Bidirectional Exact Pattern Matching Algorithm

Iftikhar Hussain, Muhammad Zubair, Jamil Ahmed and Junaid Zaffar

Abstract - **In this research we introduce a new idea to solve the problem of exact pattern matching algorithm by using two pointers, simultaneously.**

Keywords - **Algorithm, exact pattern matching, searching, Bidirectional.**

I. INTRODUCTION

The aim of exact pattern matching algorithm is to find one or all occurrences of the pattern within a larger group of text string [3]. In this paper, we proposed a new Bidirectional exact pattern matching algorithm based on window sliding method of exact string matching problem which will be helpful in various needs of pattern matching.

According to literature survey, all the authors focus to reduce the **number of character comparisons** and **processing time** in both worst/average cases.

In this paper we compare Bidirectional algorithm's results with Boyer-Moore, BM Horspool, Quick Search, and Turbo BM algorithms which considered efficient in terms of number of character comparisons and attempts take to complete the processing of selected text.

II. BIDIRECTIONAL ALGORITHM

Bidirectional exact pattern matching algorithm compares a given pattern from both sides, starts from right then from left, one character at a time within the text window. In case of a mismatch or a complete match of the pattern, scan for the mismatched and right most characters of the text window to the left of the related text characters in pattern. When rightmost and mismatched characters found in the left of related characters of text string in pattern on same distance, then align pattern with that text window. A complete match will be found when the left pointer cross the right pointer at the middle of the pattern.

Case 1: If mismatch cause by right pointer at most right position or by left pointer at most left position of pattern then scan for the rightmost character T(i+j-1) of the text window in the pattern from right to left. If character P(j′) found in the pattern then align character P(j′) with T(i+j-1).

Case 2: If mismatch cause by right pointer at any position except most right position of the pattern then scan P(j-1…1) for character P(j″) which is same as T(i+j-1) and P(j′) same to T(i+m-1). If characters found in pattern on equal shifts then align character P(j″) with T(i+j-1) and P(j′) with T(i+m-1).

Case 3: If mismatch cause by left pointer at any position except the most left position of pattern then scan P(j-1…1) for character P(j″) which is same as T(i+j-1) and P(j′) same to T(i+m-1). If characters found in pattern on equal shifts then align character P(j″) with T(i+j-1) and P(j′) with T(i+m-1).

Iftikhar Hussain, MSCS, Reg. 01713, IQRA University Islamabad. Plot No. 5, Khayaban-e-Johar, Sector H-9, Islamabad, Pakistan. E-mail: iftikhar.iftikhar786@gmail.com

Case 4: If equal shifts of mismatched (either cause by right or left pointer) and right most characters are not found, and P(j′)= T(m-1) is found at the left of mismatched character P(j) of the pattern, then align P(j′) with T(m-1).

Case 5: If equal shifts of mismatched (either cause by right or left pointer) and the most right characters did not found at the left of mismatched character P(j) of the pattern then whole pattern will be shifted.

III. ANALYSIS OF BIDIRECTIONAL ALGORITHM

The worst case time complexity of preprocessing phase of Bidirectional algorithm is *O(m)*, because only one loop is used to scan the pattern to find the characters.

The total time complexity of searching phase is *O(mn/2)*, because *O(n)* takes to shift pattern to right of the text and *O(m/2)* to search pattern in text string.

Bidirectional algorithm requires *O(m)* extra memory space in worst case in addition with the text and pattern string.

IV. RESULTS AND DISCUSSIONS

In graphical form, Fig. 1 shows total number of shifts taken by each algorithm using different pattern size. Results show that in short pattern number of shifts is closer to other algorithms but when pattern length is increased Bidirectional algorithm become more efficient.

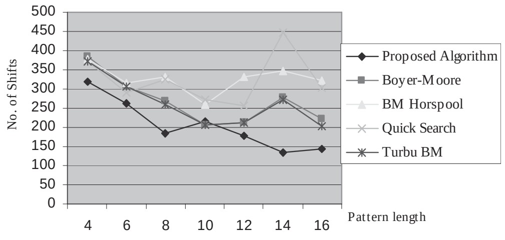

Fig. 1: Shift Base Comparison

Similarly, proposed algorithm compares more characters when pattern length is short. On long patterns, it takes less character compares than other algorithms.

V. CONCLUSION

This research presents a new idea of comparing the pattern with text string from both sides of the pattern simultaneously. Asymptotic analysis shows that Bidirectional algorithm is better than exiting algorithms, as it takes *O(mn/2)* time in searching phase.

REFERENCES

[1] R. S. Boyer, J. S. Moore, "A fast string searching algorithm," *Communication of the ACM*, Vol. 20, No. 10, 1977, pp.762–772.
[2] 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.
[3] Cormen, T.H., Leiserson, C.E., Rivest, R.L., *Introduction to Algorithms*, Chapter 34, MIT Press, 1990, pp 853-885.