

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 абстракции множества с операциями пересечения.

- Реализация операции $| \cup |$, главным образом, состоит в объединении двух множеств.
- Релизация парсера поисковых запросов выполняется с помощью алгоритма сортировочной станции. Перед реализацией алгоритма перевода инфиксной записи выражений в постфиксную необходимо предобработать исходные запросы: понизить капитализацию, учесть пробелы, убрать лишние знаки.

Результаты выполнения

Корректность результатов тестировалась вручную.

```
Vlad@Vlad-MI-LapTop-Pro-15:~/Documents/Учеба/МАН/IR/build_proj$ ./SearchApp
[2021-12-03 21:16:48.810950] [0x00007ff38bf5a740] [trace] {event_type: "performance", event_name: "deserialize_dictionary", event_value: "31 ms"}
[2021-12-03 21:16:48.812805] [0x00007ff38bf5a740] [trace] {event_type: "performance", event_name: "deserialize_doc_index", event_value: "1 ms"}
London Metropolitan Police
9
[2021-12-03 21:17:10.715421] [0x00007ff38bf5a740] [trace] {event_type: "performance", event_name: "search", event_value: "0 ms"}
15956 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=15956
16042 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16042
16202 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16202
16214 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16214
16217 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16217
16219 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16219
16222 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16222
16224 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16224
16226 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16226
16237 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16237
16265 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16265
British && London
4
338
[2021-12-03 21:17:39.572465] [0x00007ff38bf5a740] [trace] {event_type: "performance", event_name: "search", event_value: "0 ms"}
15956 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=15956
15962 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=15962
16042 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16042
16283 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16283
16287 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16287
16459 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16459
16621 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16621
16687 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16687
16792 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16792
16935 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16935
17098 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17098
County || cook
570
571
[2021-12-03 21:18:09.197851] [0x00007ff38bf5a740] [trace] {event_type: "performance", event_name: "search", event_value: "0 ms"}
16049 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16049
16274 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=16274
17553 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17553
17600 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17600
17627 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17627
17642 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17642
17673 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17673
17860 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17860
17871 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17871
17872 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17872
17873 https://en.wikisource.org/wikihttps://en.wikinews.org/wiki?curid=17873
```

Исходный код

```
#include "../bsbi/inv_index_provider.h"
#include "../boolean_queries/expression_tree.h"
#include "../boolean_queries/parser.h"
#include "../boolean_queries/tokenization.h"
#include "../tokenization/word_tokenizer.h"

namespace search_engine {

class SearchEngine {
public:
    explicit SearchEngine(std::shared_ptr<bsbi::InvIndexProvider>);
    std::vector<uint64_t> search(const std::string& query);
private:
    std::shared_ptr<bsbi::InvIndexProvider> invIndexProvider_;
    std::shared_ptr<tokenization::WordTokenizer> wordTokenizer_;
};

} // namespace search_engine
```

```
#include "search_engine.h"

#include "../common/handler.h"

namespace search_engine {

SearchEngine::SearchEngine(std::shared_ptr<bsbi::InvIndexProvider> invIndexProvider)
    : invIndexProvider_(std::move(invIndexProvider))
    , wordTokenizer_(std::make_shared<tokenization::SimpleWordTokenizer>())
{}

std::vector<uint64_t> SearchEngine::search(const std::string& query)
{
    common::PerformanceHandler handler("search");
    auto tokenizedExpression = boolean_queries::tokenizeExpression(query, *wordTokenizer_);
    auto rpnExpression = boolean_queries::reversePolishNotation(tokenizedExpression);
    auto expressionTree = boolean_queries::ExpressionTree(rpnExpression);
    return expressionTree.evaluate(*invIndexProvider_);
}
```

```

int main() {
    auto invIndexProvider = bsbi::createInvIndexProvider("./OUTPUT");
    auto documentIndex = document::createDocumentIndex("./OUTPUT/document_index.bin");

    search_engine::SearchEngine searchEngine(invIndexProvider);

    std::string query;
    while (std::getline(std::cin, query, '\n')) {
        int resultsPrinted = 0;
        for(const auto& docId: searchEngine.search(query)) {
            std::cout << docId << ' ' << URL_PREFIX + documentIndex->url(docId) << '\n';
            resultsPrinted += 1;
            if (resultsPrinted > 10) {
                break;
            }
        }
    }
}

```

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.