ISSN:0975-9646

Hash Table Based Word Searching Algorithm

Sanket Jain, Manish Pandey

Computer Science and Engineering Department
Maulana Azad National Institute of Technology
Bhopal, India

Abstract-**Word searching techniques are used to find all the appearances of the word in the given text. For offline or non-changeable text, it is carried out in two phases: preprocessing phase and searching phase. The two main criteria for word searching algorithms are search time and space overhead. The existing algorithms consume more time and space. To improve the efficiency of the proposed algorithm, SDBM hash function with a heuristic is adopted. By implementing the technique, search time is reduced proportionally to constant to find single pattern.**

Keywords- String matching; hashing; SDBM; heuristic; word searching.

I. INTRODUCTION

Finding the position where a word (or pattern) first appears in the given string is called Word Searching (Pattern Searching). In word searching, we check for presence of word 'W' in the given offline text 'T'. Many algorithms have been propounded for solving the word searching problem like Word Searching Algorithm (WSA) [1], Modified Word Searching Algorithm (MWSA) [2] etc. Search time and space overhead of these algorithms are high.

These issues motivate the implementation of a new idea to improve the overall performance of the searching algorithm.

An algorithm is a sequence of instructions that act on some input data to produce some output in a finite number of steps. It must be able to solve the problem in an efficient way. Hashing schemes generate the key to identify each word. Hash functions are used to generate unique key. A hash function is any well-defined procedure or mathematical function that converts a large, possibly variable amount of data into small datum that may serve as an index to an array. A hash table is a generalization of the simpler notion of an ordinary array. Directly addressing into an ordinary array makes effective use of our ability to examine an arbitrary position in an array in O(1) time[5]. SDBM hash function with a heuristic is used in the proposed algorithm. SDBM has very less chance of a collision, even in a very large text.

This paper is organized as follows: Section 2 describes related word searching algorithms. Section 3 describes proposed algorithm and implementation of the proposed algorithm. Section 4 describes the simulation and result analysis. Finally sections 5 conclude the paper.

II. RELATED WORK

In [1] Word Searching Algorithm (WSA) is propounded. According to WSA algorithm, the offline text splits into 'n' equal parts depending on the size of the text. A table is constructed for each part having two columns; one is for length and second is for starting position of the word. The length is taken as a key for searching.

In [2] Modified Word Searching Algorithm (MWSA) is propounded. According to MWSA, the offline text splits into 'n' equal parts and two tables are created for each splinted part. First table has two columns; length and starting position and second table also has two columns; length and hash value. The hash value is the key for searching.

III. HASH TABLE BASED WORD SEARCHING ALGORITHM (HTWSA)

This section describes the idea and implementation of proposed algorithm: Hash Table Based Word Searching Algorithm (HTWSA). In HTWSA, it is predicated that text is offline. HTWSA algorithm works in two phases: Preprocessing phase and Searching phase.

a. Preprocessing Phase

The whole text is read to get the starting position of each word. While reading starting position, hash values of words are computed. Starting positions of words are stored at respective hashed index slot of hash table HT (multiple small size hash tables can be used, if the text is very large). A heuristic SDBM hash function based on bit shifting is used to compute the hash value 'h' of the words in text 'T' and word 'W'. In SDBM hash function, the hash value 'h' is computed as follows:

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

Where ch is the ASCII value of each character 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 collisions, even in a very large text. The SDBM implementation is based on an algorithm by P.A. Larson known as "Dynamic Hashing" [4]. A heuristic of folding is used to limit the size of the key. Algorithm1 shows the preprocessing phase.

Algorithm1 explains the preprocessing phase of the algorithm

Algorithm1 Preprocessing Phase

```
1. Algorithm pre_pro (offline text T)
2. {
3. insert_(T)
4. for i := 0 to n do
5. {
6.        h:=H_SDBM_fn(w_i);
7.       HT[h]= start_ptn (w_i);
8.        }
9. }
```

b. Searching Phase

In pre-processing phase algorithm stores the starting positions at respective hashed index slot. If same hash value is found then starting positions are stored in same slot separated by a special symbol. In searching phase, a word 'W' is inserted and its hash value 'h' is computed using the heuristic SDBM hash function. Starting position is extracted from respected hashed index slot of hash table; then through starting position of word, character comparison is started. If complete match occurs, the occurrence of word 'W' in text 'T' is reported.

A word may exist more than once in the text. This can be handled while reading the text. The starting position of the same pattern is stored in the same slot of hash table. If the starting position is found more than once in same slot then the comparison is done for each starting position of the word in the text. If hash value of the word is not found then "Word not found" is reported. Algorithm2 explains the searching phase of HTWSA algorithm.

Algorithm2 Searching Phase

```
1. Algorithm Searching_ (Word W)
2. {
3. h:=H_SDBM_fn(W);
4. y:=HT[h];
5. if(y!=0) then
6. write("Word Found");
7. else write("Word not found");
8. }
```

Both phases are explained by a flow diagram in fig.-1.

c. Heuristic SDBM Hash Function

In HTWSA, key distribution is used for every word in the text. For key distribution, SDBM hash function with a heuristic of additive folding is used. SDBM hash function has very less chance of collision, even in a very large text [3][4].

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 character 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]. A heuristic of additive folding is used to limit the size of the key. Algorithm3 shows the heuristic SDBM hash function.

Algorithm3 Heuristic SDBM Hash function

```
1. Algorithm H_SDBM_fn ()
2. sdbm_hash (input : address of key)
3. {
4.         while (c != count) do
5.         {
6.                 h=(*key++)   +(h<<6)+(h<<16)-
        h;
7.                 count := count+1;
8.         }
9. Additive_hrstc(h);
10. return key;
11. }
```

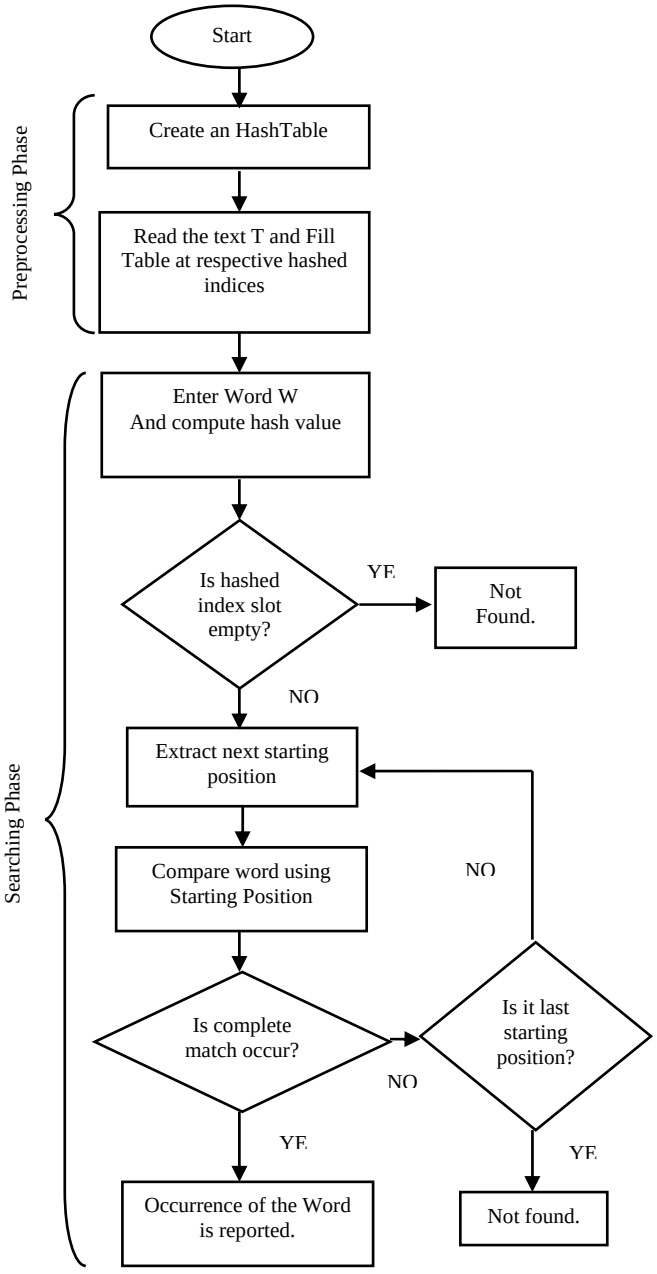

Fig-1 Flow diagram of HTWSA

IV. Simulation and Results

a. *Simulation*

The next paragraph is taken as an example to simulate the HTWSA algorithm:

"Wireless sensor network have emerged as an important application of the ad hoc networks paradigm like monitoring physical environment and these sensor networks have limitations of system resources like battery power communication range and processing capability."

Starting positions corresponding to this paragraph are as follows:

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

209 215 229 235 239 250

In pre-processing phase of simulation hash values of each word are computed using heuristic SDBM hash function and starting positions of each word are stored at respective hashed index slot. In this example, Additive folding is applied twice to make index smaller and easy understanding. The table below shows the hashed index and starting position of words:

Table1. Hashed Index and Starting Position

Hashed Index	Starting Position
002	186
109	250
122	30
136	164
150	135#235
202	179
203	201
315	229
329	73
339	41
344	38
406	215
421	123
436	137
565	1
596	10#143
649	44
666	54
698	25#159
706	80#150
744	66#176
767	239
858	76
873	98#103#196
879	209
890	17
919	69
922	89
969	114

In the second phase of simulation, a word "sensor" is taken to search in the text. The hash value of the word 'W' is computed by using the heuristic SDBM hash function. As the hash value of "sensor" is calculated as 596; the pointer goes to HT[596] and extract the starting position to match the word. If the word is matched completely, positive output is displayed otherwise negative output is displayed.

Case I. If we have same word more than once then in preprocessing phase the starting position of same word will be saved in the same hashed index slot separated by a special symbol as shown in Table1. In Table1, Hashed index 596 has two different starting positions 10 and 143 but saved at the same slot. If word 'W' does not match with the word at the current starting position then the pointer will be moved to next starting position in the same slot.

Case II. The number of comparisons is counted as the pointer moves to the next starting positions for the same hash value. If the word 'W' is not present in the text 'T' then the number of comparisons is zero.

b. *Comparative Results*

As a comparative illustration among WSA, MWSA and the proposed algorithm HTWSA, the pointer movement(character comparison) is taken as a parameter. Table2 shows the output results for the pattern "sensor" and "networks". The results show that how much comparison is done for given patterns.

For the pattern "sensor" which has hashed index 596, only 1 comparison has been taken in HTWSA algorithm for the first search, 5 comparisons have been done in MWSA algorithm and 7 comparisons have been done in the WSA algorithm for the first search. In full text 2 comparisons, 18 comparisons and 26 comparisons have been done in HTWSA algorithm, MWSA algorithm and WSA algorithm respectively.

Table2.Comparative results for number of comparisons

Algorithm comparison based on word comparison			
Pattern	*Algorithms*	*Number of comparisons*	
		First pattern comparison	*Full patterns comparison*
"sensor"	WSA Algorithm	7	26
	MWSA Algorithm	5	18
	HTWSA Algorithm	1	2
"networks"	WSA Algorithm	18	37
	MWSA Algorithm	7	13
	HTWSA Algorithm	1	2

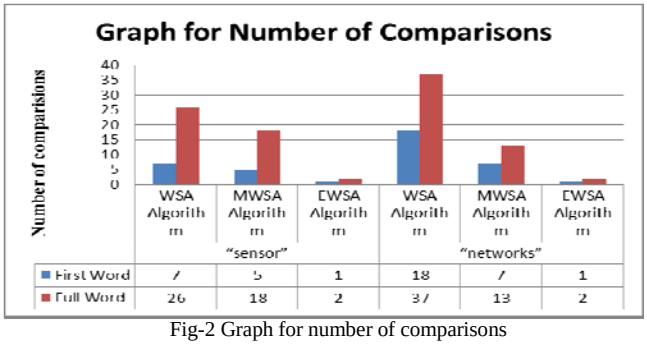

Fig-2 Graph for number of comparisons

c. Complexity Analysis

In the pre-processing phase of HTWSA, reading the text will take complexity of O(n), which execute only once in the offline text. In searching phase, there is no need of sorting the list or no need of binary search. If the number of hash-table slots is atleast proportional to the number of elements in the table, we have n = O(m) and, consequently, a = n/m = O(m)/m =O(1). Thus, searching takes constant time on average. Table3 shows the comparison of the complexities of both phases in WSA, MWSA and HTWSA algorithm. In WSA algorithm, merge sort is used for sorting the tables and brute force search is used in searching phase[1]. In MWSA algorithm, Insertion sort is used for sorting and binary search is used in searching phase[2].

Table3. Complexity Analysis

S.No.	Complexity Analysis of algorithms			
	Name of algorithm	Phase	Technique	Complexity
1	Word Searching Algorithm (ESPS)	Preprocessing phase	Read the text	O(n)
			Merge sort	$O(n\log_2 n)$
		Searching phase	Brute force manner	O((n-m+1)*m)
2	Modified Word Searching Algorithm (MWSA)	Preprocessing phase	Read the text	O(n)
			Insertion sort	$O(n^2)$
		Searching phase	Binary search	$O(\log_2 n)$
3	Hash Table Based Word Searching Algorithm (HTWSA)	Preprocessing phase	Read the text	O(n)
		Searching phase	Hashed Index Array	O(constant)

V. CONCLUSIONS

This paper has presented the new algorithm named as "Hash Table Based Word Searching Algorithm (HTWSA)" for word searching problems. Decreasing the searching time is the main aspect of word searching algorithm. The main advantage of this algorithm is excluding the search for the text which is not needed or not present in the text. For future work, this algorithm can be used in text editors. Filling the cells of the array as soon as finishing writing the text will reduce the time of pre-processing and the algorithm is ready to process at any time.

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] Thomas H.Cormen, Charles E. Leiserson, Ronald L. Rivest and Clittord Stein, "An Introduction To Algorithms", McGraw-Hill Book Publication, First Edition,1990.

[6] R. S. Boyer, and J. S. Moore, "A fast string-searching algorithm", Communication of ACM, 20(10), pp. 762-772, 1977.

[7] www.encyclopedia.com