Survey on Exact Pattern Matching Algorithm

Urmila Patel[1] Mr. Mitesh Thakkar[2]
[1]M.E. Scholar [2]Assistant Professor
[1,2]computer Engineering Department
[1, 2] L J Institute of Engineering and Technology Ahmedabad, Gujarat, India

Abstract— In today's scenario many computer science field working on the Pattern matching problem like Intrusion Detection System, Search in Text Editor, DNA Sequence Match, Digital Libraries and many more. Exact Pattern Matching Problem defined as a finding Pattern P [1...m] in to long Text String T [1...n] with its all occurrence of exact match where n>>m. This Paper having survey of Exact string matching algorithm with their working methodology, its pre-processing logic, complexity of both the pre-processing and searching phase and also the comparison table for the same . According to this survey, all the researcher focus to reduce number of character comparisons and pre-processing time.

Key words: Algorithm, Complexity, Pattern Window, Text Window.

I. INTRODUCTION

Pattern Matching problem broadly categorized in to Exact Pattern Matching and Approximate Pattern Matching. Exact Pattern match defined as finding exact match of Pattern in to Text String and its applicability in Intrusion Detection System, Text Editor, plagiarism and many other areas. Approximate pattern matching defined as finding Pattern window into text string with edit distance and its applicability in Bioinformatics, Video Retrieval. Pattern matching problem also divide based on the application like single pattern match or multi pattern match.

II. PREMILINARIES

In the context of Exact Pattern Matching algorithm, following are description of various terminologies which is prerequisite to understand the algorithm.

A. Variable

There are various variable used and described below

1) P : Pattern
2) m: Pattern Length
3) T: Text String
4) n: Text Length

B. Definitions

1) Algorithm:
Algorithm defined as Set of rules to solve the problem.

2) Space Complexity:
Space complexity defined as the space or memory required to run the algorithm or solve the problem

3) Time Complexity:
Time complexity defined as the total amount of time taken by the algorithm to run or solve the problem.

4) Searching Phase:
Searching phase is the main area of Pattern matching algorithm in which comparison of Text string and Pattern is done.

5) Pre-Processing Phase:
Pre-Processing Phase defined as the process of finding next shift decision for comparison of Pattern window with Text window in searching phase.

III. EXACT PATTERN MATCHING ALGORITHM

There are many algorithms for finding Pattern P [1...m] into given Text String with its all occurrence of exact match where n>>m.

Following are Exact Pattern Matching algorithm with their detailed description of algorithm logic, complexity of both phase (Searching phase and Pre-processing phase) and also the order of comparison in which the algorithm works.

A. Brute Force (BF) Algorithm

From the study of [1,2,3,6] , Brute Force (BF) Algorithm is the basic algorithm for Exact String Matching in which first letter of pattern window will match with given text window, if match then check next character of pattern with text otherwise shift the pattern window with one.

This algorithm does not having any pre-processing phase for shifting logic of window so every time window will shift with one character -only and also not require any extra space. The time complexity of this algorithm is O (mn).

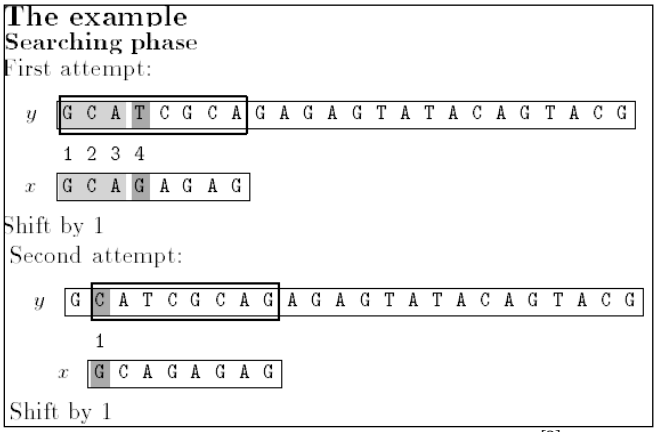

Fig. 1: Example of Brue Force Algorithm [2]

B. Morris-Pratt (MP) Algorithm

From the study of [2], Morris-Pratt (MP) algorithm matches the pattern with text window from left to right character by character. It performs at most 2n-1 text character comparisons during searching phase. Pre-processing phase calculate the shift logic of pattern charter which knowledge used in searching phase of algorithm.

The extra space require for that is O (m) and the time complexity of pre-processing is O (m) and searching complexity is O (m+n).

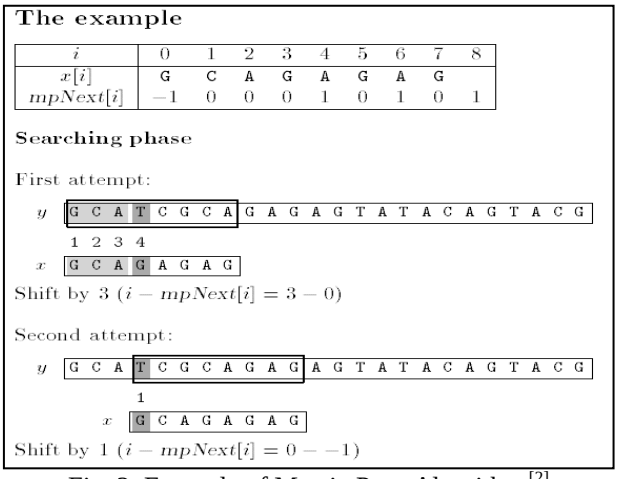

Fig. 2: Example of Morris-Pratt Algorithm [2]

C. Knuth Morris-Pratt (KMP) Algorithm

From the study of [1,2,3,6,7], Knuth Morris Pratt (KMP) algorithm is proposed in year 1977.This algorithm match the pattern with text window from left to right character by character. This algorithm has the Pre-processing logic for taking the decision for shift, so it uses the knowledge of shift in the searching phase.

The extra space require for that is O (m) and the time complexity of pre-processing is O (m) and searching complexity is O (mn).

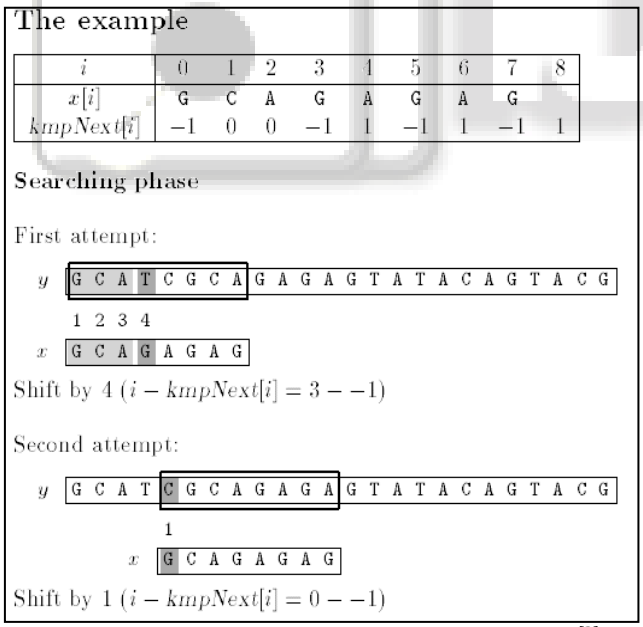

Fig. 2: Example of Knuth Morris-Pratt Algorithm [2]

D. Boyer-Moore (BM) Algorithm

From the study of [2, 3, 6, 8], Boyer-Moore (BM) algorithm is a practically efficient string matching algorithm in usual applications. This algorithm match the pattern with text window from right to left character by character and starts with right most character. This algorithm has two logic functions for shift window to the right which used when complete match or mismatch occur. Good suffix shift and Bad character shift both used in this algorithm.

Its time and space complexity of pre-processing phase is O (m+|Σ|).Its Searching phase complexity is O (mn).Best case Performance is O (n/m).

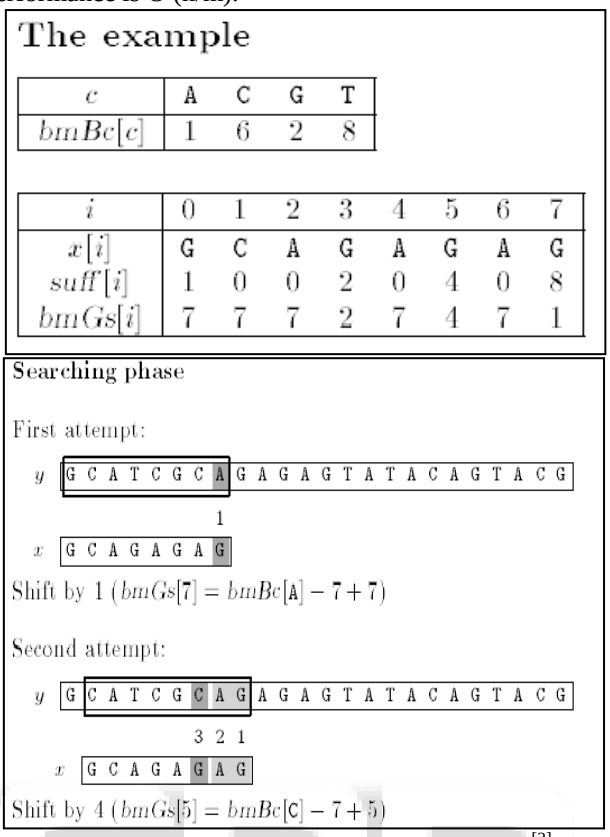

Fig. 3: Example of Boyer-Moore Algorithm [2]

E. Boyer-Moore Horspool (BMH) Algorithm

From the study [2, 3, 6, 10], Boyer-Moore Horspool (BMH) is simplified version of Boyer-Moore algorithm and easy to implement. This algorithm only use bad-character shift for computing shift. Pre-processing phase prepare the bad character shift value and that table used during the searching phase of algorithm.

Its pre-processing time complexity is O (m+|Σ|) and space complexity is O (Σ|).Searching phase complexity is O (mn).

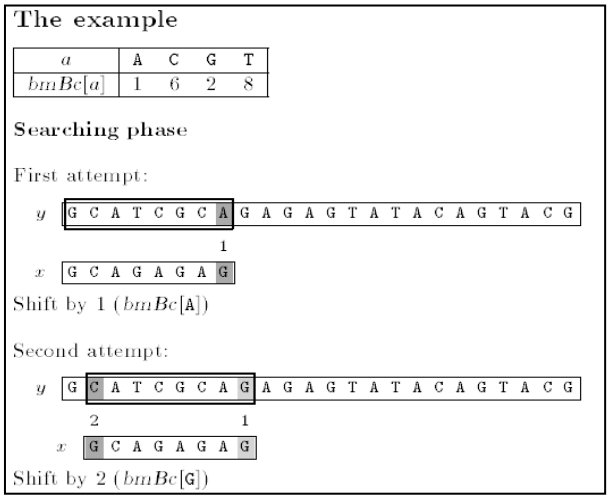

Fig. 4: Example of Boyer-Moore Horspool Algorithm [2]

F. Raita Algorithm

From the study of [2,11], Raita algorithm which first compare last character of pattern window to pattern, then if

they match compare the first character of pattern window, then if they match compare middle character of pattern to text window. And finally if they match compare other character of pattern window from second last character but possibly compare middle character again.

Its pre-processing time complexity is O (m+|Σ|) and space complexity is O (Σ|).Searching phase complexity is O (mn).

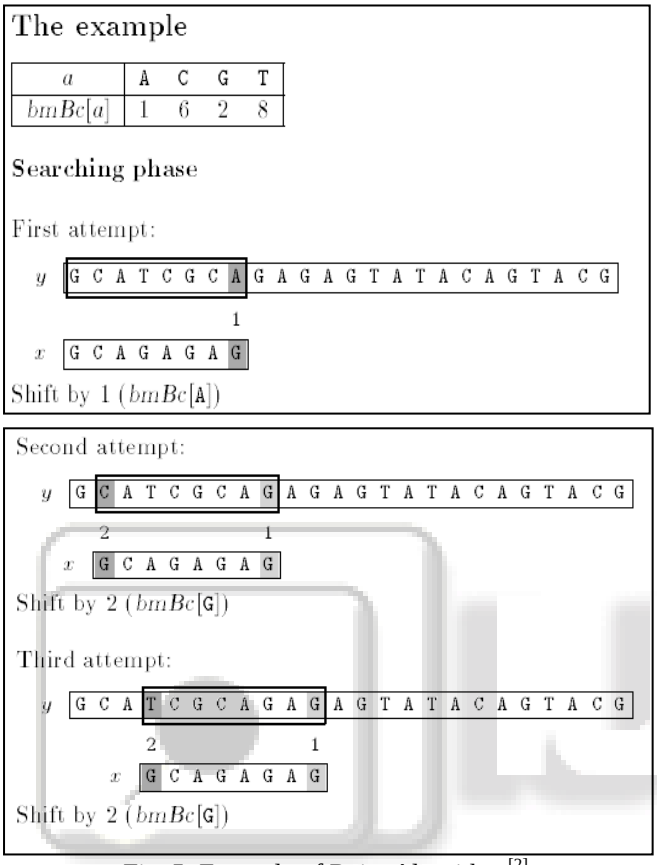

Fig. 5: Example of Raita Algorithm [2]

G. Quick Search Algorithm

From the study of [2], Quick Search algorithm (QS) is simplified version of Boyer-Moore algorithm and only uses the bad-character shift. This algorithm matches the pattern with text window in any order character by character. Its best performance in short pattern and large alphabets.

Its pre-processing time complexity is O (m+|Σ|) and space complexity is O (Σ|).Searching phase complexity is O (mn).

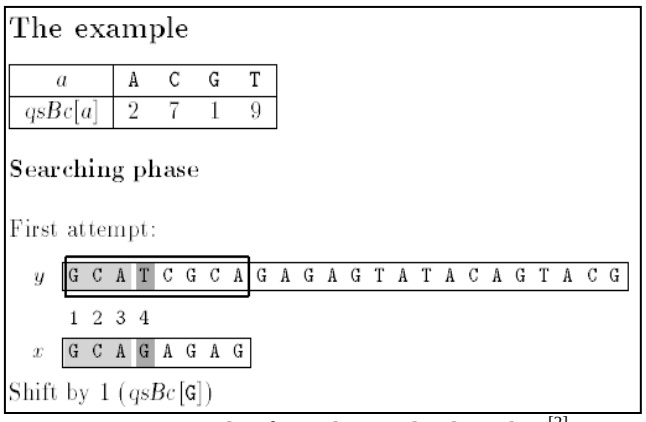

Fig. 6: Example of Quick Search Algorithm [2]

IV. COMPARSION

Algorithm Name	Comparison Order	Pre Processing complexity	Searching complexity
BF	Not Relevant	NO	O(mn)
MP	Left to Right	O(m)	O(m+n)
KMP	Left to Right	O(m)	O(m+n)
BM	Right to Left	O(m+| Σ |)	O(mn)
BMH	Right to Left	O(m+| Σ |)	O(mn)
Raita	Specific Order	O(m+| Σ |)	O(mn)
Quick Search	Any Order	O(m+| Σ |)	O(mn)

Table. 1: Algorithm Analysis Table

V. CONCLUSION

In this survey paper various Exact Pattern Matching Algorithm has been described with their time and space complexity. Comparative study of various algorithm result in applicability of algorithm based on the requirement of given problem, so rather than applying each Pattern Matching algorithm in every problem, we choose optimal algorithm for that problem. Optimal Pattern Matching algorithm in terms of faster execution and less extra space required by the algorithm.

ACKNOWLEDGMENT

I would like to express my heartfelt thanks to my teachers for providing me with much needed support and guidance. I am also very thankful to my all classmate for providing me precious help and advice continuously and would also like to thank my parents and my husband for their support. Finally would thank all my well-wishers who have played a very important role in nurturing and strengthening me.

REFERENCES

[1] Cormen, T.H., Leiserson, C.E., Rivest, R.L., "Introduction to Algorithms", Chapter 34, MIT Press, 1990.
[2] Christian Charras, Thierry Lecroq, "Handbook of Exact String-Matching Algorithm".
[3] Nimisha Singla, Deepak Garg" String Matching Algorithms and their Applicability in various Application", IJSCE, Vol.1, Issue-6, Jan-2012, pp.218-222.
[4] Jonathan Leidig and Christian Trefftz, "A Comparison of the Performance of four Exact String Matching Algorithms", IEEE EIT, May-2007, pp.333-336.
[5] Vidya SaiKrishna, Prof. Akhtar Rasool and Dr. Nilay Khare, "String Matching and its Applications in Diversified Fields", IJCSI, Vol. 9, Issue 1, No 1, January 2012,pp.219-226.
[6] Kamal M. H. Alhendawi, Ahmad Suhaimi Baharudin, "String Matching Algorithms (SMAs): Survey & Empirical analysis", International Journal of Computer Science and Management Research, Vol 2 Issue 5, May 2013 , pp.2637-2644.

[7] Knuth, D., Morris, J. H., Pratt, V., "Fast pattern matching in strings", SIAM Journal on Computing, Vol. 6, No. 2, DOI: 10.1137/0206024, 1977, pp.323–350.
[8] R.S. Boyer, J.S. Moore, "A fast string searching algorithm," Communication of the ACM, Vol. 20, No. 10, 1977, pp.762–772.
[9] Sunday, D.M., "A very fast substring search algorithm", Communications of the ACM, Vol. 33, No. 8, 1990, pp. 132-142.
[10] R. N. Horspool, "Practical fast searching in strings", Software—Practice and Experience, Vol. 10, No. 3, 1980, 501–506.
[11] TIMO RAITA, "Tuning the Boyer–Moore–Horspool String Searching Algorithm", SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 22(10), OCT-1992, pp.879–884.

