An Efficient Word Matching Algorithm For off Line Text

Sattyam Kishor Mishr[1], Manish Pande[2]

GJCST Classification (FOR)
F.2.2 , E.2, K.8.1

***Abstract*-Word processing application that are used today perform a wide variety of jobs; the most challenging of them is to search for a given sequence of characters of a word is called string matching. To find all the appearances of the pattern P in the text T, word matching problem is used. Pattern P neither encloses white space nor anticipates and pursue by space. It is predicated that our text is offline. For solving the word matching problem, word searching algorithm (WSA) has been propounded. WSA works by splitting the offline text in to number of tables and search the pattern by using the brute force manner. The main drawback of WSA algorithm is, to search each appearances of the pattern by the brute force manner in each table. Due to this WSA increase the number of comparisons of the word. This paper proposed an algorithm, that is developed to reduce the number of comparisons to search the pattern in the offline text.**

Keywords- Algorithm; string matching; hashing; offline searching; word searching

I. Introduction

In all string matching problems, all the appearances of the pattern P in the text T are to be reported. Many algorithms have been proposed for solving the word matching problem. To find all the appearances of the pattern P in the text T, word matching problem is used. Pattern P neither encloses white space nor anticipated and pursue by space. Ibrahiem et al in 2008[1] has propounded an algorithm called word searching algorithm (WSA) for solving the word matching problem by splitting the offline text in to number of tables in the pre-processing phase. In the searching phase, they perform character to character comparisons with of the same length in the table by brute force manner.In existing work one more algorithm come for solving the word matching problem called modified word searching algorithm (MWSA) [2]. In this algorithm they use the efficient hash function called SDBM hash function [3, 4] for finding the hash value of each word in the text and the hash value of the pattern P. In WSA has main problem with the searching phase, because the searching phase encloses the brute force manner [1]. In brute force manner they perform character to character comparisons with the same insearching phase. We use SDBM hash function due to very length. In MWSA, they use the SDBM hash function which reduces the time taken to search the word. SDBM hash function has very less chance of collision even in a very large text. [2, 3, 4] In this paper, the modification in WSA algorithm is done by using the SDBM hash function in pre- processing phase and balanced binary search tre less chance of collision even in a very large text. Proposed algorithm has two phases that works as follows: In pre-processing phase: whole text T is read and split into nearly equal sizes. For every part we create a table. Table enclosesstarting position of each word in first column and hash value (computed by the SDBM hash function) of every word in second column for each part of the text. For each table balanced binary search tree is created, which include the hash value of every words of each part in its nodes. The pre-processing phase is done only once. In searching phase: the hash value H of pattern P is computed using the same hash function. This hash value H is searched in the balanced binary search tree. If a match is found for a hash value, then the appearances of the pattern P is verified The main drawback of WSA algorithm is that after splitting the text into number of tables it searches the pattern in the row corresponding to pattern length by brute force manner [1, 2]. Drawback of WSA algorithm is eliminated by proposed algorithm by using the SDBM function [2] and balanced binary search tree [5, 6]. In proposed algorithm, only one table in pre-processing phase and use balanced binary search tree in searching phase. Our algorithm has taken less number of comparisons than the WSA algorithm to search the pattern in the offline text.This paper is organized as follows: In section 2, it is explained existing word matching algorithms. In section 3 it is explained our propose algorithm in section 4, simulation and comparative results are explained. Finally we conclude in section 5.

About[1]. Computer Science and Engineering Department Maulana Azad National Institute of TechnologyBhopal, India satyam_satyam@live.com
About[2]. Computer Science and Engineering Department Maulana Azad National Institute of TechnologyBhopal, India manishpandey@manit.ac.in

II. Existing Word Matching Algorithms

1) *Word Searching Algorithm (WSA)*

In this section we explain the recent optimal word searching algorithms (WSA) and modified word searching algorithm (MWSA) [1, 2]. In WSA algorithm, the given text is predicated to be offline. This algorithm has two phases that works as follows: The first phase which is pre-processing phase starts with reading and splitting the text T into k number of equal parts depending on size of the text T and constructing k number of tables with two columns for each part of the text. The first column contains length of the word and the second contain the starting position of each word in the text. The start positions of the words will be in the same row for the same length. Once the table is constructed, it is sorted in ascending order using the length of the words as a key for sorting. This phase is done once. The second phase is searching for a specific pattern. The algorithm calculates the length of the pattern and search for the same length in

the tables starting from the first table. If the length does not exist in the first table then the algorithm searches for the word in the next table and so on. If the current table is the last table, then a message will indicate that the pattern does not exist. On the other side if the length finds in the table, then the algorithm will report the words in the text using the stored start position in the table and begin to compare. If a full match occurred then occurrence of the pattern is reported. But if word does not find the same character, then the algorithm will move to the next start position and compare again. [1]

2) Modified Word Searching Algorithm (MWSA)

The main drawback of WSA algorithm was that it searches each appearances of the pattern by the brute force manner in each table. To improve WSA algorithm, MWSA algorithm was proposed. In MWSA, the whole text T is splits in to k equal parts according to size of the text T. For each part two tables A and B are created. Tables A and B have two column in each. In pre-processing phase: each part of the text is read. While reading, lengths of the words are computed and according to length of the words, starting positions of the words are stored in the table A and each starting position in table A is mapped to a corresponding hash value in table B. Starting position and hash value of all words of same length are stored in same row corresponding to length of the word in its table. Table B is sorted row wise using insertion sort with hash value H as the key to sort and perform corresponding change in table A, as hash value is mapped to a unique starting position in table. In the searching phase: When a pattern P is read its hash value H is computed using same hash function and its length is computed. This hash value H is searched in each tables in rows that corresponds to words of same length. Binary search is used to search the hash value in table B. When the hash value of pattern matches with the hash value of a word in table B, then using starting position of word in table A, the word is matched character to character with the pattern P. If a complete match occurs, the occurrence of pattern P in text T is reported. [2, 3, 4]

III. Proposed Algorithm

In this section, we explain the proposed algorithm for modification of algorithm explained in section 2. In our proposed algorithm: it is predicated that text is offline. Proposed algorithm works in two phases as follows.

1) Preprocessing Phase

The whole text splits into k equal parts according to size of the text T. For each part, one table having two columns is created. In the pre-processing phase: each part of the text is read. While reading starting position, hash value of words are computed and stored in the table. The SDBM hash function based on bit shifting is used to compute the hash value H of the words in text T and pattern P. In SDBM hash function the hash value H is computed as follows:

$$H = (H<<6) + (H<<16) - H + ch$$

Where ch is the ASCII value of each characters in the word w, H is initialized as zero, "<<" is a bitwise left shift operator. The SDBM is a standard hash function which has very less chances of collision, even in a very large text. The SDBM implementation is based on an algorithm by P.A. Larson known as "Dynamic Hashing" [4]. Algorithm 1 shows the preprocessing phase.

2) Searching Phase

In pre-processing phase we construct a table and store the starting position and hash value in the table. Table is stored row wise. In a split part if same hash value is found then starting positions are stored in same row for same hash value. In searching phase, balanced binary search tree is constructed for the hash value stored in table. When a pattern P is read then its hash value H is computed using SDBM hash function. While computing the hash value of pattern P, same hash function is used which used to compute the hash value of words stored in the table. Hash value H of the pattern P is searched in the balanced binary search tree. When the hash value H of the pattern matches with the hash value of a word in tree, then using starting position of word in table, the word is matched character to character with the pattern P. If complete match occurs, the occurrence of pattern P in text T is reported. A pattern may exist any number of times in the text so we solve this problem while reading the text. We store the starting position of words of same hash value in the same row of the table. For each hash value of same word matches with hash value of the pattern P, character to character comparison is done and occurrence is reported, if complete match occurs. If hash value of the pattern is not found then it moves to the next table and same process is performed. Algorithm 2 shows the searching phase.The drawback of WSA algorithm given by Ibrahiem et al [1] was that when a pattern P is to be searched, they compute the pattern length and search the table in row corresponds to same length in a brute force approach (i.e. every word in a row in a table is compared character to character which is very inefficient way of searching). In proposed algorithm, it is optimized the searching by tree searching the hash value instead of character comparison [2]. WSA is optimized in proposed algorithm by using a single table in pre-processing phase and balanced binary search tree in searching phase. Our proposed algorithm has less complexity than previous algorithms and requires very less number of character comparisons than the previous one. Figure.1 shows the flow chart of the proposed algorithm.

3) SDBM Hash Function

In proposed algorithm for word searching problem, we use key distribution for every words in the text. For key distribution we use SDBM hash function. We use SDBM hash function in our proposed algorithm because it has very less chance of collision, even in a very large text.SDBM hash function is based on bit shifting. In SDBM hash function the hash value H is computed as follows:

$$H = (H<<6) + (H<<16) - H + ch$$

Where ch is the ASCII value of each characters in the word w, H is initialized as zero, "<<" is a bitwise left shift operator.The SDBM hash function has a good overall distribution for many different data sets. It works well in

situations where there is a high variance in the MSBs of the elements in a data set. It was found to do well in scrambling bits, causing better distribution of the keys and fewer splits. It also happens to be a good general hashing function with good distribution. [2]The SDBM implementation is based on an algorithm by P.A. Larson known as "Dynamic Hashing" [4].

Algorithm 1 Pre-processing phase: Table creation using starting position and hash value

Input: Offline text T

1. BEGIN
2. Split _text (K_i, T)
3. FOR i=0 to n
4. Creat_table (K_i [x][y])
5. FOR x=0 to n
6. FOR y=0 to n
7. INSERT (K_i[x][y]=starting position(w_i),hashvalue(w_i))
8. END FOR
9. Creat_Binary_search_tree(hash value (w_i))
10. END FOR

Algorithm 2 Searching phase: search pattern through hash value

Input: pattern (P)

1 compute_hash_value (pattern(P))
2 search (hash value (P))
3 IF (hash value (P)=hash value(w_i)) THEN
4 GOTO creat_table (K_i [x][y])
5 FIND starting position(w_i)
6 MATCH (pattern(P) with Word(w_i))
7 ELSEIF
8 GOTO (Creat_Binary_search_tree(hash value(w_i)))
9 HALT

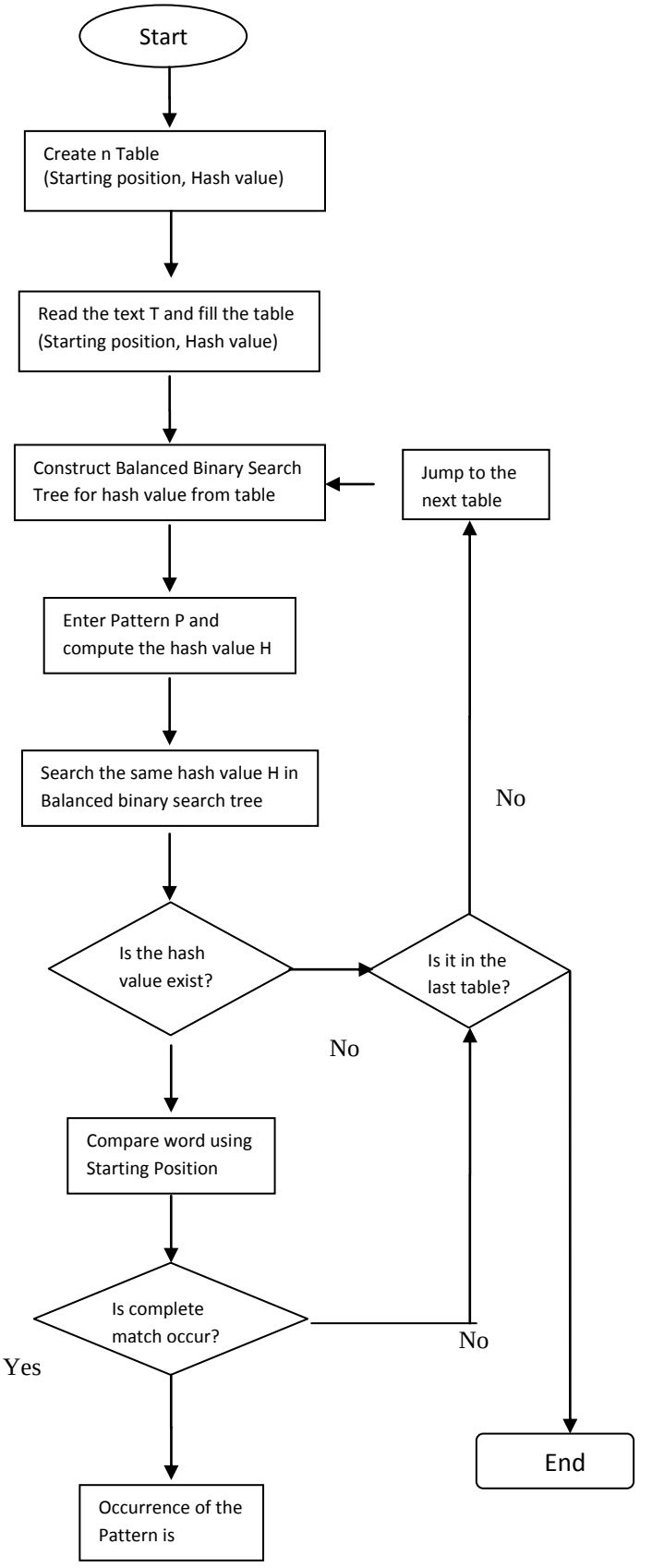

Figure.1 Flow chart of proposed algorithm

IV. Simulation and Results

We have implemented both algorithms WSA and proposed algorithm in C, compiled with GCC 4.2.4 compilers on CORE^{tm} 2 duo 2.66 GHz machine with 4 GB RAM, running Open suse 11.0 Comparison between word matching algorithm. The pattern and text are chosen from ASCII character set randomly.

1) Simulation

For the comparison, we compare WSA algorithm with propose algorithm. Now, we will make a comparison between the proposed algorithm one of the most famous algorithm in such area which is the WSA algorithm to find out the improvement in the number of character comparisons that is done in each algorithm. As an example, we'll take the next paragraph to apply the algorithm with the patterns; "sensor", "processing". Wireless sensor network have emerged as an important application of
1 10 17 25 30 38 41 44 54 66
The ad hoc networks paradigm like monitoring physical environment and
69 73 76 80 89 98 103 114 123 135
These sensor networks have limitations of system resources like battery
137 143 150 159 164 176 179 186 196 201
Power communication range and processing capability Lowprocessing power and
209 215 229 235 239 250 261 265 276 282
In first phase of the simulation each line of this paragraph forming one part of the text, so we have four parts and each part have one table. Each table has two columns. First column has hash value of each word, which is computed by the SDBM hash function and second column has starting position of each word. This is called Preprocessing Phase of the algorithm. The tables will be as following:

Table1. 1st part of the text

Hash Value	Starting Position
3249753982	1
4180602746	10
1691730926	17
385796840	25
2805811025	30
6363218	38
6363213	41
3634954210	44
220492368	54
7281591	66

Table2. 2nd part of the text

Hash Value	Starting Position
700788817	69
6363203	73
866478108	76
2492020741	80
2065330219	89
3573516087	98
1595209896	103
2646868407	114
187585459	123
808581975	135

Table3. 3rd part of the text

Hash Value	Starting Position
1652765827	137
4180602746	143
2492020741	150
385796840	159
1325831833	164
7281591	176
817658959	179
506364869	186
3573516087	196
1906312109	201

Table4. 4th part of the text

Hash Value	Starting Position
2706407781	209
4119183190	215
384379389	229
808581975	235, 282
2081762867	239, 265
1028192824	250
635155988	261
517759365	276

In second phase of the simulation, a pattern "sensor" is taken to search in the text. The hash value of the pattern H is computed by using the SDBM hash function. Now H make searched with the each hash value in each table by using balanced binary search tree. At the time of searching the every comparison is counted. Balanced binary search tree is generated for each table one by one. As the hash value is found the pointer go to the starting position to match the pattern. If the pattern is matched the match report is displayed otherwise negative report is displayed and pointer move to the next comparison in the tree.

Case I. If we have same hash value of the word in the same table then in preprocessing phase the starting position of each hash value will be saved in the same row as shown in Table4. In Table4, Hash value 808581975 has two different starting positions in the same table but saved at the same row.

Case II. If any case pattern does not match with the word at the same starting position then the negative report is displayed and the pointer will be moved to next starting position in the same row.

Case III. Number of comparisons are counted as the pointer move to the next to next node of the balanced binary search tree for searching the hash value of the pattern as well as the number of pattern match at the different starting positions for the same hash value.

2) Comprative Results

As a comparative illustration between the proposed algorithm and WSA algorithm, the character comparison is taken as a parameter. We show the output results as show in the next figure2 and figure3 for the pattern "sensor" and "processing". The results show that how much comparison is done for given patterns. 20 and 40 maximum characters have been taken for the first and second illustration respectively.For the pattern "sensor" which has hash value 4180602746, only 2 comparisons have been taken in proposed algorithm for the first search and 7 comparisons have been done in WSA algorithm for the first search within 20 maximum characters. After taken 40 maximum characters, 10 comparisons and 26 comparisons have been done in propose algorithm and WSA algorithm respectively.For the pattern "processing" which has hash value 2081762867, is not find with in 20 maximum characters comparisons , but after taken 40 maximum characters, 16 and 17 character comparisons have been done for first and full pattern search in proposed algorithm and 41 and 42 comparisons have been done for first and full pattern search in WSA algorithm.

Table5.Comparative results after number of character comparison

	Algorithm comparison based on character comparison			
	Pattern	Algorithms	Number of comparisons	
			First pattern comparison	Full patterns comparison
1	"sensor"	Word Searching Algorithm (WSA)	7	26
		Proposed Algorithm	2	10
2	"processing"	Word Searching Algorithm (WSA)	41	42
		Proposed Algorithm	16	17

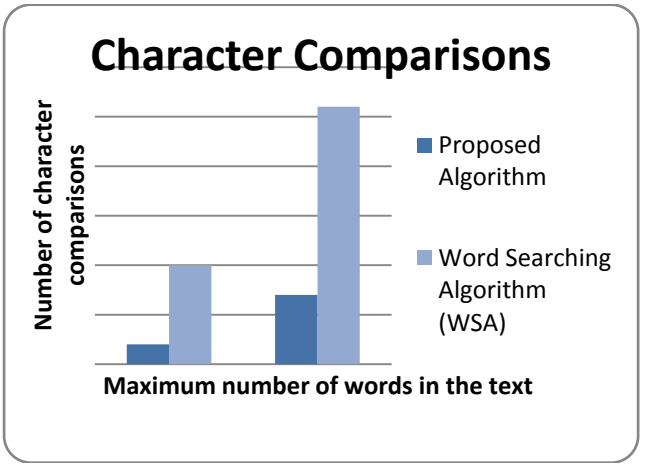

Figure2. Character comparisons in view point of number of character comparisons (Y axis) against maximum number of words in the text
(X axis) for the pattern "sensor".

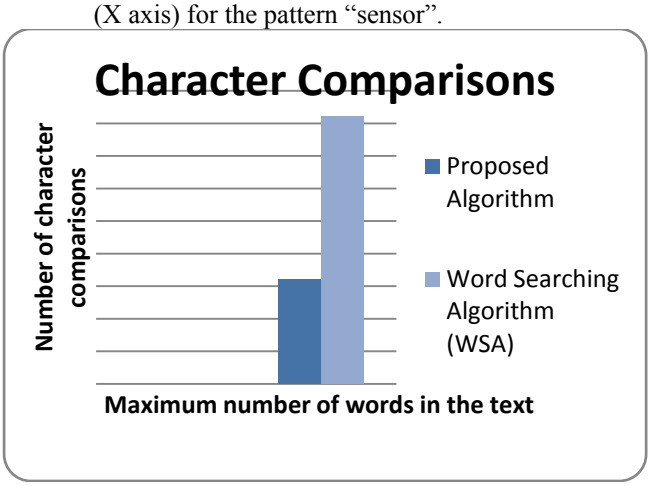

Figure2. Character comparisons in view point of number of charactercomparisons (Y axis) against maximum number of words in the text (X axis) for the pattern "processing".

V. Conclusions

We introduce a new algorithm, which reduces the comparison of word searching algorithm (WSA). From above result it is clear that the proposed algorithm has taken less comparison to find the pattern than word searching algorithms (WSA). In future work proposed algorithm can be compared with the modified word searching algorithm (MWSA) and other pattern matching algorithms.

VI. References

1) Ibrahiem M. M. Emary and Mohammed S. M. Jaber, "A New Approach for Solving String Matching Problem through Splitting the Unchangeable Text", World Applied Sciences Journal 4 (5): 626-633, 2008.

2) Bharat Singh, Ishadutta Yadav, Suneeta Agarwal, Rajesh Prasad, "An Efficient Word Searching Algorithm through Splitting and Hashing the Offline Text", artcom, pp.387-389, 2009

International Conference on Advances in Recent Technologies in Communication and Computing, 2009.

3) R. J. Enbody and H. C. Du, "Dynamic Hashing Schemes", ACM Computing Surveys, vol. 20, no. 2, 85-113, 1988.
4) P. A. Larson, "Dynamic Hashing", BIT, vol. 18, 184-201, 1978.
5) Sorting and Searching Algorithms, http://www.epaperpress.com
6) Donald E. Knuth, "Sorting and Searching, volume 3 of: The Art of Computer programming", Addison Wesley, 1981.
7) R. S. Boyer, and J. S. Moore, "A fast string-searching algorithm", Communication of ACM, 20(10), pp. 762-772, 1977.
8) A.V.Aho, and M.J. Corasick, " Efficient String Matching: An aid to bibliographic search", Communication of ACM 18(6), pp. 333-340, 1975.
9) R. Prasad and S. Agarwal, "An Efficient String Matching by using Super Alphabet" proc. of the first International Conference on Emerging Trends in Engineering and Technology (available on IEEE Xplore), Nagpur, India., pp. 1181-1186, July 16-18, 2008.