

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 с++ обладает возможностью использовать реализацию абстракции множества с операциями пересечения.
- Реализация операции \parallel , главным образом, состоит в объединении двух множеств.
- Реализация 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.

```
denis@MacBook-Pro-Denis lab4 % ./searcher

Menu:
1) Search
2) Print this menu
0) Exit

1
Input a query

александр сергеевич пушкин
Your query: alexandr && sergeevich && pushkin
searching...
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_300000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1600000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1300000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_600000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_100000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1400000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_900000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1100000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_400000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1200000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_700000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_200000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1757893.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1700000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_800000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1000000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_500000.index
Loaded index /Users/denis/MAI/IR/lab3/wikipedia_spimi_indexes/wiki_articles_1500000.index

Result: 189 records
Time elapsed: 66[sec]
#1 | doc_id: 537 | link: https://ru.wikipedia.org/?curid=537
#2 | doc_id: 1684 | link: https://ru.wikipedia.org/?curid=1684
#3 | doc_id: 2603 | link: https://ru.wikipedia.org/?curid=2603
#4 | doc_id: 3775 | link: https://ru.wikipedia.org/?curid=3775
#5 | doc_id: 7786 | link: https://ru.wikipedia.org/?curid=7786
#6 | doc_id: 8751 | link: https://ru.wikipedia.org/?curid=8751
#7 | doc_id: 10409 | link: https://ru.wikipedia.org/?curid=10409
#8 | doc_id: 15625 | link: https://ru.wikipedia.org/?curid=15625
#9 | doc_id: 17024 | link: https://ru.wikipedia.org/?curid=17024
#10 | doc_id: 19675 | link: https://ru.wikipedia.org/?curid=19675
#11 | doc_id: 21464 | link: https://ru.wikipedia.org/?curid=21464
#12 | doc_id: 26554 | link: https://ru.wikipedia.org/?curid=26554
#13 | doc_id: 30892 | link: https://ru.wikipedia.org/?curid=30892
#14 | doc_id: 32683 | link: https://ru.wikipedia.org/?curid=32683
#15 | doc_id: 32721 | link: https://ru.wikipedia.org/?curid=32721
#16 | doc_id: 32722 | link: https://ru.wikipedia.org/?curid=32722
#17 | doc_id: 40326 | link: https://ru.wikipedia.org/?curid=40326
#18 | doc_id: 44880 | link: https://ru.wikipedia.org/?curid=44880
#19 | doc_id: 46524 | link: https://ru.wikipedia.org/?curid=46524
#20 | doc_id: 54080 | link: https://ru.wikipedia.org/?curid=54080
#21 | doc_id: 54506 | link: https://ru.wikipedia.org/?curid=54506
#22 | doc_id: 54995 | link: https://ru.wikipedia.org/?curid=54995
```

Исходный код

```
C search.h
1  #ifndef TSEARCHER
2  #define TSEARCHER
3
4  #include <iostream>
5  #include <vector>
6  #include <map>
7
8
9  class TSearcher
10 {
11 public:
12     TSearcher(std::string index_path);
13     std::vector<unsigned long long> Search(std::string term);
14 private:
15     void LoadIndex();
16     std::string index_path;
17     std::map<std::string, std::vector<unsigned long long>> index;
18 };
19
20 #endif // TSEARCHER
21 |
```

search.cpp

```
1  #include <iostream>
2  #include <sstream>
3  #include <boost/filesystem.hpp>
4  #include <boost/range/iterator_range.hpp>
5  #include <string>
6  #include <boost/archive/binary_oarchive.hpp>
7  #include <boost/archive/binary_iarchive.hpp>
8  #include <boost/serialization/map.hpp>
9  #include <boost/serialization/vector.hpp>
10 #include <chrono>
11 #include <map>
12 #include <filesystem>
13
14 #include "search.h"
15 |
16
17
18 TSearcher::TSearcher(std::string index_path) {
19     if (!boost::filesystem::exists(index_path)) {
20         std::cout << "input file not found" << std::endl;
21         return;
22     }
23
24     this->index_path = index_path;
25     this->LoadIndex();
26     std::cout << "Loaded index " << index_path << std::endl;
27 }
28
29 std::vector<unsigned long long> TSearcher::Search(std::string term) {
30     auto it = this->index.find(term);
31     if (it == this->index.end()) {
32         return std::vector<unsigned long long>{};
33     }
34     return this->index[term];
35 }
36
37
38 void TSearcher::LoadIndex() {
39     std::ifstream f(this->index_path, std::ios::binary);
40     if (f.fail()) {
41         std::cout << "error" << std::endl;
42         return;
43     }
44     boost::archive::binary_iarchive ia(f);
45     ia >> this->index;
46 }
```

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.