Journal of Applied Sciences, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.

A New Exact Pattern Matching Algorithm (WEMA)

Abdallah A. Hlayel and Adnan A. Hnaif
Department of Computer Information System, Faculty of Science and Information Technology,
Al-Zaytoonah University of Jordan, P.O. Box 130, Amman, 11733, Jordan

Abstract: In pattern matching or recognition the match usually has to be exact. This study, introduce a new general approach algorithm, called the Weighted Exact Matching Algorithm (WEMA). The WEMA applied to match the exact pattern within a text depending on preparing the text in an index matrix weight with the locations of characters in alphabetical order to perform direct access matching. Whereby, WEMA check the possibility of the pattern existence before applying the matching process. The simulation result showed significant improvement in the exact string matching and therefore extreme competence in the real applications.

Key words: Pattern, exact string matching algorithms, quick search, weighted exact matching algorithm, WEMA

INTRODUCTION

Pattern matching algorithms are used to retrieve patterns from sets based on a search criterion. In computer science applications, the matching algorithms are the most used in different fields such as network security, artificial intelligence, data mining and others. Matching algorithm can be utilized for various forms of digital data including images (Abed and Zaoui, 2011; Chalabi *et al*., 2008) audio and video (Bulbul, 2007). String matching algorithm is the process to find the occurrence of pattern “P” into a text “T”, where “T” is longer than “P” (Alhaj *et al*., 2010). String matching algorithms can be classified into two techniques: Exact and approximate string matching algorithms. Exact-string-matching algorithms can be used to find the occurrence of “P” in “T” while approximate string matching algorithms are concerned with the similarity percentage between the “P” and the “T”. Moreover, there are many good works to enhance the performance of exact or approximate string matching algorithms Sleit *et al*. (2007) and Mansi and Alnihoud (2010) and also some works in parallel approach to enhance the performance (Hudaib *et al*., 2008; Raju and Babu, 2007).

This study introduces an algorithm called WEMA which can be used to find exact matching between two strings, so that the proposed algorithm consists of two steps: First, create an alphabetical matrix called index matrix weight, in order to arrange all the characters positions of the text on it. Second, run the matching process to find the exact occurrence between the pattern “P” and the text “T”.

MATERIALS AND METHODS

This section indicates some of the most famous algorithms that are used in exact string matching algorithms.

Boyer-moore algorithm: Boyer-Moore algorithm is considered as one of the most famous pattern matching algorithms, one that is considered very fast in practice and it was designed for the exact string matching of many strings against a single keyword (http://www-igm.univ-mlv.fr/~lecroq/string/). The first heuristic phrase used is “bad character shift”. Bad character shift starts a comparison from the right to the left and if a character is seen that does not exist in the text to search for, then the search algorithm can be shift forward to an “M” character where “M” is the length of the pattern. The second heuristic phrase used in the Boyer-Moore algorithm is “good suffix shift”. Good suffix shift starts a comparison from the right to the left and if it is matches, then the algorithm check the next character in the text with the next character in the pattern, until matching all the strings. In the case of mismatching, the Boyer-Moore algorithm is looking for the next occurrence of a substring that was matched before.

Quick search algorithm: The Quick Search algorithm is more simplified version of Boyer-Moore algorithm but the Quick Search algorithm used only the “bad character shift”. The Quick Search algorithm work like a Horspool algorithm as well by working on one of two shifts of

Corresponding Author: Abdallah A. Hlayel, Department of Computer Information System,
Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan,
P.O. Box 130, Amman, 11733, Jordan

pattern. The Quick Search algorithm is easy to implement and is very fast in practice for short and large patterns (Charras and Lecroq, 2004).

PROPOSED ALGORITHM (WEMA)

In order to evaluate the matching process between two strings to match the pattern “P” in the text “T”, WEMA has two stages:

Preparing stage: This stage will produce an alphabetical index matrix weight “M” and filling the character positions of text “T” in the corresponding characters “M”. Each character in “M” has weight that determine the number of occurrences each character in the text (the number of indices for each character).

Matching stage: The WEMA can be used to find the exact matching between the pattern “P” and the text “T”. Consequently, the matching stage works as follows steps:

Step 1: Create array list “L” for the pattern “P”

Step 2: Look up the corresponding distinct characters of the pattern “P” in “M” and then find the minimum character weight. If the minimum character weight is equal to zero then exit the matching process, because there is no exact matching. Otherwise, chose the character with the minimum weight (if there are more than one character having the same minimum weights, choose any one of them)

Step 3: Create the first attempt of the array list “L” and add to it the first index value of the minimum weight character that obtained in step 2 under the first matching corresponding character position L (i) of the array list “L”

Step 4: Read the next and previous characters of the pattern character position L(i) that are L (i+1) and L(i-1), then matching with the characters of “M” and fill to the current attempt their index values that equal to the index value of the current character position L(i)+1 for the next and equal to the index value of the current character position L(i)-1 for the previous. If both exist, then continue repeating this step for the next positions in both directions until reaching the end of the pattern “p” and getting the exact matching

Step 5: Otherwise, create the next attempt with reading the next index of the chosen minimum weight character in step “2" and fill it under the first matching corresponding character position L(i) of the array list “L”, then repeat step “4” until get the exact matching, or until reach the last index of the minimum weight character

RESULTS

This section introduces a business problem to clarify the WEMA implementation and show the obtained simulation results.

Problem: Consider the problem to match exact pattern P = “gcagcgag” in the text T = “gcatcgcagagagtatacagtacg”.

Solution stages

Preparing stage: Applying the preparing stage to create the index matrix weight “M” as shown by Fig. 1.

This stage runs only once, as long as no updates are available in the text “T”, in case to match more than one pattern. Also, by using the WEMA, the possibility of the pattern existence before searching can be checked and directly matching the corresponding characters with minimum check operations through using the minimum weight.

Matching stage: Regarding to the matching stage, Fig. 2 depicts “step 1” of the proposed matching algorithm which describe the process of reading the pattern “P” and creating the array list “L”.

Alphabetical	Indices of the text characters							
characters	1	2	3	4	5	6	7	8
a(8)	3	8	10	12	15	17	19	22
b(0)								
c(5)	2	5	7	18	23			
.								
g(7)	1	6	9	11	13	20	24	
.	.	.	.	.	.	.	.	
t(4)	4	14	16	21				
.	.	.	.	.	.	.	.	
z(0)								

Fig. 1: Index matrix weight “M” for “T” that is created by applying the preparing stage

	Pattern P = “gcagcgag”							
Position	1	2	3	4	5	6	7	8
Array list “L”	g	c	a	g	c	g	a	g

Fig. 2: Array list “L” of pattern “P” that is created by applying step1

	Pattern P = "gcagcgag"							
Position	1	2	3	4	5	6	7	8
Array list "L"	g	c	a	g	c	g	a	g
First attempt		2						

Fig. 3: First attempt of array list "L" that is obtained by applying step 3 (filling the first index of the minimum weight character "c" to the first corresponding position)

	Pattern P = "gcagagag"							
Position	1	2	3	4	5	6	7	8
Array list "L"	g	c	a	g	a	g	a	g
First attempt	1	2	3					

Fig. 4: Filling the next and previous indices after applying step 4

By matching stage "step 2", the distinct characters are "g, c, a" and by reference to the index matrix weight "M", the minimum weight is greater than 0 and character "c(5)" have the minimum weight with the following indices {2, 5, 7, 18, 20}.

Regarding to the matching stage "step 3", create the first attempt and add to it the first index of character "c" under the first corresponding position of the array list "L" (Fig. 3).

After that, "step 4" is applied to read the next and previous characters of the pattern character position L(i) that are L(i+1) and L(i-1), then matching with the characters of "M" and take their index values that equal to the index value of the current character L(i)+1 for the next and equal to the index value of the current character L(i)-1 for the previous which is "a" and "g". By reference to the index matrix weight "M", where the character "a" has the indices {3, 8, 10, 12, 15, 22} and "g" has the indices {1, 6, 9, 11, 13, 24}. Select and add to the first attempt the index "3" because it is equal to the current index value +1 and the index "1" because it is equal to the current index value -1 (Fig. 4).

Forwarding-up the WEMA matching stage, read the next character position of the pattern L(i+2) which is "g", where "g" has the following indices {1, 6, 9, 11, 13, 24}, so, there is no any index equal to the current index L(i+2)+1 as demonstrate in Fig. 5 and no need to continue this first attempt.

	Pattern P = "gcagagag"							
Position	1	2	3	4	5	6	7	8
Array list "L"	g	c	a	g	a	g	a	g
First attempt	1	2	3	-				

Fig. 5: Result for a case where matching is not achieved at the end of step 4 ("g" has no index equal to the current index L(i+2)+1)

	Pattern P = "gcagagag"							
Position	1	2	3	4	5	6	7	8
Array list "L"	g	c	a	g	a	g	a	g
First attempt	1	2	3	-	-	-	-	-
Second attempt	-	5	-	-	-	-	-	-
Third attempt	6	7	8	9	10	11	12	13

Fig. 6: Exact matching result after applying matching stage (three attempt are required for this example) all possible attempts for the given problem until get the exact matching

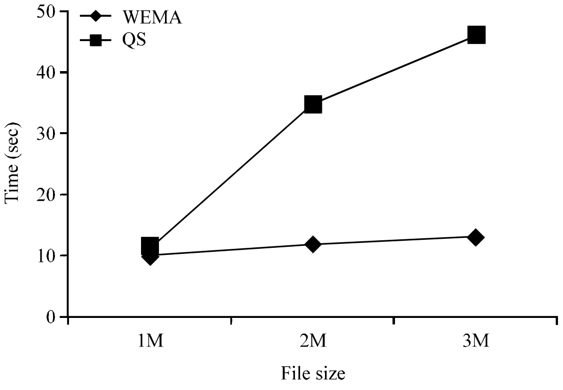

Fig. 7: Comparison execution time between WEMA and QS

Finally, and according to "step 5", create the next attempt with reading the next index of the chosen minimum weight character in "step 2" and repeat from "step 4', until to get an exact matching or until reach the minimum character weight last index. Figure 6 shows the final result of all matching attempts, where the third attempt got an exact matching with the pattern "P".

Simulation results: The simulation have built to demonstrate and to compare the performance of the WEMA with the exact string matching algorithm, the Quick Search Algorithm (QS). Our simulation runs under the laptop computer which having Intel® Core™2 Duo CPU 2900 4M, 4G DDR3 ram and windows XP. The results showed high performance of the WEMA over the QS algorithm. Three different experiments have carried out to search for the pattern p = "gcagagag" in a 3 text file size of 1, 2 and 3 M respectively. The result showed significant improvement in the executing time, wherein applying the WEMA decreased the executing time (Fig. 7).

DISCUSSION

In all exact matching techniques, the performance, accuracy and time complexity plays major rules for developing new algorithms of finding the exact patterns. As mentioned above, there are many good works to enhance the performance of the exact pattern matching algorithms. Whereby, the QS algorithm is a simplified version of Boyer-Moore algorithm which considered one of the fast and accurate algorithms in this field. In fact, the QS algorithm and all previous studies start the matching process directly, meanwhile the WEMA check the possibility of the pattern existence and then start the matching process, this will lead to eliminate unnecessary matching process. In addition, WEMA is faster than QS based on execution time which required finding the pattern “P” in the text “T”, as shown by the obtained results. Moreover, WEMA have roughly an equal execution time; regardless of the text file size, because of its direct access matching and not depends on the position of the pattern inside the text, in the beginning of the text as best case or in the end of the text as worst case. WEMA have O(n) time complexity for the algorithm, but it have only O(1) for the matching stage while QS have O(n) time complexity for the algorithm and O(n) for the matching stage.

Finally, the WEMA preparing stage for creating the index matrix weight “M” is a very important step towards speeding up the matching process, because of the following points:

- The index matrix weight “M” runs only once, as long as no updates are available in the text “T”, to match more than one pattern
- From the character weight, WEMA check the possibility of the pattern existence before applying the matching process
- From the character weight, WEMA perform fewer operations in the matching process
- WEMA apply direct access to the character position without having to search sequentially in the text

CONCLUSION

In this study, the proposed algorithm WEMA have been done, in order to improve the performance and accuracy of the exact string matching algorithms. QS Algorithm has applied as one of the best exact string matching algorithms. The result showed high performance for WEMA over QS algorithm.

ACKNOWLEDGMENTS

We would like to thank Al-Zaytoonah University of Jordan, Faculty of Science and Information Technology for its support that enabled us to complete this study.

REFERENCES

Abed, H. and L. Zaoui, 2011. Partitioning an image database by K_means algorithm. J. Applied Sci., 11: 16-25.

Alhaj, M.M.A., M. Halaiyqah, M.A.A. Hashem, A.A. Hnaif, O. Abouabdalla and A.M. Manasrah, 2010. An innovative platform to improve the performance of exact string matching algorithms. Int. J. Comput. Sci. Inform. Sec., 7: 280-283.

Bulbul, H.I., 2007. Application of bernstein and pattern recognition methods for speech command recognition. J. Applied Sci., 7: 3063-3068.

Chalabi, Z., N. Berrached, N. Kharchouche, Y. Ghellemallah, M. Mansour and H. Mouhadjer, 2008. Classification of the medical images by the kohonen network SOM and LVQ. J. Applied Sci., 8: 1149-1158.

Charras, C. and T. Lecroq, 2004. Handbook of Exact String Matching Algorithm. King's College Publications, London, UK., ISBN-13: 9780954300647, Pages: 238.

Hudaib, A., R. Al-Khalid, D. Suleiman, M. Itri and A. Al-Anani, 2008. A fast pattern matching algorithm with Two Sliding Windows (TSW). J. Comput. Sci., 4: 393-401.

Mansi, R.H. and J.Q. Alnihoud, 2010. An efficient ASCII-based algorithm for single pattern matching. Inform. Technol. J., 9: 453-459.

Raju, S.V. and A.V. Babu, 2007. Parallel algorithms for string matching problem on single and two dimensional reconfigurable pipelined bus systems. J. Comput. Sci., 3: 754-759.

Sleit, A., W. AlMobaideen, A.H. Baarah and A.H. Abusitta, 2007. An efficient pattern matching algorithm. J. Applied Sci., 7: 2691-2695.