Hacettepe University

Department of Computer Engineering

BBM204 Software Laboratory II

Spring 2019

Experiment 4

Subject : *String Search and Sort*

Submission Date : *13.05.2019*

Due Date : *03.06.2019*

Programming Language: *Java SE - JDK8*

Background

String Matching

String-matching is a very important subject in the wider domain of text processing. String-matching algorithms are basic components used in implementations of practical software including most operating systems. Moreover, they emphasize programming methods that serve as paradigms in other fields of computer science (system or software design). Finally, they also play an important role in theoretical computer science by providing challenging problems.

Although data is encoded in various forms, text remains the main form to exchange information. This is particularly evident in literature or linguistics where data is composed of huge corpora and dictionaries. This applies to computer science as well where a large amount of data is stored in linear files. Furthermore, the quantity of available data in these fields tend to double every eighteen months. This is the reason why algorithms should be efficient even if the speed and capacity of storage of computers increases regularly.

The **Boyer-Moore** algorithm is considered as one of the most efficient string-matching algorithms in these applications. A simplified version of it or the entire algorithm is often implemented in text editors for the “search” commands. The algorithm scans the characters of the pattern from right to left beginning with the rightmost one. In case of a mismatch (or a complete match of the whole pattern) it uses two precomputed functions to shift the window to the right. These two shift functions are called the good-suffix shift (also called the occurrence shift). [1]

Sorting

In computer science, a sorting algorithm is an algorithm that puts elements of a list in a certain order. The most frequently used orders are numerical order and lexicographical order. Efficient

sorting is important for optimizing the efficiency of other algorithms (such as search and merge algorithms) which require input data to be in sorted lists.

In computer science, a binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child. Binary trees labelled this way are used to implement binary search trees and binary heaps and are used for efficient searching and sorting.

Tree sort is a sort algorithm that builds a binary search tree from the elements to be sorted, and then traverses the tree (in-order) so that the elements come out in sorted order. It's typical use is sorting elements online: after each insertion, the set of elements seen so far is available in sorted order [2].

Country Information Files

Country Information Files (CIF) provides a summary of the demographics, geography, communications, government, economy, and military of international entities in the world.

CIF consists of 188 entities. These entities can be divided into categories. They are:

- Afirca:52
- America:18
- Asia:34
- Australia:19
- Europe:48
- Middle-east:17
  Total: 188

Geographical, population, health, economic and political information belonging to all assets are structurally filed. The file name for each entity is saved as an abbreviation of country name with a two-character letter combination.

Experiment

In this experiment, you will design and implement an application that will search and sort text on CIF data. The categorical information of every entity in the CIF is saved in structural text files.

First of all, your application should do text search on these text files with the Boyer - Moore algorithm. Your application must store the search result in arrays. The arrays that you need to create must contain the answers to the categorical queries. All queries will be made on the following 10 fixed arrays.

- population

```
population:
aa:116,576
ac:95,882
ae:9,701,315
…
```

- area-total
- area-land

- area-water
- median_age-male
- median_age-female
- birth_rate
- death_rate
- literacy-female
- airports

Each array must be sorted in descending or alphabetically order with the ***Tree Sort*** algorithm. After the entities are sorted and listed in each category, you should respond to queries similar to the following.

- birth rate, top, 5 + population, top, 10 (Print the ~~intersection~~ combination of elements of the top 5 in the birth rate category and the top 10 in the population category.)

- area-total, last, 3 + airports, top, 6 (Print the ~~intersection~~ combination of elements of the last 3 in the area-total category and the top 6 in the airports category.)

Query Format :

<category 1>, [top/last], <integer> [+ category 2, [top/last], <integer> [+ category 3, [top/last], <integer>]]
*Optional

Example Queries:

unemployment rate, top, 3
unemployment rate, top, 5 + population, top, 10
unemployment rate, top, 5 + population, top, 10 + Median age, last, 4

At least one, up to three categories will be queried.

It is forbidden to use string functions such as "contains, equal, split, indexof and etc." for sub string search in this assignment. In addition, you should not use java comparator interface.

Input-Output Format

Your program will read folder path of entities and read all files in there. Entity files has categorical and structural textual data. You should construct arrays for each category. Then you should construct a binary search tree for each array.

Example Input File

```
area-total,top,3
area-land,top,2+airports,top,4
area-water,last,2+population,top,3
median_age-female,last,6+population,top,3+birth_rate,top,2
area-land,last,3+death_rate,top,3
```

Example Output File

An output line must be generated for each input line. The names of the countries in the output will be written in alphabetical order. The input and output format is very important, pay special attention to the blanks and misspellings. The program should not crash even if the input file contains incorrect input. The output files will be automatically checked in grading. Therefore, if a character differs between your output and the actual output, your program will be considered as flawed and you will get zero points for that part of the assignment.

```
[Canada, Russia, United States]
[Brazil, Canada, China, Mexico, Russia, United States]
[Algeria, Angola, China, India, United States]
[Burundi, China, India, Malawi, Mali, Niger, Uganda, United States, Zambia]
[Ethiopia, Gibraltar, Latvia, Lithuania, Monaco, Ukraine]
```

Arguments and Execution

>cd n010101010/src

>javac Exp4.java

>java Exp4 folder_of_CIF queries.txt output.out

NOTES

- Use Eclipse for development
- Use UNDERSTANDABLE names for your variables, functions and classes (please be sure you are obeying naming conventions).
- Write READABLE SOURCE CODE blocks.
- Use EXPLANATORY COMMENTS in your source codes.
- Don't miss the deadline.
- CAUTION: Don't start thinking about assignment close to the deadline. Experiment requires less code but a more algorithmic approach.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments will be considered as cheating.
- You can ask your questions through course's piazza group and you should be aware of the discussions in Piazza. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms or source codes.
- You will submit your work from https://submit.cs.hacettepe.edu.tr with the file hierarchy as below:

  |--src

  -- Exp4.java

  -- *.java

REFERENCES

[1] http://www-igm.univ-mlv.fr/~lecroq/string/

[2] https://en.wikipedia.org/wiki/Tree_sort