BAD CHARACTER SHIFT BASED STRING MATCHING ALGORITHMS: CRITICAL REVIEW

[1]Anushka Dixit, [2]Rinki Singh
[1]M.Tech Scholar, [2]Assistant Professor
[1]Department of Computer Science and Engineering,
[1]VIET, G.B. Nagar, INDIA

Abstract*— Collection of genome sequences from different organisms and performing matching for finding similarity between them sets the requirement of pattern matching algorithms where one or more than one occurrences of pattern are found within a large body of text. Pattern matching algorithms can be exact, parameterized and approximate. In this paper, we will discuss about most efficient exact pattern matching algorithms. We are taking several algorithms present in the literature right from Horspool to TVSBS for performing comparison on them depending on the pattern size and text size. Algorithms are being analyzed by taking different patterns of different length. An algorithm with less number of character comparisons and less number of passes in proportion provides better result.

***Index Terms*— Character comparison, exact pattern matching, Horspool algorithm, TVSBS, Bad character shift.**

I. INTRODUCTION

String-matching is important in the wider domain of text processing. They play an important role in theoretical computer science by providing solution to challenging problems. First computer programs to use pattern matching were text editors, which uses Boyer Moore algorithm [1]. Later on searching amino acid sequence patterns in genome and protein sequence databases started using concept of pattern matching [2] [3]. Pattern matching algorithms are divided into three categories: Exact, Approximate and Parameterized. Exact matching [4] [5] finds all the exact occurrences of a pattern inside a large body of text. Approximate matching [6] uses edit distance to find the number of matches. Parameterized matching [7] [8] use bijective matching to find the number of occurrences of a pattern inside the body of a text.

In the past, various algorithms have been developed and analyzed [9] [10]. Sunday et al developed a very fast algorithm called quick search [5], which uses only bad character shift. This algorithm was enhancement of Boyer Moore [1] algorithm. Later on in [11], Boyer Moore was further modified by Horspool algorithm. It also uses bad character shift for shifting the window, but consider the character next to window for shifting, whereas quick search uses last character of window for shifting the window. Boyer Moore algorithm was further modified by Berry and Ravindran [12], where shift is performed by two characters immediately to the right of window. Later on, the quick search algorithm was further extended by SSABS [13] to get the better shift. This SSABS algorithm was further extended by TVSBS [14], which inherits property of Berry-Ravindran and SSABS algorithm and was claimed to have better results.

This paper is organized as follows. Sec. 2 presents related terminologies. Sec. 3 presents the literature review. Sec. 4 presents the working example and their analysis. Finally, we conclude in Sec. 5.

II. RELATED TERMINOLOGIES

Exact Pattern Matching

Exact pattern matching finds all occurrences of a single pattern in the text. The pattern is denoted by x=x [0 .. m-1] and its length is equal to m. Text is denoted by y=y [0 .. n-1] and its length is equal to n. Scanning of text is performed by first aligning left ends of the window and the text, then comparing characters of window with the characters of the pattern known as *attempt.* Same procedure is followed until right end of the window goes beyond the right end of the text. Different algorithms work according to changes in their searching phase and shift distance either from left to right or from right to left.

Preprocessing Phase

This is the first phase in every of the pattern matching algorithms. In this phase a table is constructed based on different functions used in different algorithms.

Searching Phase

This is the second phase of algorithm. In this phase searching of the pattern is started in the given text. Shifting of window is done based on the shift values calculated and written within the table. Different algorithms use different techniques for shifting of window.etc.

III. LITERATURE REVIEW

Pattern matching algorithms are used for finding several substrings within text or sequences provided. Many Exact pattern matching algorithms are reported in the literature depending on their number of shift values. Algorithms with large shift values are far better than those having small shift value. Further we have discussed some exact pattern matching algorithms in concise.

Quick Search Algorithm: *Quick Search* [5] algorithm uses only bad character shift table (qsBc) during Pre-processing phase to store the shift values corresponding to each character in the pattern. Time complexity for quick search algorithm is O (m+σ) in preprocessing phase and O (m*n) in searching phase. Shift values are given in the pattern from right to left. Provided pattern length as m, if any character is present in the text and absent in the pattern then shift value corresponding to that character is m+1. In searching phase, comparisons are done between pattern and text from left to right. Shifting is done depending on the shift value of character lying just rightmost to the window where window size is equal to the pattern length. Its main characteristics are:

- Fast over large alphabets
- Very fast for short patterns

- Easy to implement
- It uses only bad character shift

Horspool Algorithm: Horspool Algorithm [11] was published by Nigel Horspool in 1980. It bases the shift on a single character based on bad character. In pre-processing phase, shift table is constructed by finding out shift values of each character in the text provided. Further in searching phase, bad-character shift for the rightmost character within the window is found which is fair enough to compute the value of shift. Time complexity is O (m+σ) in preprocessing phase and O (m*n) in searching phase. Its main characteristics are:

- Good for large texts
- Good for small patterns
- Easy to implement
- Uses only bad-character shift

Berry and Ravindran Algorithm: Berry and Ravindran [12] designed an *algorithm* which performs shifts by considering the bad character shift. It uses two dimensional arrays which are used for calculating shift values in the preprocessing phase. Time complexity is O (m+σ2) in preprocessing phase and O (m*n) in searching phase. It constructs Berry-Ravindran bad character table using brBc function that works for each pair of character in the pattern. In searching phase, shift value of pattern depends on the two successive characters in the text immediately to the right of window. This reduces number of comparisons between text and pattern. Its main characteristics are:

- Reduces number of comparisons
- It uses Quick search and Zhu Takaoka algorithm

SSABS Algorithm: SSABS algorithm was proposed by S. S. Sheik, Sumit K. Aggarwal,†Anindya Poddar, N. Balakrishnan,and K. Sekar in 2004 [13].SSABS algorithm uses Quick-Search bad character (qsBc) shift table for calculating the shift values corresponding to characters present in the pattern. In the searching phase, new order for comparision is applied. First the rightmost character of window is compared to that of pattern. If match is found then first character of the window is compared again if a match is found then from right remaining characters are compared towards left. Time complexity of preprocessing phase for this algorithm is O (m+σ).

TVSBS Algorithm: TVSBS algorithm was proposed by Thathoo Rahul, Virmani Ashish, Sai Lakshmi S., Balakrishnan N., Sekar K. in 2006 [14].This algorithm is a combination of Berry-Ravindran and SSABS as it uses brBc function for constructing the shift table in preprocessing phase and implementing concept of SSABS while performing shifting in the searching phase so as to reduce time complexity from O (m+σ) to O (σ+σ*k). Using brBc function provides maximum skip of window over Quick-Search bad character and Boyer-Moore bad character. Its main characteristics are:

- Its average time is less that SSABS algorithm
- Uses Berry-ravindran and SSABS algorithm
- Uses less number of character comparison
- It stops working when reaches n-m+1 position..

IV. WORKING EXAMPLE & COMPLEXITIES

This section shows the functioning of all five algorithms by taking a text and a pattern and applying different techniques used in algorithms. We will get difference in the number of comparisons and number of passes which will specify the better algorithm of all five.

Text: AACATAGAGAAACATAGAGAACATAG
Pattern: ATCTGACG (Length=8)
Quick- Search Algorithm: Constructing first the quick search bad character table in preprocessing phase.

a	A	G	C	T
qsBc(a)	3	1	2	5

Searching Phase
First pass:

```
[AACATAGA]GAAACATAGAGAACATAG
 12
 ATCTGACG
```

There is a mismatch on second comparision. Shift is done based on G, qsBc [G] = 1

Seventh pass:

```
 AACATAGAGAAACATAGA[GAACATAG]
                    1
                    ATCTGACG
```

Number of comparisons: 12
Number of passes: 7

Berry-Ravindran Algorithm: Constructing table using brBc function

brBc(a)	A	T	C	G
A	9	8	3	10

T	9	10	7	5
C	9	6	10	2
G	1	1	1	1

First pass:

```
[AACATAGA]GAAACATAGAGAACATAG
 12
ATCTGACG
```

There is a mismatch on second comparision. Shift is done based on brBc [G][A]=1

Fourth pass:

```
AACATAGAGAA[ACATAGAG]AACATAG
           1 2
           ATCTGACG
```

Number of comparisons: 8
Number of passes: 4

SSABS Algorithm: qsBc ( ) function is used
First pass:

```
[AACATAGA]GAAACATAGAGAACATAG
        1
ATCTGACG
```

There is a mismatch on first comparision. Shift is done based on qsBc (G) = 1

Eighth pass:

```
AACATAGAGAA[ACATAGA[GAACATAG]
                   2      1
                   ATCTGACG
```

Number of comparisons: 13
Number of passes: 8

TVSBS Algorithm: Table is constructed using brBc function as in Berry-Ravindran
First pass:

```
[AACATAGA]GAAACATAGAGAACATAG
        1
ATCTGACG
```

There is a mismatch on second comparision. Shift is done based on brBc [G][A]=1

Fourth pass:

```
AACATAGAGAA[ACATAGAG]AACATAG
           2     3 1
           ATCTGACG
```

Number of comparisons: 8
Number of passes: 4

Horspool Algorithm: Uses Boyer –Moore bad character shift to draw table in preprocessing phase

a	A	G	C	T
bmBc (a)	2	3	1	4

First pass:

```
[AACATAGA]GAAACATAGAGAACATAG
        1
ATCTGACG
```

There is a mismatch on first comparision. Shift is done based on rightmost character within window so, bmBc [A] = 2
Tenth pass:

```
AACATAGAGAAACATAG[AGAACATA]G
                         1
```

ATCTGACG

Number of comparisons: 10
Number of passes: 10

V. Comparison

Assume **C-** Number of comparisons, **P** -Number of passes and pattern length=**8**.

Table 1: Comparison among algorithms for pattern length = 8

Algorithms	**C/P**	**C/P**	**C/P**	**C/P**	**C/P**	**C/P**	**C/P**	**C/P**	**Avg. of C**	**Avg. of P**
Horspool	20/5	38/10	10/8	10/10	9/9	11/7	8/5	12/9	14.75	7.875
Quick Search	19/5	18/3	6/4	12/7	15/7	15/11	3/3	12/10	12.5	6.25
Berry-Ravindran	19/5	18/3	3/3	8/4	15/7	14/6	2/2	2/2	10.125	4
SSABS	22/5	19/3	8/7	13/8	11/7	21/11	5/3	14/10	14.125	6.75
TVSBS	22/5	19/3	5/4	8/4	2/2	12/6	4/2	2/2	9.25	3.5

Table 2: Comparison among algorithms for pattern length = 10

Algorithms	**C/P**	**C/P**	**C/P**	**C/P**	**C/P**	**C/P**	**C/P**	**C/P**	**Avg. of C**	**Avg. of P**
Horspool	2/2	8/8	6/5	2/2	15/6	4/3	4/4	8/8	6.125	4.75
Quick Search	4/2	12/8	4/4	4/3	7/6	6/2	4/4	8/8	6.125	4.625
Berry-Ravindran	7/2	3/3	2/2	2/2	5/5	6/2	3/2	3/3	3.875	2.625
SSABS	2/2	12/8	4/4	5/3	15/7	4/2	5/4	10/8	7.125	4.75
TVSBS	2/2	3/3	3/2	2/2	10/5	2/2	3/2	3/3	3.5	2.625

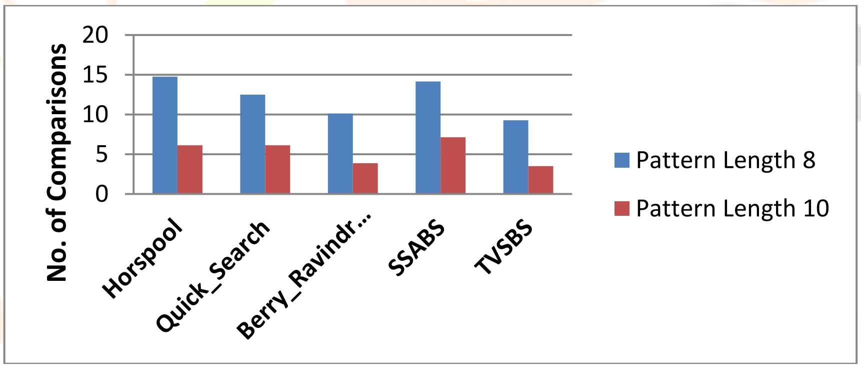

Fig.1 Average no. of comparisons for different algorithms

Table 3: Complexity Analysis

Algorithms	**Preprocessing phase**	**Searching phase**
Horspool	O(m+σ)	O(m*n)
Quick Search	O(m+σ)	O(m*n)
Berry-Ravindran	O(m+σ2)	O(m*n)
SSABS	O(m+σ)	---
TVSBS	O(σ+σ*k)	----

VI. Conclusion

This paper presents state of art comparison of the very fast algorithms on exact string matching. The algorithms have been analyzed on the basis of their time complexity and number of characters comparison. Table 1 and 2 shows thenumber of character comparison, algorithms are taking. The same is shown with the help of a graph in figure 1 for better clarity. Table 3 shows their asymptotic analysis. Out of all the algorithms discussed in this paper, performance of TVSBS is better. In near future, algorithms can be tested on increased size of patterns and text. The algorithm can be implemented on RNA and other huge data sets.

References

[1] Boyer, R. S., And Moore, J. S. A fast string-searching algorithm. Communication of ACM 20, 10 (1977), 762-772.
[2] Aoe, Jun-ichi, 1951- *Computer algorithms : string pattern matching strategies*. IEEE Computer Society Press, Los Alamitos, Calif, 1994
[3] Aho, A. V. "Algorithms for finding patterns in strings Handbook of Theoretical Computer Science, Vol. A, J. van Leeuwen." (1990).
[4] Knuth, D. E. Fast Pattern Matching in Strings.SIAM Journal of Computing 6, 2 (1977), 323-50.

[5] Sunday, D. M. A very fast substring search algorithm. Communications of the ACM 33, 8 (1990), 132-142.

[6] Wu, Sun, and Udi Manber. "Agrep–a fast approximate pattern-matching tool."*Usenix Winter 1992* (1992): 153-162.

[7] Baker, B. S. Parameterized duplication in strings: algorithms and an application to software maintenance.SIAM Journal of Computing 26, 5 (1997), 1343-1362.

[8] K. Fredriksson and M. Mozgovoy. Efficient parameterized string matching. IPL: Information Processing Letters, 100, 2006.

[9] Tevatia, S., Prasad, R. and Rai, D., 2013, August. An offensive algorithm for multi-pattern parameterized string matching. In *Control Computing Communication & Materials (ICCCCM), 2013 International Conference on*(pp. 1-5). IEEE.

[10] Tevatia, Swati, and Rajesh Prasad. "Multi-patterns parameterised matching with application to computational biology." *International Journal of Information and Communication Technology* 7.4-5 (2015): 469-480.

[11] Horspool, R. N. Practical fast searching in strings. Software-Practice & Experience 10, 6 (1980), 501-506.

[12] Berry, Thomas, and S. Ravindran. "A Fast String Matching Algorithm and Experimental Results." *Stringology*. 1999.

[13] S. S. Sheik, S. K. Aggarwal, A. Poddar, et al, "A fast pattern matching algorithm", Journal of Chemical Information and Computer Sciences, (44):1251-1256, 2004.

[14] Thathoo, Rahul, et al. "TVSBS: A fast exact pattern matching algorithm for biological sequences." *CURRENT SCIENCE-BANGALORE-* 91.1 (2006): 47.