MANSOURA UNIVERSITY
FACULTY OF COMPUTERS AND INFORMATION

IMPROVED APPROACH FOR EXACT PATTERN MATCHING

(BIDIRECTIONAL EXACT PATTERN MATCHING)

Outlines

- Abstract & Introduction
- Literature Review
- Bidirectional Exact Pattern Matching cases

- ## Abstract & Introduction

  - In this paper we will talk about present Bidirectional exact pattern matching algorithm .
  - This algorithm focus to reduce the number of character comparisons and processing time.
  - Bidirectional (BD) exact pattern matching (EPM) introduced anew idea to compare pattern with select text by using two pointer (right and lift) simultaneously .
  - **BD** solve the problem of time complexity which take **O(m)** to find the pattern in Text and searching phase takes **O(MN/2)** SO **BD** effective than the number of existing algorithms in many cases .
  - The purpose of **BD** string matching algorithm is to find all occurrences of pattern in the text string and we will compare between **BD** and other algorithm .

- ## Literature Review

  - Naive algorithm this algorithm compares a given pattern with all substring of the text in case of mismatch make shift by one position until find all pattern in text the time complexity of this algorithm O(mn) .

  - Knuth-Morris-Pratt (KMP) algorithm is proposed in 1977 to speed up the procedure of exact pattern matching by improving the lengths of the shifts . we compares the characters from left to right of pattern . The time complexity of preprocessing phase is O(m) and of searching phase is O(nm).

  - Boyer-Moore Horspool (BMH) did not use the shifting as BoyerMoore algorithm used. it used only the occurrence to maximizethe length of the shifts . time complexity is O(mn) .

- ## Bidirectional Exact Pattern Matching cases

  ➢ ### Working of Bidirectional EPM :

  Bidirectional EPM algorithm has number of cases to shift the pattern maximum to right of text window. Suppose T(1…n) is the text string and P(1…m) is the pattern and we compare P(1…m) with T(i…i+m-1) from both sides of the pattern, one character at a time, start from right side of the pattern.

  - **Case 1:** If mismatch cause by right pointer at most right position T(i+j-1) ≠ P(j) or by left pointer at most left position T(i) ≠ P(1) of the pattern here j = m then scan P(j-1…1) for character P(j′) which is equal to T(i+j-1). If character found in the pattern then align character P(j′)=P(j-1…1) with T(i+j-1) as in Figure 2.

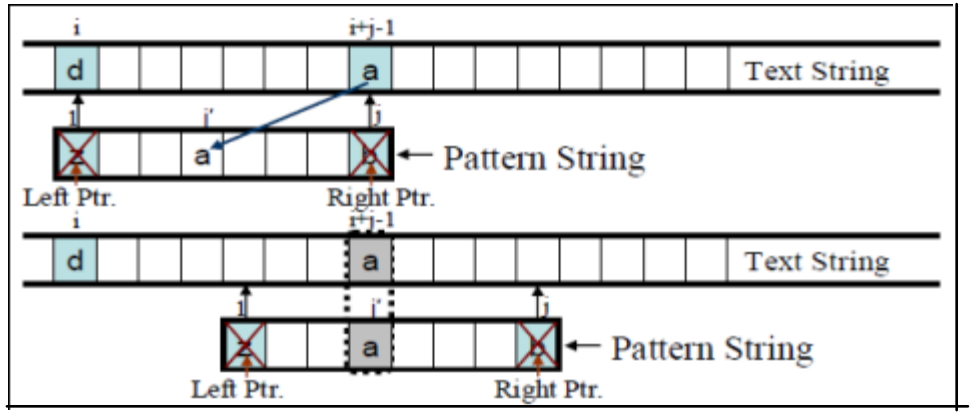


Figure 2: When right most or left most character mismatched.

- **Case 2:** If mismatch cause by right pointer at position T(i+j-1) ≠ P(j) where 1≤j≤m and it is not the right most character of the pattern then scan P(j-1…1) for character P(j″) which is equal to T(i+j-1). And also scan P(m-1…1) for the character P(j′) which is equal to T(i+m-1). If characters found in the pattern then align character P(j″)=P(j-1…1) with T(i+j-1) and P(j′)=P(m-1…1) with T(i+m-1), if shift‟s length of both characters are equal as shown in Figure 3.

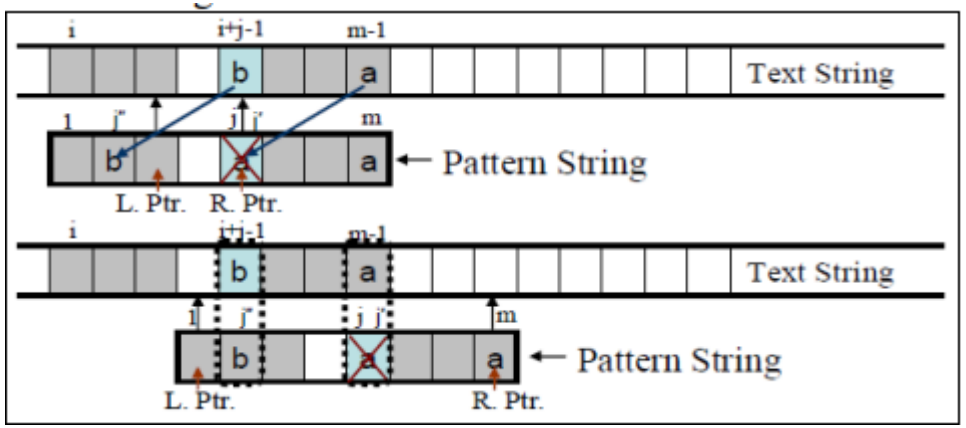


Figure 3: Mismatch by right pointer other than rightmost character.

- **Case 3:** If mismatch cause by left pointer at position T(i+j-1) ≠ P(j) where 1≤j≤m and it is not the left most character of the pattern then scan P(j-1…1) for character P(j″) which is equal to T(i+j-1). And also scan P(m-1…1) for the character P(j′) which is equal to T(i+m-1). If characters found in the pattern then align character P(j″)=P(j-1…1) with T(i+j-1) and P(j′)=P(m-1…1) with T(i+m-1) if shifts of both characters are at equal length as shown in Figure 4.

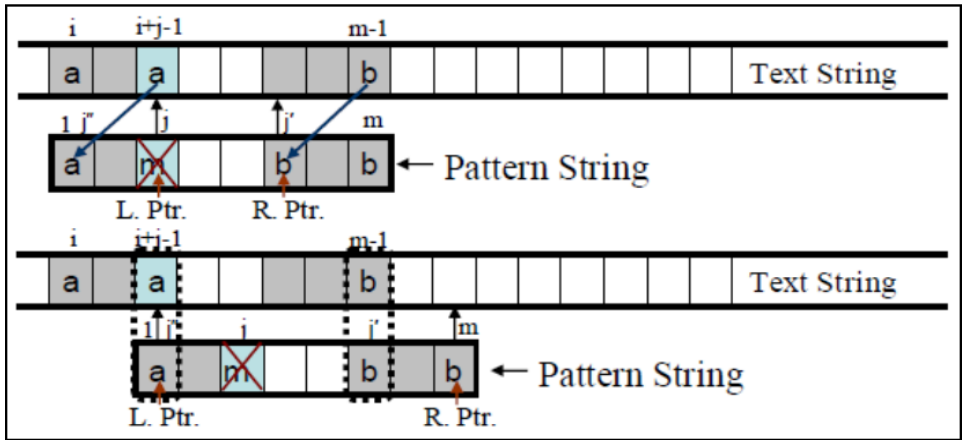


Figure 4: Mismatch by left pointer other than left most character.

- **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 right of mismatched character P(j) of the pattern, then align P(j′) with T(m-1) as in Figure 5.

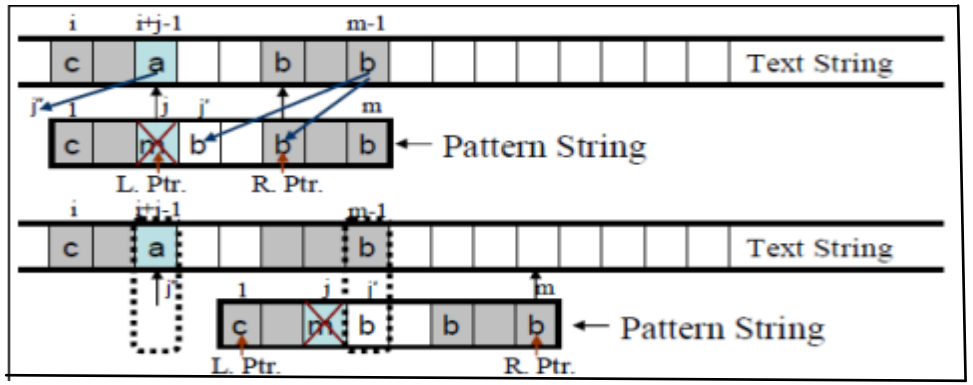


Figure 5: Mismatched character did not find in pattern.

- **Case 5:** If equal shifts of mismatched (either cause by right or left pointer) and right most characters, did not find at the left of mismatched character P(j) of the pattern then align left most character of pattern with T(i+m) as shown in Figure 6.

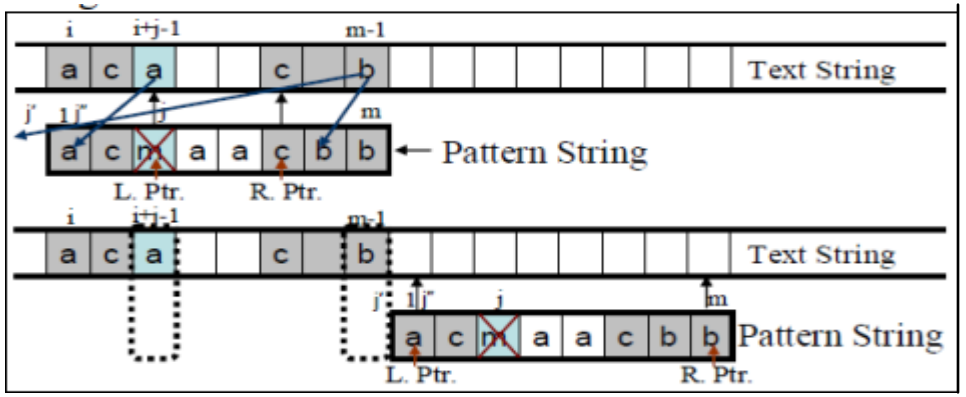
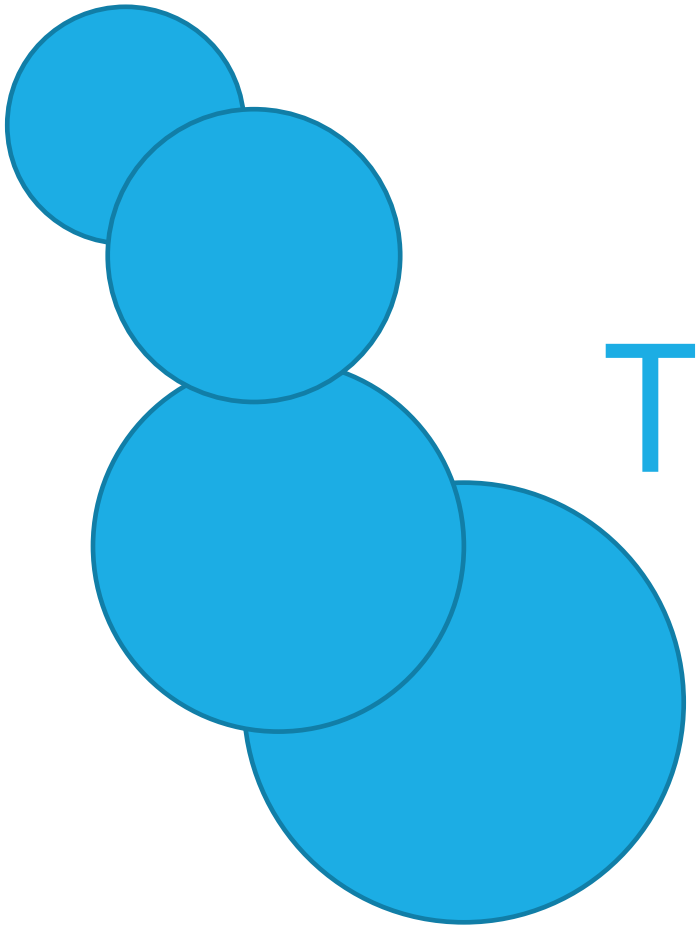


Figure 6: Maximum shift; if rightmost character did not find.



Thank You