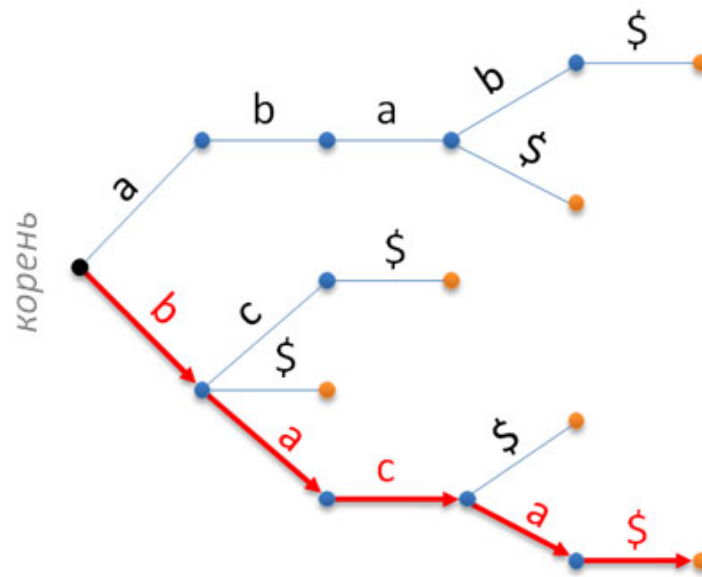


Рис. 1: бор

1.2 Сжатие бора

Для сокращения размера используемой памяти дерево хранится в частично сжатой форме, когда все участки без ветвлений, упаковываются в строки.

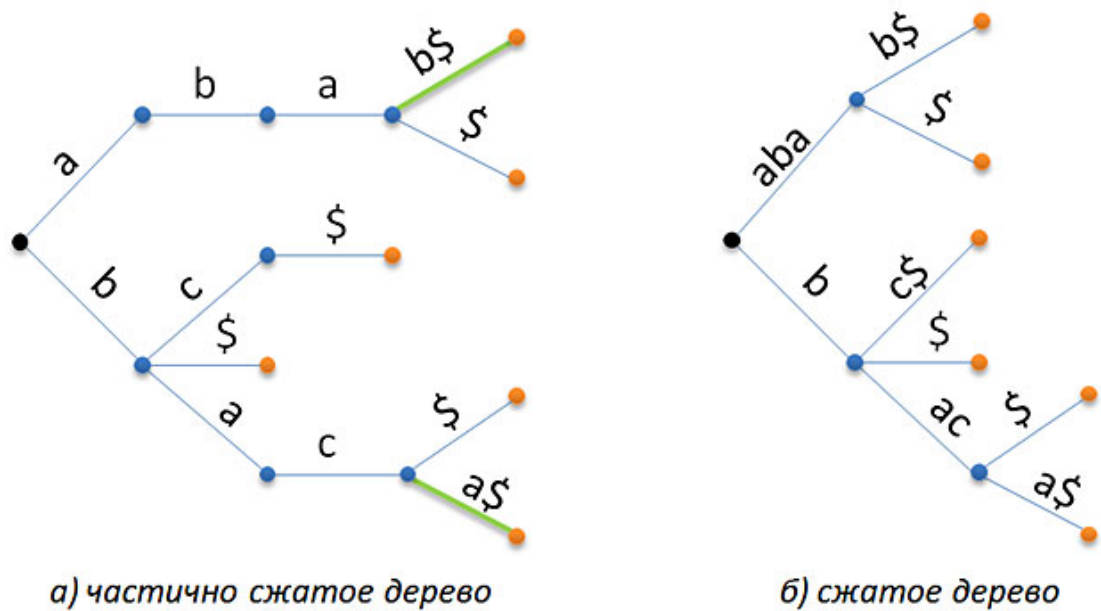


Рис. 2: Сжатие бора

1.3 Реализация структуры

Алгоритм был реализован на языке программирования C++. Это было сделано по двум причинам, во-первых, в C++ удобно работать с указателями, а во-вторых C++ достаточно просто интегрировать с python, на котором работали мои коллеги. Сам бор был построен на указателях.

Для хранения вершин бора бала написан класс `node` (листинг 2). Поле `prev` хранит указатель на предыдущую вершину, массив `next` — на следующую (37 символов: 26 — букв, пробел и 10 цифр), `count` — количество префиксов заканчивающихся в этой вершине, `pos` — позиция префикса в списке, отсортированном по убыванию количества префиксов, `part` — часть строки хранящаяся в этой вершине.

Листинг 1: Класс `node`

```
1 class node{
2 public:
3     node* prev;
4     node* next[37];
5     int count;
6     int pos;
7     std::string part;
8     node(){
9         count = 0;
10        pos = 0;
11        prev = nullptr;
12        for (int i = 0; i < 37; i++)
13            next[i] = nullptr;
14    }
15};
```

Для хранения самого дерева используется класс `statistic_counter` (рис. 4, листинг 2). Массив `statistic` хранит ссылки на вершины, отсортированные по убыванию количества префиксов/суффиксов, заканчивающихся в этой вершине, `root` — ссылка на корень дерева, `size` — количество различных префиксов, массив `count` под индексом `i` хранит левую и правую границу подмассива префиксов, которые встречаются `i` раз.

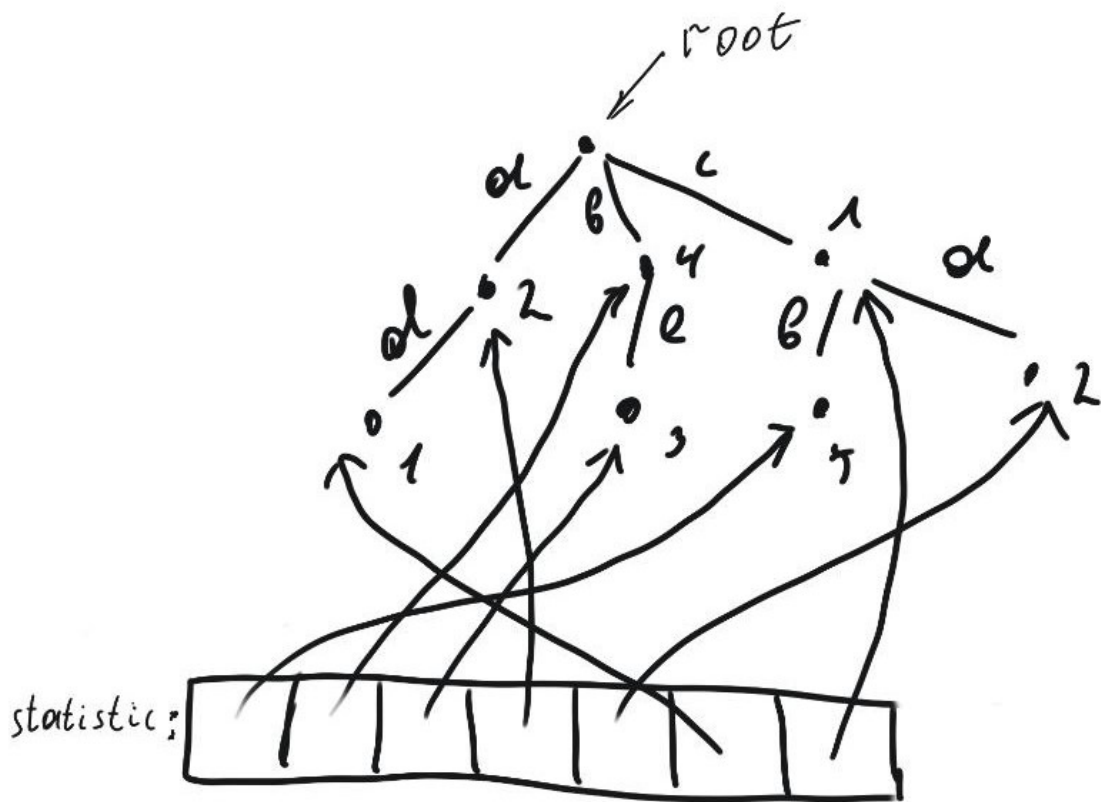


Рис. 3: Схема устройства структуры statistic_counter

Листинг 2: Класс statistic_counter

```

1 class statistic_counter{
2 private:
3     node** statistic = new node*[2000000];
4     node* root;
5     std::pair<int, int>* count = new std::pair<int, int>[2000000];
6     int size;
7     int pointer;
8     // разбивает вершину на две
9     node* split(node* cur, int& p);
10
11    // обновляет массив statistic
12    void update_statistic(node* cur);
13
14    // создает новую вершину и добавляет в дерево

```

```

15     node* create_node(std::string part);
16 public:
17     // Конструктор
18     statistic_counter();
19
20     // Добавление префикса/ суффикса
21     void add(std::string pref);
22
23     // Получение числа префиксов/ суффикса pref
24     int get_by_pref(std::string pref);
25
26     // Получение kго— по встречаемости префикса/ суффикса
27     std::string get_by_number(int k);
28
29     // Получение следующего по встречаемости префикса/ суффикса количество
        возвращается через пробел
30     // Можно while get_next(): использовать чтобы все слова получить.
31     std::string get_next();
32
33     // Изменение текущего по встречаемости префикса/ суффикса возвращает текущий
        номер префикса/ суффикса
34     int set_pointer(int new_value);
35
36     // деструктор, необходим для очистки динамической памяти при удалении структуры
37     ~statistic_counter();
38 };

```

2 Тестирование

Все методы были протестированны на различных входных данных, содержащих как просто латинские буквы, так и пробелы и цифры.

```
===== test session starts =====
collecting ... collected 3 items

test.py::test_statistic_class PASSED [ 33%]
test.py::test_prefix_statistic PASSED [ 66%]
test.py::test_suffix_statistic PASSED [100%]

===== 3 passed in 0.64s =====

Process finished with exit code 0
```

Рис. 4: результаты тестов

3 Заключение

В результате проделанной работы удалось выполнить все задачи и достичь поставленной цели, то есть реализовать алгоритм построения статистики суффиксов и префиксов по заданным текстам.

В процессе работы были получены навыки работы в команде с другими разработчиками, изучена такая структура данных как бор и алгоритм сжатия бора, отработано умение работать с динамической памятью и указателями на языке программирования C++.

Список литературы

1. Керниган, Брайан У., Пайк, Роб. Практика программирования. : Пер. с англ. - М. : ООО "И.Д. Вильямс -288 с.
2. URL: <https://habr.com/ru/articles/151421/>
3. URL: <https://neerc.ifmo.ru/wiki/index.php?title=Бор>