ЛР4: Булев поиск

Задание

Нужно реализовать ввод поисковых запросов и их выполнение над индексом, получение поисковой выдачи.

Синтаксис поисковых запросов:

- Пробел или два амперсанда, «&&», соответствуют логической операции «И».
- Две вертикальных «палочки», «||» – логическая операция «ИЛИ»
- Восклицательный знак, «!» – логическая операция «НЕТ»
- Могут использоваться скобки.

Парсер поисковых запросов должен быть устойчив к переменному числу пробелов, максимально толерантен к введённому поисковому запросу. Примеры запросов:

- [московский авиационный институт]
- [(красный || желтый) автомобиль]
- [руки !ноги]

Должна быть реализована утилита командной строки, загружающая индекс и выполняющая поиск по нему для каждого запроса на отдельной строчке входного файла. В отчёте должно быть отмечено:

- Скорость выполнения поисковых запросов.
- Примеры сложных поисковых запросов, вызывающих длительную работу.
- Каким образом тестировалась корректность поисковой выдачи.

Метод решения

1. Реализация поиска по индексам
2. Реализация операций && и ||
3. Реализация парсера поисковых запросов
4. Сбор статистики, тестирование, оценка результатов

Журнал выполнения

№	Действие	Проблема	Решение
1	Тестирование поиска	Неправильно строится индекс	Отладка индексатора

Реализация

- Поиск по индексам реализован итерационным, т.е. за одну итерацию происходит поиск по одному индексу. После всех итераций результаты поиска для каждого индекса объединяются в один результирующий список. В память одновременно

загружается только один индекс, таким образом мы соблюдаем ограничение по использованию RAM.

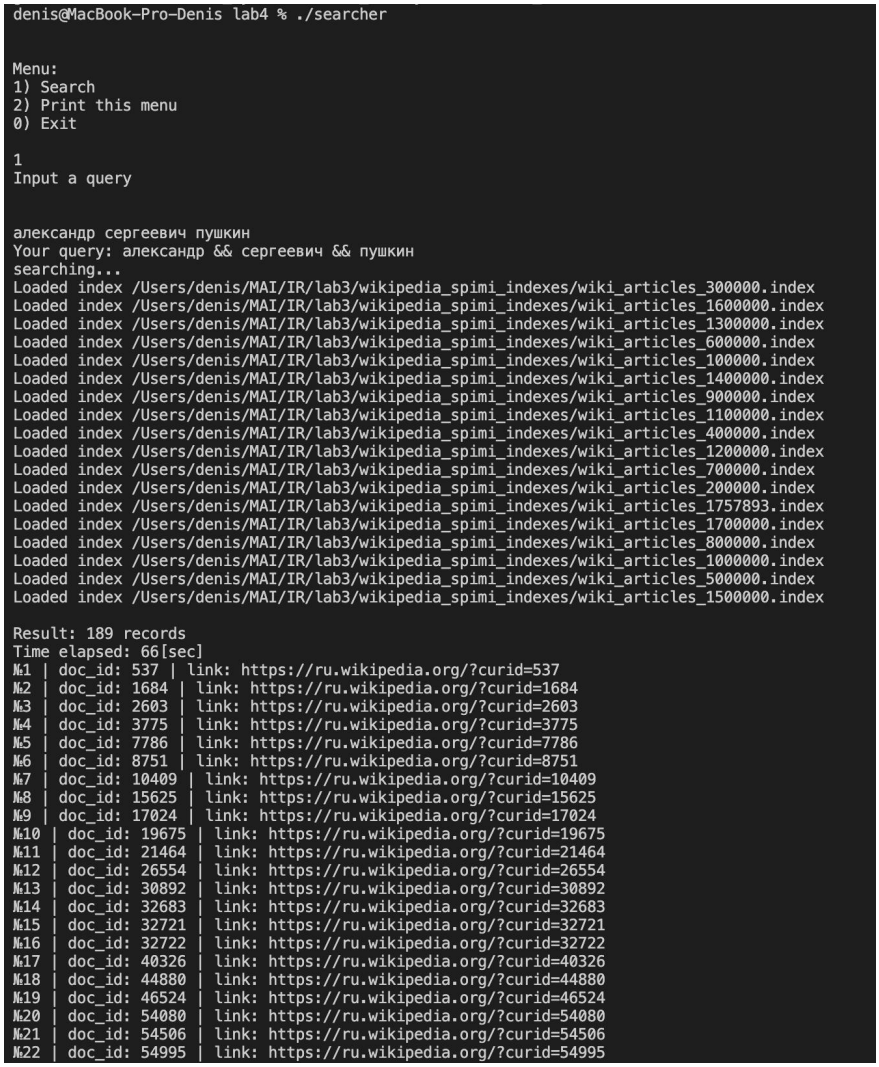
- Реализация операции && состоит в поиске совпадений между двумя списками. Для решения данной задачи отлично подходят множества. Совпадение между списками результатов поиска - есть ни что иное как пересечение двух множеств. Язык c++ обладает возможностью использовать реализацию абстрации множества с операциями пересечения.

- Реализация операции || , главным образом, состоит в объединении двух множеств.

- Релизация парсера поисковых запросов выполняется с помощью алгоритма сортировочной станции. Перед реализацией алгоритма перевода инфиксной записии выражений в постфиксную необходимо предобработать исходные запросы: понизить капиталиацзию, учесть пробелы, убрать лишние знаки.

Результаты выполнения

Корректность результатов тестировалась вручную.



Исходный код

search.h

```h
#ifndef TSEARCHER
#define TSEARCHER

#include <iostream>
#include <vector>
#include <map>


class TSearcher
{
public:
    TSearcher(std::string index_path);
    std::vector<unsigned long long> Search(std::string term);
private:
    void LoadIndex();
    std::string index_path;
    std::map<std::string, std::vector<unsigned long long>> index;
};

#endif // TSEARCHER

```

search.cpp

```cpp
#include <iostream>
#include <sstream>
#include <boost/filesystem.hpp>
#include <boost/range/iterator_range.hpp>
#include <string>
#include <boost/archive/binary_oarchive.hpp>
#include <boost/archive/binary_iarchive.hpp>
#include <boost/serialization/map.hpp>
#include <boost/serialization/vector.hpp>
#include <chrono>
#include <map>
#include <filesystem>

#include "search.h"


TSearcher::TSearcher(std::string index_path) {
    if (!boost::filesystem::exists(index_path)) {
        std::cout << "input file not found" << std::endl;
        return;
    }

    this->index_path = index_path;
    this->LoadIndex();
    std::cout << "Loaded index " << index_path << std::endl;
}

std::vector<unsigned long long> TSearcher::Search(std::string term) {
    auto it = this->index.find(term);
    if (it == this->index.end()) {
        return std::vector<unsigned long long>{};
    }
    return this->index[term];
}


void TSearcher::LoadIndex() {
    std::ifstream f(this->index_path, std::ios::binary);
    if (f.fail()) {
        std::cout << "error" << std::endl;
        return;
    }
    boost::archive::binary_iarchive ia(f);
    ia >> this->index;
}
```

Выводы

В процессе выполнения данной лабораторной работы был реализован и протестирован булев поиск. В целом было интересно реализовывать поиск над статьями википедии. По результатам ЛР сразу видны отличия данной реализации от эталонных поисковых систем: скорость, ранжирование, интерфейс с пагинацией.