**Московский авиационный институт**

**(национальный исследовательский университет)**

**Институт информационные технологии и прикладной математики**

**Кафедра вычислительной математики и программирования**

**Лабораторная работа №3 по курсу**

**«Информационный поиск»**

|  |  |
| --- | --- |
| Студент: | Е.М. Стифеев |
| Преподаватель: | А.А. Кухтичев |
| Группа: | М8О-109М-21 |
| Дата: | 28.10.21 |
| Оценка: |  |
| Подпись: |  |

**Москва, 2021**

# Лабораторная работа №3 «Булев индекс»

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 (или побитовом) 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.

Среди результатов и выводов работы нужно указать:

* Количество термов.
* Средняя длина терма. Сравнить со средней длинной токена, вычисленной в ЛР1 по курсу ОТЕЯ. Объяснить причину отличий.

Скорость индексации: общую, в расчёте на один документ, на килобайт текста.

* Оптимальна ли работа индексации? Что можно ускорить? Каким образом? Чем она ограниченна? Что произойдёт, если объём входных данных увеличится в 10 раз, в 100 раз, в 1000 раз?

# Описание

## Корпус

Напомню, что корпус документов имеет следующую структуру, полученную по результатам ЛР1 (доступен по ссылке <https://cloud.mail.ru/public/ZfkX/gccM7hnDR> ):

* Корпус документов
  + films1.txt (94 Мб, 15000 документов)
  + films2.txt (96 Мб, 15000 документов)
  + films3.txt (184 Мб, 15000 документов)
  + films4.txt (219 Мб, 15000 документов)
  + films5.txt (322 Мб, 15000 документов)
  + films6.txt (711 Мб, 15000 документов)
  + films7.txt (823 Мб, 15000 документов)
  + films8.txt (226 Мб, 15000 документов)
  + films9.txt (67 Мб, 15000 документов)
  + films10.txt (75 Мб, 15000 документов)
  + films11.txt (99 Мб, 15000 документов)
  + films12.txt (78 Мб, 15000 документов)
  + films13.txt (41 Мб, 6109 документов)

Также, напомню, что получение одного документа могло включать проход по нескольким html-страницам и обработку динамически подгружаемых страниц, поэтому общее количество обработанных страниц было >800’000.

В каждом файле \*.txt документы хранятся следующим образом:

* 1 строка 1 документ {….}
* 2 строка 2 документ {….}
* *n* строка *n* документ {….}

Каждый документ снабжён прямой ссылкой на источник, откуда был скачен, и хранит только выделенный из html-кода текст в кодировке UTF-8. Например, 234 строка файла films1.txt выглядит так:

{"page\_url": "https://www.kinopoisk.ru/media/article/1773537/", "title": "Артур Смольянинов: «Я сомневался, что смогу сыграть ангела»", "body": "2 января в российский прокат вышла романтическая комедия Веры Сторожевой „Мой парень — ангел“, главные роли в которой исполнили Артур Смольянинов и Анна Старшенбаум. Мы подготовили небольшой видеосюжет с участием создателей картины...Студентка Саша с большим трудом верит в чудеса. Ангелу Серафиму приходится приложить немало усилий, чтобы доказать ей, что ангелы существуют. Но он не учел одного: если девушка тебе поверит, она, скорее всего, тебя полюбит.\n\n\n\n\n\n\n\nАвтор: Дарико Цулая", "comments": ""}.

## Хранение индекса

В ходе ЛР для нужд булевого поиск был разработан следующий формат хранения индекса, который состоит из трёх файлов:

* **docs\_id.data** (35 Мб)

Файл служит для отображения индекса документа (doc\_id) в его текстовое представление в файлах \*txt. Поддерживается переменная длина пути до файлов с документами.

*Структура*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n\_docs |  | | | | | | | |
| offset[0] | | offset[1] | … | offset[n\_docs-1] | offset[n\_docs] | |  | |
| n\_chars[0] | | name[0] | | doc\_offset[0] | | doc\_size[0] | |  |
| n\_chars[1] | | name[1] | | doc\_offset[1] | | doc\_size[1] | |  |
| … | | … | | … | | … | |  |
| n\_chars[n\_docs-1] | | name[n\_docs-1] | | doc\_offset[n\_docs-1] | | doc\_size[n\_docs-1] | |  |

*Описание полей*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| n\_docs | uint | Число документов в корпусе |
| offset[0],…,  offset[n\_docs] | uint | Смещения в байтах до строк таблицы с описанием документов, расположенной ниже. Таким образом, если понадобится открыть документ с doc\_id = 5, то можно будет сразу переместить головку диска (fseek) до offset[5], прочитать это поле и сразу сместиться до нужной строки в таблице на offset[5], чтобы попасть в начало n\_chars[5]. offset имеет на один элемент больше чем нужно (offset[n\_docs]), чтобы работала блочная индексация и слияние блоков, о котором позже |
| n\_chars[0],…,  n\_chars[n\_docs-1] | uint | Число символов wchar\_t в абсолютном пути до файла, где хранится документ |
| name[0],…,  name[n\_docs-1] | \*wchar\_t | Абсолютный путь до файла \*.txt в кодировке UTF-16, т.е. два байта на символ |
| doc\_offset[0],…  doc\_size[n\_docs-1] | uint | Смещение в байтах до начала документа в файле \*.txt |
| doc\_size[0],…  doc\_size[n\_docs-1] | uint | Размер документа в байтах. |

* **terms.data** (81 Мб)

Файл служит для хранения словаря с терминами и ссылок на файл с словопозициями. Поддерживается переменная длина термина. Термины упорядочены в лексикографическом порядке.

*Структура*

|  |  |  |  |
| --- | --- | --- | --- |
| n\_terms |  | | |
| n\_chars[0] | | term[0] | offset\_post\_list[0] |
| n\_chars[1] | | term[1] | offset\_post\_list[1] |
| … | | … | … |
| n\_chars[n\_terms-1] | | term[n\_terms-1] | offset\_post\_list[n\_terms-1] |

*Описание полей*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| n\_terms | uint | Число терминов в корпусе/ число списков словопозиций |
| n\_chars[0],…,  n\_chars[n\_terms-1] | uint | Число символов в термине |
| term[0],…,  term[n\_terms-1] | \*wchar\_t | Термин в кодировке UTF-16 |
| offset\_post\_list[0],…,  offset\_post\_list[n\_docs-1] | uint | Смещение в файле со словопозициями |

* **postings\_list.data** (889 Мб)

Файл служит для хранения словопозиций и частот терминов в документе. Слопозиции упорядочены по возрастанию идентификаторов документов.

*Структура*

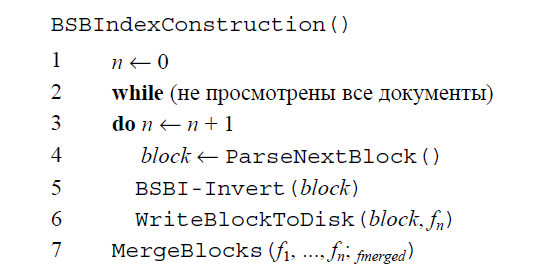
|  |  |  |  |
| --- | --- | --- | --- |
| npl |  | | |
| n\_docs[0] | | docs\_id[0][0],…, docs\_id[0][n\_docs[0]-1] | docs\_freq[0][0],…,  docs\_freq[0][n\_docs[0]-1] |
| n\_docs[1] | | docs\_id[1][0],…, docs\_id[1][n\_docs[1]-1] | docs\_freq[1][0],…,  docs\_freq[1][n\_docs[1]-1] |
| … | | … | … |
| n\_docs[npl-1] | | docs\_id[npl-1][0],…, docs\_id[npl-1][n\_docs[0]-1] | docs\_freq[npl-1][0],…,  docs\_freq[npl-1][n\_docs[npl-1]-1] |

*Описание полей*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| npl | uint | Число терминов в корпусе/ число списков словопозиций |
| n\_docs[0],…,  n\_docs [npl-1] | uint | Число словопозиций и частот термина |
| docs\_id [0],…,  docs\_id [npl-1] | \*int | Вектор идентификаторов документов, в которых встречается термин (слопозиции) |
| docs\_freq[0],…,  docs\_freq[n\_docs-1] | \*int | Вектор частот |

## Алгоритм построения индекса

Был реализован алгоритм блочного индексирования, основанный на сортировке (blocked sort-based indexing), описанный в [1]. В виде псевдокода он описывается следующим образом:



От себя добавлю, что я распараллелил основной цикл по блокам на 4 потока с помощью библиотеки OpenMP, благодаря чему получил ускорение в 4 раза. Блоком в моём случае считается один файл films\*.txt, в котором хранятся документы. Поскольку блоки имеют различные размеры, то для равномерного распределения нагрузки на потоки следует отсортировать блоки по их размеру, например, в порядке возрастания. Этап сортировки термов в лексикографическом порядке моём случае излишен, т.к. я добавлял новые термины и обновлял старые, используя красно-чёрное дерево (std::map), обход которого в порядке Л-К-П даёт отсортированную последовательность термов. Последний этап алгоритма выполняет слияние блоков в однопоточном режиме. Основное время выполнения программы занимает блочная посимвольная обработка документов. Термином считается токен из ЛР1 по курсу ОТЕЯ с пониженной капитализацией.

# Исходный код

## Инструментарий

|  |  |
| --- | --- |
| Инструмент | Назначение |
| #include <stdio.h> | Стандартная библиотека ввода-вывод, в т.ч. UTF-16 |
| #include <iostream> | Стандартная библиотека ввода-вывод, в т.ч. UTF-16 |
| #include <string> | Хранение термов, путей и прочего в контейнере std::wstring |
| #include <set> | Для удобства поиска в небольшой коллекции ключей, фактически «синтаксический сахар» |
| #include <map> | Хранение термов (ключ), словопозиций (значение) и частот (значение) при посимвольной обработке документов |
| #include <filesystem> | Поиск, создание и удаление файлов |
| #include <vector> | Буфера чтения/записи, словопозиции, частоты |
| #include <queue> | Применяется при слиянии блоков. В вектор очередей загружаются все термины из всех блоков. После чего выбирается минимум из элементов в голове очереди, который и будет текущим термином для обработки |
| #include <ctime> | Время выполнения программы |
| #include <time.h> | Вывод раз в секунд текущего прогресса слияния |
| #include <algorithm> | Сортировка файлов по их размеру |
| #include <locale> | Настройка консольного вывода UTF-16 |
| #include <Windows.h> | Работа с кодировками UTF-8 <-> UTF-16 |
| #include <omp.h> | Распараллеливание основного цикла по блокам |

Исходный код доступен в проекте VS 2019 и состоит из одного файла main\_mt.cpp

## Структура main\_mt.cpp

|  |  |
| --- | --- |
| Сигнатура | Назначение |
| #define ERROR\_HANDLE(call, message, ...) | Враппер для экстренного закрытия программы возможно некорретного вызова call |
| #define INFO\_HANDLE(message, ...) | Враппер для логинга в процессе выполнения |
| #define BUF\_SIZE 50000 | Начальный размер буфера для чтения одного документа (предполагается, что один документ может не поместиться в него, поэтому предусмотрен механизм реаллокации) |
| #define OMP\_NUM\_THREADS 4 | Количество потоков |
| set<std::wstring> EXTENSIONS = {  L".json",  L".jsonlines",  L".txt" }; | Множество расширений файлов, подлежащих токенизации (программа будет корректно работать с любыми текстовыми файлами из этого списка) |
| int  string\_to\_wide\_string(const vector<char> &str,  vector<wchar\_t> &wstr) | Преобразование UTF-8->UTF-16 |
| int  wide\_string\_to\_string(const vector<wchar\_t> &wstr,  vector<char> &str) | Преобразование UTF-16->UTF-8 |
| bool get\_doc(FILE \*fp, vector<wchar\_t> &buf) | Считать документ в буфер (с возможной реаллокацией буфера) |
| struct postings\_list  {  vector<int> docs\_id;  vector<int> docs\_freq;  }; | Структура для хранения слопозиций и частот |
| void get\_terms(const vector<wchar\_t> &document,  int doc\_id,  map<wstring, postings\_list \*> &tree) | Вытащить термины, слопозиции и частоты из документа в красно-чёрное дерево |
| int wmain(int argc, wchar\_t \*argv[]) | Главная точка входа в программу |

## Запуск

Исполняемый файл, скомпилированный под ОС Windows 10 лежит в папке \ЛР3\Булев индекс\Release\Булев индекс.exe.

Запуск:

$ ./Булев индекс.exe –i ‘абс. путь к корпусу'

–o ‘абс. путь к индексу’

-t ‘абс. путь к директории с временными файлами’

-create –merge – clear

-create - создать блочный индекс;

-merge - выполнить слияние блочного индекса;

-clear - очистить папку с временными файлами после слияния.

В моём случае программа отработала следующим образом:

$ ./Булев индекс.exe -merge -create -clear -i "D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус" -o "D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\_index" -t "D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\ЛР3\Булев индекс\tmp"

[INFO] Создание индекса для блоков

[INFO] Thread 0 processing block 1/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films13.txt

[INFO] Thread 1 processing block 2/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films9.txt

[INFO] Thread 2 processing block 3/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films10.txt

[INFO] Thread 3 processing block 4/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films12.txt

[INFO] Block 1 has 232216 terms

[INFO] Thread 0 processing block 5/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films1.txt

[INFO] Block 2 has 329619 terms

[INFO] Thread 1 processing block 6/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films2.txt

[INFO] Block 3 has 360379 terms

[INFO] Block 4 has 355242 terms

[INFO] Thread 2 processing block 7/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films11.txt

[INFO] Thread 3 processing block 8/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films3.txt

[INFO] Block 5 has 314758 terms

[INFO] Thread 0 processing block 9/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films4.txt

[INFO] Block 6 has 331747 terms

[INFO] Thread 1 processing block 10/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films8.txt

[INFO] Block 7 has 395161 terms

[INFO] Thread 2 processing block 11/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films5.txt

[INFO] Block 8 has 541555 terms

[INFO] Thread 3 processing block 12/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films6.txt

[INFO] Block 9 has 582062 terms

[INFO] Thread 0 processing block 13/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films7.txt

[INFO] Block 10 has 597482 terms

[INFO] Block 11 has 669497 terms

[INFO] Block 12 has 921883 terms

[INFO] Block 13 has 1383148 terms

[INFO] Создание очередей термов: 13 блок из 13

[INFO] Слияние docs\_id: 13 блок из 13

[INFO] Слияние слопозиций термов

[INFO] Осталось термов: 0

[INFO] Очистка временных файлов

[INFO] Общее число термов в словаре = 2809203

Время выполнения = 145,5 sec, размер корпуса = 2,899 Gb, документов = 186109

Средняя скорость на документ = 0,782 ms

Средняя скорость на килобайт = 0,048 ms

# Выводы

После завершения обработки получились следующие цифры

|  |  |
| --- | --- |
| Общее количество термов | 259’167’384 |
| Размер словаря | 2’809’203 |
| Средняя длина терма в символах | 5,43 |
| Общее время выполнения | 145,5 sec |
| Общий объём обработанных файлов | 2,899 Gb |
| Среднее время обработки КБайта исходного текста документа | 0,048 ms |
| Среднее время обработки одного документа | 0,782 ms |

Средняя длина терма не отличается от средней длины токена, потому что по сравнению с токеном термин лишь отличается пониженной капитализацией (однако средняя длина термина в словаре будет отличаться).

На мой взгляд, работа индексации не совсем оптимальна, т.к. можно распараллелить некоторые этапы алгоритма слияния, но так как основное время работы всё равно занимает обработка блоков, я не стал этого делать. Отмечу, что для получения терма из токена лучше всего было бы подключить какой-нибудь алгоритм NLP (Natural Language Processing).

Можно оптимизировать хранение термов и путей до файлов, путём использования кодов переменной длины (UTF-8).

Сложность алгоритма индексации можно оценить сверху, как , где – число символов в документе, поэтому если размер входных данных увеличится в -раз, то время увеличится в

раз.

В ходе выполнения лабораторной работы я научился обрабатывать текстовые файлы в UTF-8 кодировке, разбивать текст на термины и строить индексы с помощью C++.

# Литература

1. Кристофер Д.Маннинг, Прабхакар Рагхаван, Хайнрих Шютце. Введение в информационный поиск. 2020, изд. Вильямс.