Pattern Matching Algorithm Development

Kanumporn Asawasiriroj
Department of Computer Engineering
Faculty of Engineering
Mahidol University, Thailand

Abstract— String matching is a fundamental problem in computer science. The main problem is to find all occurrences in the text file. Normally, string matching algorithms consist of two processes. First is the comparison between each character of a pattern and a substring. Second is to determining the position of the next comparison. Those are called the pre-processing phase and the searching phase. In the searching phase if increase the shifting value, the algorithm must has more performance. This thesis has improved the algorithm by choosing the maximum of shift value and apply with Raita compare ordering technique. The testing environments are simulated and the results are compared with other traditional algorithms and many types of text files used too. From the experimental results, the improved algorithm outperformed other string matching algorithms currently in use in all cases of text files that are tested in this research.

Key words: String Matching, Pattern Matching, Exact String Matching, Single Pattern, Character Comparison

I. INTRODUCTION

The problem of pattern matching or string searching is one of the oldest and most ubiquitous fields in computer science. Applications that require string searching can be found almost everywhere. The pattern matching problem is to find one or more occurrence of the interested string within the text string. String matching is divided into two categories, exact and approximate string matching. Exact string matching, pattern is compared every characters with the selected text string but approximate string matching, if some of pattern matched with selected text then it shows the result.

In this research, we study about exact pattern matching algorithm based on window sliding method which is the important task in various pattern matching algorithm.

Previous pattern matching algorithm used to complete that goal. According to literature review, many of researchers focus to increase the shift value [4]. In this research we improved the algorithm by using the maximum shift value from two shift table of pattern matching algorithms (Boyer-Moore-Horspool [8] and Quick Search [7]) and using the concept of Raita's [9] order comparing. Then compare the results with Boyer-Moore-Horspool, Quick Search and Raita algorithms. Those three algorithms are well known and have good performance in practical cases.

We brief some of pattern matching algorithms in Section II. In Section III describes the concept of improved algorithm and show some example. Then we tested this algorithm along with those three algorithms in terms of time performance on different text corpuses and different pattern length. Finally in Section IV, we conclude the experiments results of comparisons between the improved algorithm and the other three algorithms. And section V is the conclusion.

II. LITERATURE REVIEW

The goal of pattern matching is finding one or more usually all of the occurrence of a given pattern (P = p[0…m-1]) of length m in a given text (T = t[0…n-1]) of length n. All these characters are built over finite alphabet set ∑ which has size equal to σ.

The pattern matching algorithm consists of two main phases. First is pre-processing phase, in this phase algorithms process the pattern and gather the information such as index of character and calculate the shift value to reduce the number of comparisons. And searching phase is the main point of pattern matching by comparison the character between the pattern (P) and text string (T). This phase uses the information collected in pre-processing phase.

Boyer-Moore-Horspool Algorithm uses the rule named bad-character to compute the shift value by comparing the rightmost character in the sliding window. The comparison order is slide from right to left till the match or mismatch happens. Pre-processing phase prepare the bad character shift value and store in the shift table. The shift table was used during the searching phase.

Raita Algorithm uses the same rule as Boyer-Moore-Horspool algorithm to calculate the shift value but the order of comparing is last, first and middle characters of the searched string respectively. If all matched, the procedures continues from the second character to the last character. Therefore the unnecessary comparison was made at the middle character again. Raita algorithm uses the same equation as Boyer-Moore-Horspool algorithm to calculate the shift table.

Quick Search Algorithm uses the same rule as Boyer-Moore-Horspool too but it calculates its shifts with the occurrence shift of the character of the text (T) immediately after the right end of the window.

A. *Pre-processing Phase:*

This phase computes the shift value The value qsBc for a particular alphabet is defined as the position of that character in the pattern from right to left, and if it does not occur in the pattern, then the value is equal to m in Boyer-Moore-Horspool and Raita algorithm and value is equal to m+1 in Quick Search algorithm. Then the skip value for each character is stored in the shift table.

Equation for bmBc Shift Table	**Equation for qsBc Shift Table**
$shift[a] = \begin{cases} m-i-1 & \text{if } P[i]=a \\ m & \text{otherwise} \end{cases}$	$shift[a] = \begin{cases} m-i & \text{if } P[i]=a \\ m+1 & \text{otherwise} \end{cases}$

Table I: Equation for shift table

Example

Let. P = gcagagag. The bmBc shift table of P is displayed as follows:

x	a	c	g	t
bmBc[x]	1	6	2	8

Table II: The bmBc shift table of P = aggttgaat

And qsBc shift table of P is displayed as follows

x	a	c	g	t
qsBc[x]	2	7	1	9

Table III: The qsBc shift table of P = aggttgaat

B. Searching Phase:

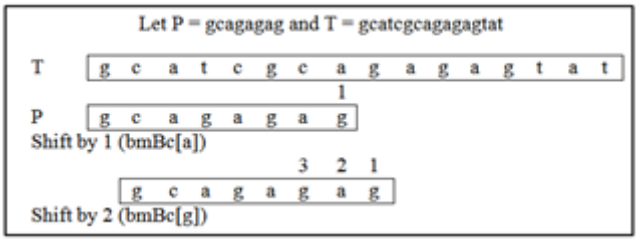

Fig. I: Example of Boyer-Moore-Horspool Algorithm

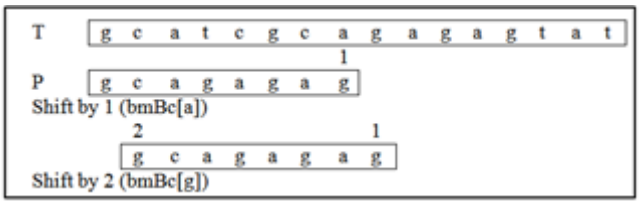

Fig. II: Example of Raita Algorithm

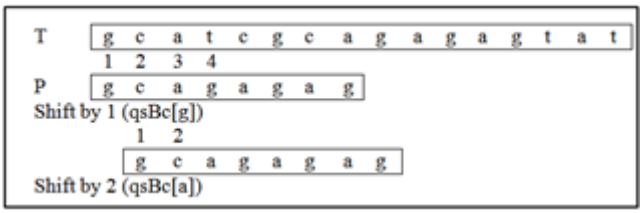

Fig. III: Example of Quick Search Algorithm

Algorithm Name	Comparison Order	Shift Table	Preprocessing		Searching Time Complexity
			Time Complexity	Space Complexity	
Boyer-Moore-Horspool	Right to Left	bmBc table	O(m+σ)	O(σ)	O(mn)
Quick Search	Left to Right	bmBc table	O(m+σ)	O(σ)	O(mn)
Raita	Rightmost -> Leftmost -> Middle	qsBc table	O(m+σ)	O(σ)	O(mn)

Table IV: Algorithm Analysis Table

III. IMPROVED PATTERN MATCHING ALGORITHM

A. Basic Idea

After looking at each algorithm, the interesting thing is some shift value from QsBc and BmBc are not the same so we used this idea to improve new algorithm. Moreover, Horspool and Raita algorithm are used the same shift table but Raita algorithm has the better performance than Boyer-Moore-Horspool so we can assume that Raita's ordering of comparison is better than Boyer-Moore-Horspool algorithm.

If categorize those 3 algorithms based on order or comparison as follows:

- *Right to left:* Boyer-Moore-Horspool algorithm
- *Left to right:* Quick Search algorithm
- *Specific order*
  - Rightmost -> leftmost -> middle: Raita algorithm
  - Leftmost -> rightmost -> middle

if categorize based on shift table

- bmBc, *Horspool algorithm and raita algorithm* are used this table for shifting the pattern P.
- qsBc, *Quick Search algorithm* are used this table for shifting the pattern P.

To see all the result of shift table and ordering combination, we classified all of possible cases as follows:

Named each EXP. for easy to understand as

- EXP.1 -> Boyer-Moore-Horspool Algorithm
- EXP.2 -> QSRL
- EXP.3 -> RLMAX
- EXP.4 -> HSLR
- EXP.5 -> Quick Search Algorithm
- EXP.6 -> LRMAX
- EXP.7 -> Raita Algorithm
- EXP.8 -> QSRLM
- EXP.9 -> AAA
- EXP.10 -> HSLRM
- EXP.11 -> QSLRM
- EXP.12 -> AAA2

So we used that two ideas to improving new algorithm.The program of improved algorithm and the other algorithms are written in C programming by the code of other pattern matching algorithms is developed by Christian Charras and Thierry Lecroq [4] and use smart (string matching research tool) as tool in this study. In this tool contains of 85 pattern matching algorithms' source code and 12 corpus texts. It is a standard framework for researchers, easy to use and support another algorithm you developed.

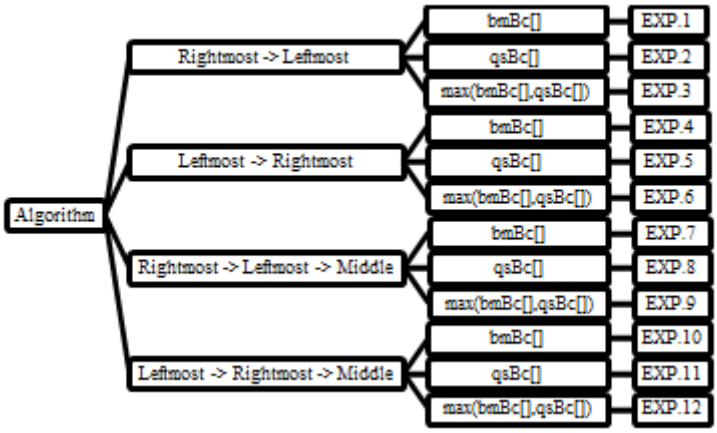

Fig. IV: Classified algorithm

IV. RESULTS AND DISCUSSION

We compare proposed algorithm with existing three algorithms. (Boyer-Moore-Horspool, Raita and Quick Search algorithm) The types of test data we use in testing are:

(1) English text: the English King James version of the Bible composed of 4,047,392 characters. The alphabet σ is 63. And the CIA World Fact Book composed of 2,473,400 characters. The alphabet σ is 94. (6.1 MB)

(2) Genome sequence: the genome sequence of 4,638,690 base pairs of Escherichia coli. The alphabet σ is 4. (σ = {A,G,C,T}) (4.4 MB)

(3) Protein sequence: the protein sequence of 3,295,751 bytes of the Saccharomyces cerevisiae genome. The alphabet σ is 20. (σ = {A,C,D,E, F,G,H,I,K,L,M,N,P,Q,R,S,T,V,Q,Y}) (3.1 MB)

(4) Random text: a random text over an alphabet with a uniform distribution. The alphabet σ size are 4, 8, 16, 32 and 64. (5 MB)

The results are shown in the form of table in table V - XII.

Pattern Length / Algorithms	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
BMH	8.10	5.52	4.02	3.31	2.92	2.69	2.54	2.41	2.37	2.30
QSRL	6.06	4.88	3.86	3.33	3.00	2.79	2.6	2.46	2.41	2.32
RLMAX	6.74	5.02	3.77	3.07	2.72	2.54	2.39	2.3	2.26	2.21
HSLR	7.77	5.39	3.90	3.08	2.74	2.54	2.43	2.33	2.29	2.24
QS	6.14	4.79	3.72	3.04	2.71	2.53	2.42	2.33	2.28	2.24
LRMAX	7.10	5.28	3.92	3.03	2.66	2.51	2.38	2.28	2.24	2.20
Raita	7.25	4.81	3.6	3.00	2.69	2.53	2.41	2.32	2.28	2.23
QSRLM	5.92	4.49	3.56	3.02	2.75	2.6	2.47	2.38	2.32	2.27
AAA	6.59	4.81	3.61	2.95	2.63	2.46	2.31	2.27	2.23	2.19
HSLRM	7.51	5.11	3.74	3.00	2.69	2.51	2.40	2.31	2.27	2.23
QSLRM	5.77	4.45	3.50	2.91	2.63	2.48	2.38	2.29	2.26	2.22
AAA2	6.88	5.04	3.73	2.96	2.64	2.48	2.37	2.27	2.24	2.20

Table V: Result On English Text

Pattern Length / Algorithms	2	4	8	16	32	64	128	256	512	1024

BMH	10.13	8.97	7.43	7.25	7.07	7.47	7.02	6.91	6.98	6.92
QSRL	9.25	9.49	8.17	8.27	8.15	8.44	8.11	7.93	8.09	8.06
RLMAX	9.42	8.22	6.63	6.51	6.34	6.61	6.33	6.24	6.29	6.26
HSLR	12	9.07	6.76	6.23	6	6.29	6.13	6.06	6.11	6.15
QS	9.92	8.05	6.46	6.19	5.98	6.09	6.11	6.05	6.07	6.07
LRMAX	11.28	8.47	6.21	5.69	5.48	5.97	5.89	5.83	5.86	5.87
Raita	10.14	7.35	6.06	5.71	5.51	5.75	5.6	5.5	5.57	5.57
QSRLM	9.6	7.97	6.94	6.78	6.55	6.66	6.67	6.57	6.65	6.6
AAA	9.86	7.39	5.95	5.63	5.39	5.61	5.51	5.43	5.48	5.48
HSLRM	11.26	8.39	6.38	5.83	5.59	5.85	5.68	5.62	5.69	5.69
QSLRM	9.49	7.41	6.08	5.81	5.62	5.7	5.7	5.65	5.68	5.65
AAA2	10.91	7.82	5.87	5.36	5.11	5.32	5.22	5.15	5.21	5.2

Table VI: Result On Genome Sequence

Pattern Length / Algorithms	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
BMH	7.77	5.23	3.77	3.13	2.8	2.67	2.6	2.56	2.56	2.58
QSRL	5.72	4.59	3.58	3.17	2.89	2.76	2.66	2.62	2.62	2.64
RLMAX	6.51	4.86	3.62	2.98	2.66	2.55	2.47	2.44	2.44	2.46
HSLR	7.81	5.04	3.62	2.97	2.66	2.54	2.5	2.46	2.46	2.49
QS	5.95	4.45	3.46	2.92	2.65	2.54	2.49	2.46	2.46	2.48
LRMAX	6.94	5	3.67	2.96	2.62	2.55	2.49	2.45	2.45	2.48
Raita	6.9	4.58	3.41	2.88	2.62	2.53	2.47	2.44	2.45	2.46
QSRLM	5.53	4.23	3.33	2.9	2.67	2.6	2.54	2.51	2.51	2.52
AAA	6.39	4.64	3.49	2.89	2.6	2.5	2.43	2.4	2.4	2.42
HSLRM	7.4	4.82	3.52	2.91	2.63	2.52	2.48	2.45	2.45	2.46
QSLRM	5.51	4.16	3.3	2.83	2.58	2.5	2.45	2.42	2.43	2.44
AAA2	6.61	4.75	3.54	2.9	2.61	2.51	2.46	2.42	2.43	2.44

Table VII: Result On Protein Sequence

Pattern Length / Algorithms	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
BMH	10.3	9.06	7.49	7.33	7.4	6.99	7	7.13	7.04	7.18
QSRL	9.33	9.53	8.42	8.42	8.55	8.58	8.26	8.12	8.29	8.41
RLMAX	9.45	8.18	6.72	6.57	6.64	6.33	6.3	6.37	6.34	6.46
HSLR	12.04	9.04	6.88	6.31	6.25	5.94	6.13	6.23	6.16	6.26
QS	9.99	7.73	6.59	6.24	6.13	6.15	6.14	6.04	6.14	6.21
LRMAX	11.24	8.41	6.34	5.75	5.7	5.75	5.88	5.94	5.91	6
Raita	10.14	7.35	6.14	5.77	5.73	5.52	5.58	5.67	5.59	5.68
QSRLM	9.65	8.01	7.14	6.91	6.79	6.83	6.71	6.59	6.73	6.79
AAA	9.46	7.36	6.06	5.67	5.62	5.46	5.48	5.54	5.52	5.59
HSLRM	11.99	8.37	6.44	5.93	5.86	5.59	5.7	5.77	5.71	5.81
QSLRM	9.57	7.4	6.19	5.88	5.76	5.77	5.73	5.64	5.74	5.8

AAA2	10.85	7.82	5.97	5.42	5.36	5.15	5.21	5.27	5.25	5.32

Table VIII: Result On Rand4

Pattern Length / Algorithms	2	4	8	16	32	64	128	256	512	1024
BMH	8.69	6.21	4.64	4.04	3.91	3.95	3.83	3.78	3.9	3.83
QSRL	6.93	5.92	4.74	4.35	4.26	4.3	4.16	4.11	4.19	4.11
RLMAX	7.35	5.66	4.28	3.69	3.55	3.59	3.5	3.47	3.55	3.48
HSLR	9.32	6.16	4.46	3.71	3.52	3.56	3.5	3.48	3.56	3.5
QS	7.23	5.45	4.26	3.69	3.52	3.57	3.52	3.48	3.53	3.5
LRMAX	8.33	5.96	4.36	3.57	3.35	3.53	3.49	3.43	3.53	3.46
Raita	7.8	5.23	4	3.52	3.38	3.43	3.37	3.33	3.41	3.35
QSRLM	6.69	5.15	4.18	3.75	3.62	3.67	3.62	3.58	3.64	3.59
AAA	7.23	5.23	3.98	3.46	3.3	3.34	3.28	3.25	3.32	3.28
HSLRM	8.76	5.72	4.22	3.59	3.41	3.45	3.4	3.36	3.44	3.39
QSLRM	6.63	5.02	3.98	3.51	3.36	3.41	3.36	3.32	3.37	3.33
AAA2	7.76	5.49	4.11	3.46	3.27	3.3	3.24	3.21	3.3	3.24

Table IX: Result On Rand8

Pattern Length / Algorithms	2	4	8	16	32	64	128	256	512	1024
BMH	7.77	5.22	3.8	3.15	2.89	2.79	2.78	2.78	2.8	2.78
QSRL	5.72	4.58	3.63	3.2	2.98	2.88	2.86	2.87	2.87	2.86
RLMAX	6.48	4.82	3.63	2.98	2.73	2.65	2.63	2.64	2.64	2.63
HSLR	7.85	5.09	3.68	3.01	2.73	2.66	2.65	2.65	2.66	2.64
QS	6.01	4.5	3.51	2.97	2.73	2.65	2.65	2.65	2.66	2.65
LRMAX	6.96	5.01	3.7	2.98	2.69	2.66	2.65	2.67	2.66	2.64
Raita	6.9	4.58	3.44	2.91	2.7	2.61	2.62	2.62	2.63	2.61
QSRLM	5.53	4.23	3.36	2.93	2.76	2.7	2.69	2.7	2.7	2.7
AAA	6.33	4.63	3.5	2.9	2.66	2.58	2.58	2.59	2.59	2.58
HSLRM	7.42	4.85	3.56	2.95	2.71	2.63	2.63	2.63	2.63	2.62
QSLRM	5.54	4.2	3.33	2.87	2.67	2.59	2.59	2.6	2.6	2.59
AAA2	6.63	4.77	3.56	2.93	2.68	2.61	2.59	2.6	2.61	2.57

Table X: Result On Rand16

Pattern Length / Algorithms	2	4	8	16	32	64	128	256	512	1024
BMH	7.20	4.74	3.44	2.81	2.53	2.42	2.37	2.37	2.37	2.37
QSRL	5.06	3.98	3.17	2.80	2.59	2.46	2.41	2.40	2.41	2.40
RLMAX	6.00	4.45	3.38	2.76	2.46	2.35	2.31	2.3	2.31	2.31
HSLR	6.91	4.5	3.31	2.71	2.45	2.35	2.32	2.31	2.31	2.32
QS	5.29	4.01	3.17	2.68	2.44	2.35	2.32	2.31	2.32	2.32
LRMAX	6.21	4.52	3.4	2.76	2.46	2.37	2.33	2.32	2.32	2.32
Raita	6.42	4.26	3.19	2.66	2.43	2.35	2.31	2.31	2.31	2.31
QSRLM	4.96	3.82	3.06	2.63	2.45	2.4	2.37	2.36	2.36	2.36

AAA	5.93	4.35	3.32	2.71	2.43	2.34	2.3	2.29	2.29	2.29
HSLRM	6.69	4.39	3.26	2.69	2.44	2.35	2.32	2.31	2.31	2.31
QSLRM	4.97	3.81	3.06	2.62	2.41	2.33	2.3	2.29	2.29	2.29
AAA2	6.03	4.4	3.36	2.74	2.44	2.37	2.33	2.32	2.33	2.33

Table XI: Result On Rand32

Pattern Length / Algorithms	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
BMH	7.01	4.56	3.31	2.69	2.38	2.28	2.2	2.18	2.18	2.18
QSRL	4.79	3.74	2.99	2.62	2.45	2.31	2.23	2.2	2.2	2.2
RLMAX	5.83	4.33	3.29	2.69	2.37	2.24	2.18	2.15	2.15	2.15
HSLR	6.52	4.28	3.16	2.6	2.33	2.23	2.17	2.15	2.14	2.16
QS	5.02	3.81	3.03	2.57	2.32	2.24	2.17	2.16	2.15	2.15
LRMAX	5.9	4.34	3.31	2.68	2.36	2.26	2.18	2.16	2.15	2.15
Raita	6.26	4.15	3.1	2.57	2.32	2.23	2.18	2.16	2.16	2.15
QSRLM	4.74	3.65	2.94	2.52	2.31	2.27	2.22	2.19	2.19	2.19
AAA	5.79	4.27	3.26	2.66	2.36	2.24	2.17	2.15	2.14	2.15
HSLRM	6.4	4.22	3.13	2.59	2.33	2.24	2.18	2.16	2.15	2.16
QSLRM	4.5	3.66	2.94	2.52	2.29	2.22	2.17	2.15	2.15	2.14
AAA2	5.81	4.29	3.27	2.67	2.35	2.27	2.19	2.16	2.16	2.17

Table XII: Result On Rand64

400 different strings are searched in each corpus for each string length, total searching time is given in different tables above.

The searching time results of the corpuses are given in table V-XII. Highlight cells of the tables indicate the best performance for each corpus.

Firstly, in the case of English Texts, the small-middle pattern length (2-32 characters), the algorithm QSLRM performs better for search strings then for pattern length more than 32 characters the algorithm AAA performs better for searching of varying length. Moreover AAA2 show better timing as well.

Secondly, in the case of the genome sequence the algorithm QSRL performs better for 2 characters pattern length then AAA2 has the best performance of various length up to 1024 characters pattern length followed by AAA.

Thirdly, in the case of protein sequence, for the small pattern length (2-8 characters) the algorithm QSLRM performs better timing over other algorithms then for pattern length more than 8 characters, the algorithm AAA has the best performance of various length up to 1024 characters followed by AAA2.

Fourthly, in the case of random texts, we can conclude that for the small pattern length (2-8 characters) the algorithm QSLRM is always has better timing over other algorithms. And we can note that when testing the small random text (2-8 characters) along with middle to large pattern length , the algorithm AAA2 always has better performance over other algorithms but for the longer random text (more than 8 characters) the algorithm AAA2 always has a better performance over other algorithms.

At the outset, QSLRM work well on the small pattern length in every case and for the corpuses that have the number of alphabet between 2-8 characters then for the pattern length more than 8 characters, the algorithm AAA2 work better than other algorithms. In addition for corpuses that have number of alphabet more than 8 characters, the algorithm AAA works better than other algorithms.

It's interesting to note that for corpuses that have number of alphabet more than 8 characters, the algorithm AAA works better than the algorithm AAA2.

V. CONCLUSIONS

This research presents a new idea to choose the maximum shift value from both shift table (bmBc and qsBc) and by using the Raita's compare ordering. The preprocessing phase compute the shift value and then choose the maximum value instead of used one of them. The order of comparing is also the factor of performance. The results from various text corpuses show that this improved algorithm is faster than existing algorithms especially when mid and long pattern is used. Moreover it shows that the speed of pattern matching algorithm can be changed when the structure of text is changed. The results can be used to determine which pattern matching algorithm is more appropriate for a particular text.

REFERENCES

[1] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Introduction to Algorithms, Chapter 34, MIT Press, 1990, pp 853-885.

[2] Christian Charras and Thierry Lecroq, Handbook of exact string matching algorithms, http://www-igm.univ-mlv.fr/~lecroq/string/, 1997.

[3] R. NIGEL HORSPOOL, "Practical fast searching in strings", Software — Practice and Experience, Vol. 10, No. 6, 1980, 501 - 506.
[4] Simone Faro and Thierry Lecroq, SMART string matching research tool website. [Online]. Available: http://www.dmi.unict.it/~faro/smart/index.php
[5] SIMONE FARO, THIERRY LECROQ, The Exact Online String Matching Problem: a Review of the Most Recent Results, [Online]. Available: www-igm.univ-mlv.fr/~lecroq/articles/acmsurv2013.pdf
[6] Smith P. D., "Experiments with a very fast substring search algorithm", Software — Practice and Experience, Vol. 21, No. 10, 1991, 1065-1074.
[7] Smit G. de V., "A comparison of three string matching algorithms", Software — Practice and Experience, Vol. 12, 1982, 57-66.
[8] Sunday, D.M., "A very fast substring search algorithm", Communication of the ACM, Vol. 33, No. 8, 1990, pp.762-772.
[9] R. N. Horspool, "Practical fast searching in strings", Software —Practice and Experience, Vol. 10, No. 3, 1980, 501-506.
[10] TIMO RAITA, "Tuning the Boyer-Moore-Horspool String Searching Algorithm", SOFTWARE — PRACTICE AND EXPERIENCE, VOL. 22(10), OCT-1992, pp.879-884.

