

ТЕСТОВОЕ ЗАДАНИЕ СТАЖИРОВКИ JAVA-РАЗРАБОТЧИКОВ

Библиотека `autocomplete` вводимого текста

Постановка задачи

Требуется написать консольное Java-приложение (**JDK 11**), позволяющее быстро искать аэропорты по определенному свойству.

Данные для программы берутся из файла [airports.csv](#). В нем находится таблица аэропортов со свойствами в формате CSV. Название аэропорта — 2 колонка. За что отвечают другие колонки — не важно, они используются для поиска.

Поиск заключается в поиске номеров строк (1 колонка [airports.csv](#)) по колонке, которая указана при запуске программы (нумерация с 1).

Поиск префиксный. Например, если указана колонка поиска 2, поиск по строке **Bow** вернет номера **3600, 4275, 7848**.

При запуске программы указываются следующие параметры:

`--data airports.csv` — путь до файла csv с аэропортами,
`--indexed-column-id 3` — колонка, по которой происходит поиск,
`--input-file input-path-to-file.txt` — путь до файла с входными строками поиска.

Формат — обычный текст. Каждая строка — текст, по которому программа должна выполнить поиск. Пример содержимого файла:

Bower
Asa
Ret

`--output-file output-path-to-file.json` — путь до файла с результатами поиска. Если файла нет, он должен создаваться, если есть, перезаписаться. Формат файла — **json**, содержащий следующие поля:

1. **initTime** — число, время в миллисекундах инициализации от начала запуска программы до готовности к выполнению первого поиска. Может включать в себя в том числе вычитку файла `--input-file`.
2. **result** — массив, каждый элемент которого является результатом поиска по строке файла `--input-file`. У объектов массива есть следующие поля:
 - a) **search** — строка поиска;
 - b) **result** — массив номеров строк, подходящих под поиск, отсортированных по колонке поиска. Сортировка для строковых колонок лексикографическая, для числовых — числовая. Номер строки — первая колонка;
 - c) **time** — число в миллисекундах, затраченное на выполнения поиска по строке.

Пример содержимого файла:

```
{
  "initTime": 100,
  "result": [
    {
      "search": "Bower",
      "result": [1, 2],
      "time": 10
    },
    {
      "search": "Asa",
      "result": [8, 4],
      "time": 10
    },
    {
      "search": "Ret",
      "result": [5, 100],
      "time": 10
    }
  ]
}
```


- / Проверка на ряд условий осуществляется автоматически.
- / На сдачу задания дается 3 попытки.
- / Учитывается лучший результат.
- / Проверка происходит на linux системах.

Процесс проверки

1. Сборка: `mvn clean package`
2. Копирование артефакта `airports-search-*.jar` из папки `target` в директорию проверки.
3. Автоматизированный запуск артефакта под различные критерии проверки: `java -Xmx7m -jar airports-search.jar --data /home/test/airports.csv --indexed-column-id 3 --input-file /temp/input1.txt --output-file /temp/result1.json`
4. Проверка результатов из файлов результатов, которые были созданы приложением.

Нефункциональные требования

1. Перечитывать все строки файла при каждом поиске нельзя.
В том числе читать только определенную колонку у каждой строки.
2. Создавать новые файлы или редактировать текущий нельзя.
В том числе использовать СУБД.
3. Хранить весь файл в памяти нельзя.
Не только в качестве массива байт, но и в структуре, которая так или иначе содержит все данные из файла.
4. Для корректной работы программе требуется не более 7 МБ памяти.
Все запуски `java -jar` должны выполняться с `jvm` флагом `-Xmx7m`.
5. Скорость поиска должна быть максимально высокой с учетом требований выше.
В качестве ориентира можно взять число: на поиск по «Vo», который выдает 68 строк, требуется 25 мс, поиск по «Bower», который выдает 1 строку без фильтров — 5 мс.
6. Сложность поиска меньше, чем $O(n)$, где n — число строк файла.
7. Должны соблюдаться принципы ООП и SOLID.
8. Ошибочные и краевые ситуации должны быть корректно обработаны.
9. Использовать готовые библиотеки для парсинга CSV формата нельзя.

 В случае, если возникает вопрос, который не покрывает данная постановка задачи, выбирайте любое его решение не противоречащее постановке. В `readme` должен быть отражен вопрос и принятое решение.