Project 1 – Automaton. Cloning egrep command supporting simplified ERE.

BM Bui-Xuan

Regular Expressions (RegEx) and ERE specifications : A regular expression is the formalisation of an algebraic language that is acceptable by a deterministic finite automaton (DFA), in the sense of formal language theory. We will only address regular expressions defined by the ERE specifications given in the following link, that we also refer to as RegEx : http://pubs.opengroup.org/onlinepubs/7908799/xbd/re.html

Pattern recognition by a RegEx : Given a text file with l lines, the problem of finding a RegEx in the file is usually decomposed into l times the search for the RegEx in every different line. The general goal of this project is to implement a solution for such a problem. Several strategies can be taken into account, among which, the two strategies presented in the lectures of UE DAAR. However, it is up to each project team to implement their own strategies, provided that what is described in their final report describes indeed what is implemented (! !). The two strategies addressed in the first and third sessions of UE DAAR process as follows.

— If the RegEx is not reduced to a series of concatenations, we first transform it into a syntax tree (1) then into a non-deterministic finite automaton (NDFA) with ϵ-transitions according to Aho-Ullman algorithm (2) then into a deterministic finite automaton (DFA) with the subset method (3) then into an equivalent automaton with a minimum number of states (4) and finally the automaton is used to test whether a suffix of a line of the input text is recognizable by this automaton (5). This is essentially the 1st lecture of UE DAAR.

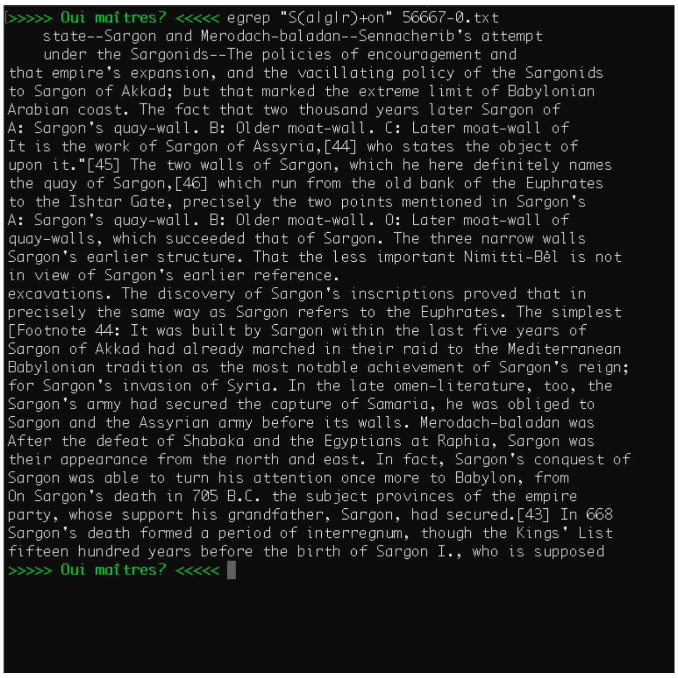
— If the RegEx is reduced to a series of concatenations, our problem is reduced to substring recognition : Knuth-Morris-Pratt (KMP) algorithm can then be used instead of the above described war machine. This is essentially the 3rd lecture of UE DAAR.

— There could also be a third strategy, which is specific to the case when the RegEx is not only reduced to a series of concatenations, but also composed exclusively of alphabetic characters. We assume here that : (forward indexing) the input text file has been pre-processed in order to produce a cache-file containing a table of indices representing the list of all the useful words (in human language) appearing in the text file, these are also called “tokens” ; (inverted indexing) furthermore, the cache-file containing all tokens can be represented by optimised search structures such as a radix tree containing all these words. Then, finding the RegEx is performed by browsing the index table and/or the radix tree.

Performance analysis : Each project team is free to choose its pattern recognition strategies [1], provided that their implementation gives the same result as `egrep` command. In particular, it is required to implement what has been presented in the 1st lecture of UE DAAR as a fallback solution. If a special case of pattern is targeted in order to speed up the implementation, *e.g.* KMP or radix tree, the theoretical approach should be explained in the final report. Then, the performance of this new implementation must be compared to that of the implementation of the 1st lecture of UE DAAR. It could be interesting to compare one’s implementation with the `egrep` command’s performance itself. However, we stress that `egrep` is finely optimized.

WARNING : if you decide to implement KMP or the radix tree method, you must explain it in the final report, and include the performance tests compared to the implementation of the 1st lecture of UE DAAR.

1. We can in particular consider hashtables or Bloom-filters instead of the radix tree.



```
>>>>> Oui maîtres? <<<<< egrep "S(a|g|r)+on" 56667-0.txt
    state--Sargon and Merodach-baladan--Sennacherib's attempt
    under the Sargonids--The policies of encouragement and
that empire's expansion, and the vacillating policy of the Sargonids
to Sargon of Akkad; but that marked the extreme limit of Babylonian
Arabian coast. The fact that two thousand years later Sargon of
A: Sargon's quay-wall. B: Older moat-wall. C: Later moat-wall of
It is the work of Sargon of Assyria,[44] who states the object of
upon it."[45] The two walls of Sargon, which he here definitely names
the quay of Sargon,[46] which run from the old bank of the Euphrates
to the Ishtar Gate, precisely the two points mentioned in Sargon's
A: Sargon's quay-wall. B: Older moat-wall. O: Later moat-wall of
quay-walls, which succeeded that of Sargon. The three narrow walls
Sargon's earlier structure. That the less important Nimitti-Bêl is not
in view of Sargon's earlier reference.
excavations. The discovery of Sargon's inscriptions proved that in
precisely the same way as Sargon refers to the Euphrates. The simplest
[Footnote 44: It was built by Sargon within the last five years of
Sargon of Akkad had already marched in their raid to the Mediterranean
Babylonian tradition as the most notable achievement of Sargon's reign;
for Sargon's invasion of Syria. In the late omen-literature, too, the
Sargon's army had secured the capture of Samaria, he was obliged to
Sargon and the Assyrian army before its walls. Merodach-baladan was
After the defeat of Shabaka and the Egyptians at Raphia, Sargon was
their appearance from the north and east. In fact, Sargon's conquest of
Sargon was able to turn his attention once more to Babylon, from
On Sargon's death in 705 B.C. the subject provinces of the empire
party, whose support his grandfather, Sargon, had secured.[43] In 668
Sargon's death formed a period of interregnum, though the Kings' List
fifteen hundred years before the birth of Sargon I., who is supposed
>>>>> Oui maîtres? <<<<<
```

FIGURE 1 – Example output of UNIX `egrep` command taking as input the RegEx ``S(a|g|r)+on'' and the text file http://www.gutenberg.org/files/56667/56667-0.txt.

1 Specific requirements for project 1

Each project team must provide a clone of the UNIX `egrep` command. However, some conventions of the ERE specification are quite tedious to be fully implemented. We will restrict ourselves to the following elements : parentheses, alternative, concatenation, star, period (*i.e.* the universal character), and any ASCII letter. An example output of the `egrep` command is given in Fig. 1.

A particular effort must be put on the performance test and the final report. Two distinct assessments will be given : one for the implementation works ($\approx 50\%$) and one for the report ($\approx 50\%$).

The following points must be explicitly explained (even for the algorithms already presented in the lectures). Your final report must be self-contained.

— problem definition and data structure to be implemented.

— analysis and theoretical aspects of algorithms solving this problem from literature (including references).

— concise arguments supporting any appreciation, improvement, or criticism about these algorithms.

— test part : how the testbeds are obtained. It is recommended to use the Gutenberg database available at the following address : http://www.gutenberg.org/

— performance analysis : it is better to present diagrams, bar charts (mean + standard deviation), frequency charts, and so on. Not endless columns of numbers. . .

— a discussion about the performance test results is always better than none.

— conclusion and perspectives about the pattern recognition problem.

It is recommended to return between 5 and 10 pages for a report with average content. The formal limitation for this report is 12 pages. This limitation must be observed : pages 13+ will not be read !

Constraints :

— Team size is either 2 or 1.

— Compress all project files into a single file containing the documentation (report ≈ 5-10 pages), a binary and a README explaining how to run the binary, commented source code, a Makefile (or Apache Ant if Java), a directory containing the test instances, and all that can be useful for reading the project without exceeding 10Mo.

- Send the compressed file to `buixuan@lip6.fr`, 3 emails maximum for each project team. The use of online drive (Google, Dropbox, etc) is strictly forbidden. The naming should be daar-project-algorithm-STUDENTNAME1-STUDENTNAME2.piki, where piki could belong to $\{tgz, zip, rar, 7z, etc\}$. This naming convention is important for an automatic classification of your files on the PC of the poor person who will have to assess 70+ student projects...
- Deadline : 09 October 2022, 23h59, by **receiver**'s mail server timestamp. Late report penalty : penalty of $2^{h/24}$ points for h hours late.