RESEARCH OF PATTERN MATCHING ALGORITHM BASED ON KMP AND BMHS2

Ever used a text editor and you had to find a word? I'm sure this is a common scenario and it has happened to you many times. Of course, your first instinct isn't to scan the entire article, you make the computer do that for you by using the find function. But have you ever wondered how that works? What makes the find function so fast? How does it find the matches? Here's where "string matching" or "pattern matching" algorithms are used.
What do we mean by string matching? We have two strings, a-the pattern and b - the string in which we want to find the pattern, and we need to find whether a matches some or the complete string b. Essentially, we need to see if the string a is a part of string b.

AUTHORS
Yansen Zhou , Ruixuan Pang

AFFILIATIONS
Department of Information Science and Technology University of International Relations Beijing, China

RESEARCH PAPER INFO
2019 IEEE 5th International Conference on Computer and Communications

RESEARCH PAPER LINK
https://ieeexplore.ieee.org/document/9064076

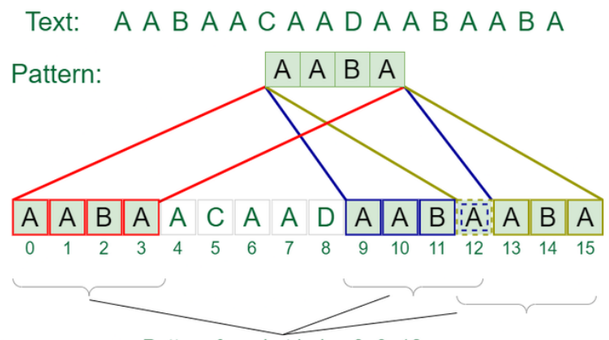
INTRODUCTION

- Pattern matching algorithm is widely used in search engine and text matching and has become a research hotspot. The room for performance improvement of a single pattern matching algorithm is limited.
- The two algorithms have different matching sequences in each match, but they need to move the text string matching window to the right in case of mismatch
- The current research focuses on combining multiple pattern matching algorithms and combining the advantages of each algorithm to further improve the time performance of pattern matching.
- The improvement of performance of pattern matching algorithm mainly lies in reducing the number of character comparisons and increasing the distance of the matching window moving to the right when matching.

IDEA OF THE ALGORITHM

- . This algorithm mainly eliminates the backtracking problem of the main string pointer in the matching process of BF algorithm, and makes use of the partial matching results to slide the pattern string to the right as far as possible and then continue to compare, so as to improve the efficiency of the algorithm to a certain extent. In the worst matching case, the time complexity of KMP algorithm is O(m+n), where the length of the pattern string is m and the length of the main string is n.
- The basic idea: whenever a match fails, the i pointer does not have to backtrack, but to use the "partial match" result which has been obtained to see if it is necessary to adjust the value of I, and then "slide" the pattern to the right several positions before continuing the comparison.

VISUALIZATIONS



1. Compute the partial match table pi for the pattern.
2. Compute the suffix table and prefix table for the pattern using the compute_suffix_table and compute_prefix_table functions, respectively.
3. Initialize i and j to 0, representing the current indices of the text and pattern strings, respectively.
4. While **i** is less than or equal to **n - m**, where **n** is the length of the text and **m** is the length of the pattern, do the following:
   a. While **j** is less than **m** and the **j**-th character of the pattern matches the **i+j**-th character of t he text, increment j.
   b. If **j** is equal to **m**, a match has been found, so record the starting index of the match and set j to the value of the first entry in the suffix table.
   c. Otherwise, set **j** to the maximum of the following values:
   d. Increment i by the maximum of j - pi[j-1] - 1 and 1.
5. Return the list of starting indices of all matches found.

IMPORTANT RESULTS

CASE 1: MATCHING TIME WHEN THE NUMBER OF PATTERN STRINGS VARIES (MS)

Algorithm	Number of Pattern Strings							
	100	200	300	400	500	600	700	800
I_KMP	316	303	301	302	304	299	307	302
BMHS2	302	291	289	289	296	289	299	296
I_KMP_BMHS2	248	246	242	250	243	246	251	248

CASE 2: . MATCHING TIME WHEN PATTERN STRING LENGTH VARIES (MS)

Algorithm	Length of Pattern String							
	30	40	50	60	70	80	90	100
I_KMP	281	253	232	223	201	176	150	145
BMHS2	268	242	221	212	189	163	139	132
I_KMP_BMHS2	236	215	206	172	149	132	112	104

CONCLUSION

- The performance of the improved KMP algorithm combined with BMHS2 algorithm can be improved, mainly lying in the matching window of text string is moved to the right by using the larger jump distance between KMP and BMHS2 in each mismatch.
- Compared with the multi-pattern matching algorithm, An improved algorithm combining KMP with BMHS2 algorithm still has the problem of low efficiency.

APPLICATIONS OF KMP

1. Text editors and word processors: The KMP-BMHS2 algorithm can be used in text editors and word processors to find and replace words and phrases in a document.
2. Information retrieval systems: The algorithm can be used in information retrieval systems to search for keywords in large databases of text.
3. DNA sequencing: The KMP-BMHS2 algorithm can be used in DNA sequencing to search for patterns in genetic sequences.

MADE BY
Aarya Tiwari

ROLL NUMBER
16010421119

BATCH
B2

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering