### Московский авиационный институт

**Факультет прикладной математики и физики**

**Лабораторная работа №3**

**по курсу:**

**«Информационный поиск»**

**по теме:**

**«Булев индекс»**

**2 семестр**

Студент: Ахмед С. Х.

Преподаватель: Калинин А. Л.

Группа: 8О-106М

**Москва, 2019 г**

Постановка задачи

Требуется построить поисковый индекс, пригодный для булева поиска, по подготовленному в ЛР1 корпусу документов. Требования к индексу:

• Самостоятельно разработанный, бинарный формат представления данных. Формат необходимо описать в отчёте, в побайтовом представлении.

• Формат должен предполагать расширение, т.к. в следующих работах он будет меняться под требования новых лабораторных работ.

• Использование текстового представления или готовых баз данных не допускается.

• Кроме обратного индекса, должен быть создан «прямой» индекс, содержащий в себе как минимум заголовки документов и ссылки на них (понадобятся для выполнения ЛР4, при генерации страницы поисковой выдачи).

• Для термов должна быть как минимум понижена капитализация. В отчёте должно быть отмечено как минимум:

• Выбранное внутренне представление документов после токенизации.

• Выбранный метод сортировки, его достоинства и недостатки для задачи индексации.

Оборудование:

Компьютер HP Omen 15 под управлением операционной системы Windows 10, Intel Core i5-7300HQ 2.50 GHz, 12 Gb RAM

Программное обеспечение

|  |  |
| --- | --- |
| Язык программирования | Python 3.6 |
| Среда программирования | Anaconda, Jupyter Notebook |

Структура

В данной лабораторной работе я решил рассмотреть два формата для булева индекса:

1. Использования дерева для хранения токенов и использования смещения для координатных блоков (два файла: словарь + файл координатных блоков со смещениями)
2. Использования одного дерева где имеется ключ, который является хэшом строки и координатные блоки. <36 байт хэш строки><координатный блок = 120 байт>

На практике второй вариант не распространён, обычно словарь влезает в память, а координатные блоки хранятся отдельно, так как могут не влезть в память. Обычно используется первый подход, только вместе таблицы используется хэш и сортируется хэш

Стоит заметить, что время выполнения двух этих подходов примерно равны:

Для первого:



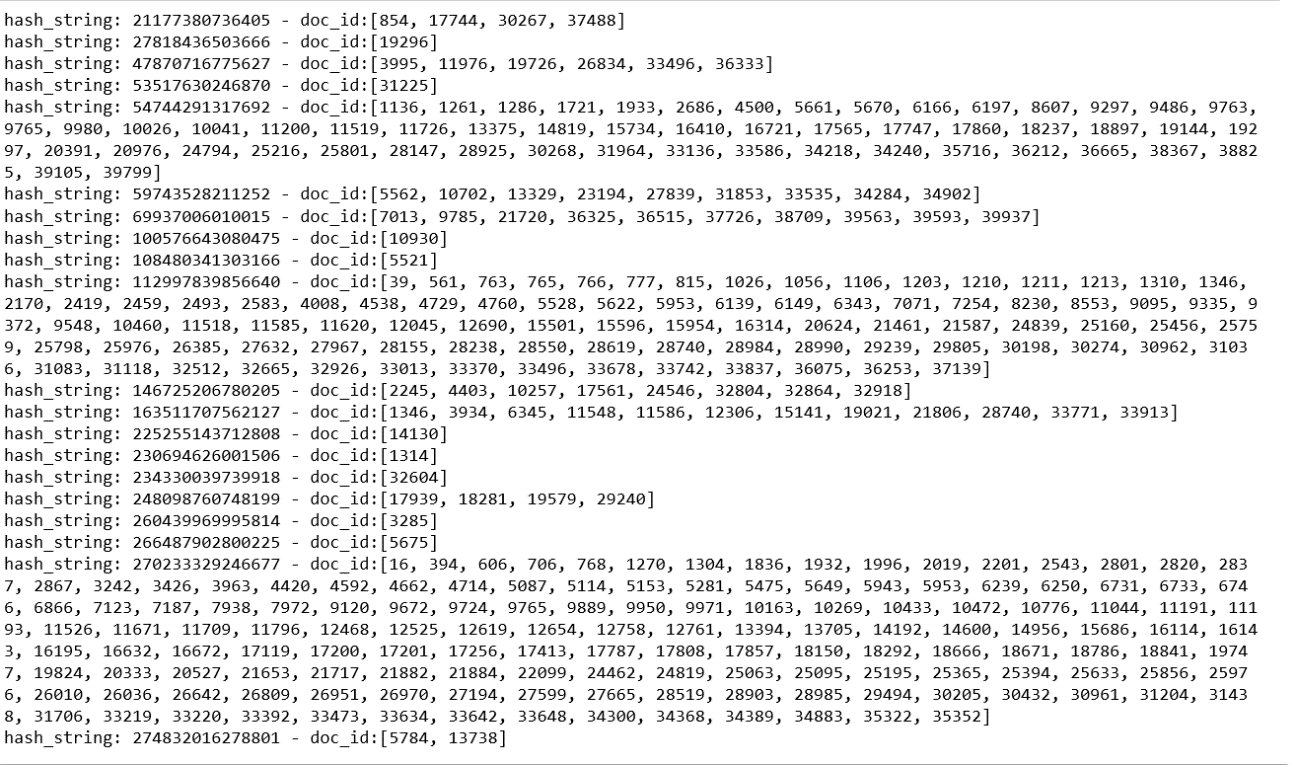
Для второго:



Прогон осуществлялся при размере выборки 40.000. В качестве хэш фукнции применялся sha256.

Также на изображении отображено сколько времени требуется для работы с одним файлом.

Стоит заметить, что в моем случае алгоритм создания булева индекса зависит от скорости вставки в дерево, от скорости считывания записи с диска/ оперативки и для записи в файл: скорость обхода дерева

Как выглядят токены: 

Тестировалось все следующим образом: Взял выборку из статей(три статьи) и проверил вхождение терминов из этих статей в координатный блок, если какого-то термина не было в той статье в которой была, то значит вся проверка выпадает по эксепшену.  Из возможных ошибок, коллизия, местами я не позаботился о обработке тестирующего текста