Volume 5, Issue 6, June 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Comparative Study of Bit Parallel String Matching Algorithm

Sonam Jain
Computer Science (M. Tech)
LNCTS, Bhopal, (M.P.) India

Prof. Vivek Kumar
Computer Science
LNCTS, Bhopal, (M.P.) India

Abstract— String matching algorithms become one of the important topics in computer engineer's life. As the technology upgraded the requirement of the fast pattern searching is needed. Character based searching is one of the popular types of string matching used in various algorithm whose speed up can be increased up to certain level by making maximum shift distance. Due to this bit parallel string matching algorithms comes into existence which uses the bitwise operator which is faster than the comparison operator. Various bit parallel string matching algorithms are Shift OR, BNDM, TNDM, BNDMq, Shift OR with Q gram and Filtering based multiple BNDM. This paper discus the various bit falls of the various bit parallel string matching algorithm with their detailed explanation and comparative analysis between these algorithms.

Keywords— String Matching,Bit Parallel string matching, Shift OR, BNDM, TNDM, Shift OR with Q gram, BNDMq, Filtering based Multiple BNDM.

I. INTRODUCTION

String Matching is a traditional problem in the computer world. It is uses most of the places where comparison is required. String Matching [1] is to find all occurrences of a given Pattern in a large Text and report the position of the occurrences. String matching algorithms are used widely in the real world like as Intrusion Detection system [2][3], Plagiarism detection [4], Data Mining [5] and Bioinformatics [6].There are many algorithms for string matching which is based on the character comparison like as KMP [7], BM [8], BMH [9] and Aho-Corasick [10] etc. While on other hand we have algorithm based on bit operation called as Bit Parallel Algorithm for string matching [14]. These algorithms are faster than the character based algorithm because it uses the inherent property of computer to perform Bit Wise operation. By the use of this method the speed of string matching is increased. Bit Parallelism algorithm is based on the non-deterministic automata but there is no such automata are present. It is simply the efficient simulation of non-deterministic automata. So as compare to other algorithm like as Aho-Corasick [10] where we have to store the whole automata in memory which become very complex when pattern size is increases Bit Parallel algorithm are very space efficient. Firstly it is introduced by Domolki and later revisited by Baeza–Yatesand Gonnet and Wu-Manber [14] so on but it is performed only on data available in single computer word. Bit parallelism inherently favours parallelism of bit operations within computer word. It comprises the machine level operations which can be carried out in parallel by utilizing the word size of computer. On modern architectures bit wise operation has same speed as addition but is significantly faster than the multiplication and division.

Sting Matching using Bit Parallelism can be classified into two broad categories that are single pattern matching and multiple pattern matching which is shown in figure 1. In single pattern matching there is a single pattern which is searched in to the text and reports occurrences. In multiple pattern matching we have number of pattern whose occurrences we have to report. As compared to single pattern matching multiple patterns matching algorithm has lots of practical application in real life.

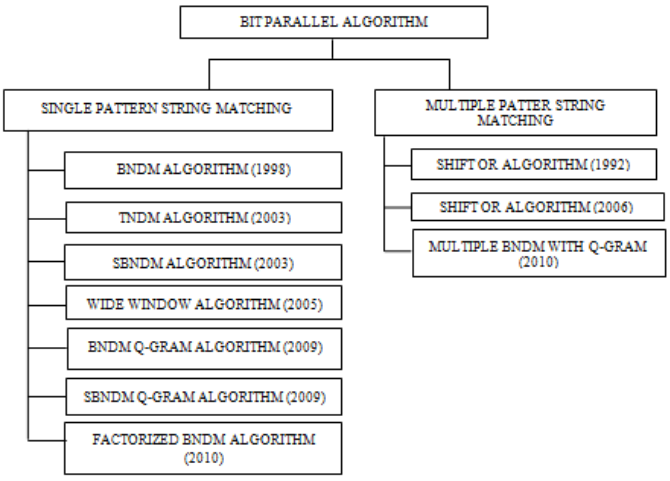

Figure 1: Evolution of Bit Parallel Algorithm

II. SINGLE PATTERN BIT PARALLEL ALGORITHM

Single pattern matching means finding the pattern in the text and reports the occurrences of the pattern. We have various single pattern bit parallel string matching algorithm which are discussed one by one below with the help of the example.

a) BNDM Algorithm

BNDM stands for Backward Non Deterministic Matching which is introduced by Navarro and Raffinot in 1998[11]. It is exact single pattern string matching algorithm. In BNDM order of searching is from right to left. It uses the concept of bit parallelism from shift OR algorithm and suffix automata from BDM algorithm. This algorithm is a bit parallel simulation of BDM algorithm. BDM skips character using suffix automata which is deterministic in preprocessing. To construct Deterministic automata is complex task. BNDM simulates the non-deterministic version using bit parallelism.

BNDM algorithm consist in two phase

1. Pre-processing phase 2. Searching phase

Let us understand the working of whole algorithm with the use of an example. Example of BNDM is as follow. Let Text T = 'STRINGCARE' and Pattern P= CARE

Pre-processing: In pre-processing phase we find Bit Vector of each Character of the Pattern calculated by putting 1 for occurrence and 0 for non-occurrence.

B[C] =1000, B[A] =0100, B[R] =0010, B[E] =0001 and B[OTHER] =0000

Searching Phase: Initially we take some variable and set to the pattern length

i.e. J=4, last=4, pos. = 0 and bit vector D =1111

Than we perform the BNDM algorithm whose various step are shown in the table 1 with their explanation of each steps.

Table 1: Shows the various steps perform by BNDM

SCANNING PHASE							
STEP	TEXT	D	B[k]	D'	J	LAST	MATCHING
1	STRINGCARE	1111	-	-	4	4	
2	STRINGCARE	1111	0000	0000	3	4	
3	STRINGCARE	1111	-	-	4	4	
4	STRINGCARE	1111	0100	0100	3	4	
5	STRINGCARE	1000	1000	1000	2	2	
6	STRINGCARE	0000	0000	0000	1	2	
7	STRINGCARE	1111	-	-	4	4	
8	STRINGCARE	1111	0001	0001	3	4	
9	STRINGCARE	0010	0010	0010	2	4	
10	STRINGCARE	0100	0100	0100	1	4	
11	STRINGCARE	1000	1000	1000	0	4	CARE

BNDM algorithm become very fast string matching algorithm except very sort (0-6) or very long (90-150) pattern. It is faster than the previous algorithm Shift OR, BDM and Occupies very less space perform various operation in parallel. It is very Simple and flexible algorithm. But we have all pattern assume to less than or equal to the word size of computer.

b) TNDM Algorithm

TNDM stands for Two ways Non Deterministic Matching which is introduced by Peltola and Tarhio in 2003[12]. It is Exact Single pattern string matching algorithm. Here order of scanning is from left to right. It is almost same as the BNDM algorithm there is slight changes in the case of mismatch occur at first position instead of shifting TNDM look forward to find suffix of reverse pattern. The number of examined characters is less than BNDM therefore matching is faster. The simulation of TNDM Algorithm can be understood by the help of the Figure 2.

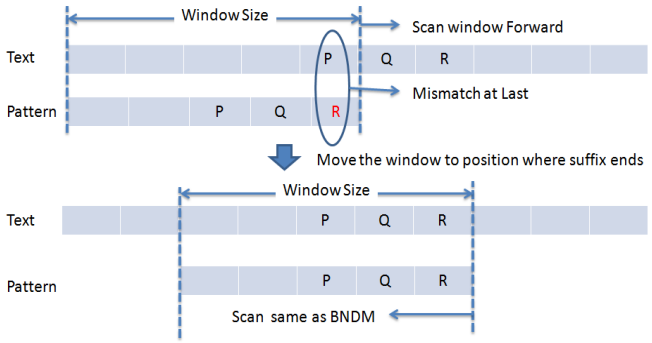

Figure 2: Working explanation of TNDM algorithm

TNDM uses bit parallelism through which it become faster algorithm. It uses forward scan so the number of examined character is less than the BNDM in general. Remaining algorithm is work same as BNDM algorithm where all pattern assume to less than or equal to the word size of computer.

c) SBNDM Algorithm

SBNDM [12] stands for Simple Backward Non Deterministic Matching. SBNDM is much similar to the BNDM algorithm there is a slight change in term of shifting. Due to this it is quit faster than the BNDM algorithm. Here we do not require finding the longest prefix. Let T is the text of length n and P is the pattern of length m to be searched. At each alignment window of P in T, Scan T from right to left until the suffix of the window is not a factor of P or an occurrence of P is found. Shifting of SBNDM is according to this cases i.e. shift window by m if no factor is found, shift by 1 if P found otherwise next alignment is start at last factor. Table 2 describes the various steps of the SBNDM algorithm.

Table 2: Working Explanation of SBNDM

Pattern= 'DESIGN', Text = 'SFZIGNBACDESIGN'

ALIGNMENT	S	F	Z	I	G	N	B	A	C	D	E	S	I	G	N
						N									
					G	N									
				I	G	N									
NOT A FACTOR			Z	I	G	N									
NEXT ALIGNMENT	S	F	Z	I	G	N	B	A	C	D	E	S	I	G	N
NOT A FACTOR									C						
NEXT ALIGNMENT	S	F	Z	I	G	N	B	A	C	D	E	S	I	G	N
PATTERN FOUND										D	E	S	I	G	N

By using the SBNDM concept the average length of shift is reduced by doing so the innermost loop of algorithm become simpler. It is faster than the BNDM algorithm.

d) Forward SBNDM Algorithm

Forward SBNDM [15] is also known as FSB introduced by Faro and Lecroq. It is enhanced version of the BNDM algorithm which use the Non Deterministic Automaton augmented of new initial state in order to take into account the forward character of the current window of the text. FSB reads a 2-gram x1x2 before a factor test. Only x1 is matched with the end of P in FSB and x2 is a look ahead character.

Let UV be a q-gram, where |V| = f. After reading UV there are 3 alternatives:

If U is a suffix of P, reading continues leftwards.

Else if UV is a factor of P, reading continues leftwards.

Else the state vector is zero and P is shifted m-q+f+1 positions

e) BNDM with Q-gram

BNDM with Q-gram [19] is an improved variation of the BNDM algorithm which reads the q gram at each alignment before testing the state variable. In this algorithm loop has been made as sort as possible in order to quickly advance m-q+1 position. Here q can be varies according to our requirement. The whole algorithm can be easily understood with the help of the example given below.

Let Text T = 'STRINGCARE' and Pattern P= 'CARE' Assume we have 2-gram

Pre-processing: In pre-processing phase we find Bit Vector of each Character of the Pattern calculated by putting 1 for occurrence and 0 for non-occurrence.

B[C] =1000, B[A] =0100, B[R] =0010, B[E] =0001 and B[OTHER] =0000

Searching Phase: Initially we take variable i and set to m-q+1 where m is pattern length

Than we perform the BNDMq algorithm whose various step shown in the table 3.

Table 3: working of BNDMq algorithm

SCANNING PHASE								
STEP	TEXT	I	D	B[k]	D'	J	FIRST	MATCHING
1	STRINGCARE	3	0000	-	-	-	-	
2	STRINGCARE	6	0000	-	-	-	-	
3	STRINGCARE	9	0010	0010	0100	8	6	
4	STRINGCARE	9	0100	0100	1000	7	6	
5	STRINGCARE	9	1000	1000	0000	6	6	CARE

It is become very fast string matching algorithm except very sort (0-6) or very long (90-150) pattern. It is faster than the previous algorithm BNDM and Occupies very less space and simple to implement.

III. MULTIPLE PATTERN BIT PARALLEL ALGORITHM

Multiple pattern string matching means we have number of pattern and text so we have to find out the occurrence of these pattern in the text. We have various bit parallel multiple string matching algorithm. In this section we discuss these multiple algorithm one by one with the help of example.

a) Shift OR Algorithm

It is a first algorithm which uses the concept of bit parallelism introduced by Baeza Yates and Gonnet in 1992[13]. It is a approximate multiple pattern string matching algorithm which means it search number of pattern at a time but there is a possibility of error. In Shift OR algorithm order of searching is from left to right. It is design for large pattern have equal length and equal or less than the word size.

Many multiple pattern algorithm build a trie of the pattern in pre-processing phase so as the pattern size increases size of tree also increase which is not practical to maintain. Shift OR algorithm is a simulation of nondeterministic automata where we do not need to build any trie. They don't need to buffer the input. It is real time algorithm suitable to be implemented in hardware.

Shift OR consist into two phase:

1. Pre-Processing Phase 2. Searching Phase

Let's take an example to understand Shift OR algorithm. Suppose CARE and TINB be the patterns of length 4 and STRINGCARE be the text.

Pre-processing phase: In pre-processing phase we find out the bit vector of the every character of alphabet. In this 'i_{th}' bit is zero if and only if character appears at position 'i' otherwise place 1 and write it into reverse order.

B[C] = 1110, B[A] = 1101, B[R] = 0100, B[E] = 011, B[T] = 1110, B[I] = 1101, B[N] = 1011, B[B] = 0111 and B[OTHER] = 1111.

Searching Phase: The automation has a transition from state 'i' to 'i+1' on character c if and only if 'i_{th}' bit in B[c] is 0. In state vector D where 'i_{th}' bit is 0 if and only if state 'i' in the automation is active. If 0 is occurs at MSB means we find the pattern at position. Occurrence position = pos - pattern length + 1. All the steps involved are shown in the table 4.

Table 4: Working of Shift OR algorithm

SCANNING PHASE						
STEP	TEXT	D	B[k]	D'	POSITION	MATCHING
1	STRINGCARE	1111	-	-	-	
2	STRINGCARE	1110	1111	1111	1	
3	STRINGCARE	1111	1110	1110	2	
4	STRINGCARE	1101	1111	1111	3	
5	STRINGCARE	1111	1101	1111	4	
6	STRINGCARE	1111	1011	1111	5	
7	STRINGCARE	1111	1111	1111	6	
8	STRINGCARE	1111	1110	1110	7	
9	STRINGCARE	1101	1101	1101	8	
10	STRINGCARE	1011	1011	1011	9	
11	STRINGCARE	0111	0111	0111	10	CARE

In Shift OR algorithm pre-processing and search are very simple and only bitwise logical operations Shift and AND are used and no Buffering is required. It is an real time algorithm. Time delay to process one text character is bounded by a constant depend only on pattern length. It is approximate string matching algorithm so possibility of the error (false match) and Here all pattern length must be equal.

b) Shift OR with Q-Gram

Shift OR with Q-gram [17] is an enhanced version of the Shift OR algorithm. In Shift OR with Q-garm algorithm we take q character at a time for comparison by doing so the size of automata is reduced and the number of comparison for finding pattern is reduced up to certain level. The Q-gram can be of two types: Consecutive Q-gram or Overlapped Q-gram. In Consecutive Q-gram WE read pattern in a sequence of q character at a time while in Overlapped Q-gram we take q character from each character of the patterns. Here the pattern length of each pattern must be same. Shift OR with Q-gram carried out in three phase First phase: Initialization Phase where initialization of the variable is carried out. Second phase: Pre-processing phase where bit vector of the various q gram are taking place. Third phase: Searching phase here searching of the pattern in the text is carried out.

Let's take an example of Shift OR Consecutive 2-Gram where Text is STRINGCARE and Patterns 'CARE' and 'TINB' hence the consecutive 2-gram of first pattern is "CA" and "RE" and second pattern is "TI" and "NB" by doing so the number of bit in bit vector is reduced to the half of the pattern length. Here 2-gram of the pattern is treated as single character. The ith bit of the bit vector is set to zero if there is an occurrence of 2-gram in the ith position of the pattern otherwise set to one. So bit vector of our pattern is as follow.

B [CA] = 10, B [RE] = 01, B [TI] = 10, B [NB] = 01 and B [OTHER] = 11.

Initialise D = 11 and Update D when a character c is read from the text by

D= (D<<1) | B[c]

Various steps involved are shown in table 5.

Table 5: Working of shift Or with Qgram

SCANNING PHASE					
STEP	TEXT	D	B[k]	D'	MATCHING
1	STRINGCARE	11	-	-	
2	STRINGCARE	10	11	11	
3	STRINGCARE	10	11	11	
4	STRINGCARE	10	11	11	
5	STRINGCARE	10	11	11	
6	STRINGCARE	10	11	11	
7	STRINGCARE	10	11	11	
8	STRINGCARE	10	10	10	
9	STRINGCARE	01	01	01	CARE

c) Multiple Pattern BNDM by Combining Q-Gram

BNDM algorithm is a single pattern string matching algorithm which is very fast one because it uses the concept of bit parallelism. Changsheng Miao, Guiran Chang and Xingwei Wang convert the BNDM algorithm in multiple pattern BNDM algorithms [18]. They develop Filtering Based Multiple String Matching Algorithm by Combining q-Grams and BNDM.

Let's understand the concept of algorithm by considering an example Text: gooddooning, Text size: 11, Patterns: good, ning pattern length (m): 4Bit Vector: B[g] = 1001, B[o] = 0110, B[n] = 1010, B[d] = 0001 and B[i] = 0100 and Initialize q = 2, I = m-q+1. The whole searching process is described in the Table 6 step by step with each step explanation.

Table 6: Shows the various steps perform by Multiple BNDM

Text Window	D	I	first	j	Explanation
good	0010	3	0	3	Read 'od', D=B[o]&(B[d]<<1), q matched
good	0100	3	0	2	D = D<<1 & B[o];
good	1000	3	0	1	D = D<<1 & B[g];
good	0000	3	0	0	j=first, Report an occurrence at j+1 D = D<<1; Shift to i=6
ddoo	0100	6	3	6	Read 'oo', D=B[o]&(B[o]<<1), q matched
ddoo	0000	6	3	5	D = D<<1 & B[d]; D=0,Shift to i = 9
onin	0100	9	6	9	Read 'in', D=B[i]&(B[n]<<1)
onin	1000	9	6	8	D = D<<1 & [n];
onin	0000	7	6	7	j>first therefore i=j D = D<<1 & [o]; D=0,Shift to i = 10
ning	0010	10	7	10	Read 'ng', D=B[n]&(B[g]<<1), q matched
ning	0100	10	7	9	D = D<<1 & B[i]
ning	1000	10	7	8	D = D<<1 & B[n]
ning	0000	10	7	7	j=first, Report an occurrence at j+1; D = D<<1 i=13, greater than text Size, Exit.

IV. COMPARISION AND ANALYSIS

Above we describe the general bit parallel string matching algorithms in detail. Each of those algorithms works on the concept of the bit wise operator but these algorithms differ in several case. Here the table 7 gives the detail comparison of the various bit parallel string matching algorithm based on various parameter.

Table 7: Comparison of Various Wu Manber String Matching Algorithms

Algorithm / Parameter	SHIFT OR ALGORITHM	SHIFT OR WITH Q-GRAM ALGORITHM	BNDM ALGORITHM	TNDM ALGORITHM	FILTERING BASED MULTIPLE BNDM ALGORITHM
Operator Used	OR	OR	AND	AND	AND
Scanning	Left to Right	Left to Right	Right to Left	Right to Left	Right to Left
Pattern	Multiple Pattern	Multiple Pattern	Single Pattern	Single Pattern	Multiple Pattern
Number of Character Read	Single	Q-Character	Single	Single	Q-Character
MAXIMUM SHIFT	1	1	M	M	M-Q+1
SIZE OF PATTERNS	EQUAL	EQUAL	-	-	EQUAL

V. CONCLUSION

After development of the processor technology, a whole new series of algorithms has been started for the string matching. Since, string matching plays a big role in applications like DNA matching, plagiarism, data mining; web searching etc. the research is going on all around the world to develop fast and efficient string matching algorithms. The development of Bit Parallel has set a new milestone in this field. Till time many new algorithms, especially those applying bit-parallelism, has been introduced which are much faster than the old ones. When comparing the search speed of two string matching algorithms, several factors affect the result like processor, compiler, and stage of tuning, text, and pattern. Even a small change in the pattern may switch the order of the algorithms. Thus there is no absolute truth which algorithm is better. Because the continuing development of processor and compiler technologies, it is also difficult to anticipate, how present algorithms manage after a few years.

REFERENCES

[1] Christian Charras and Thierry Lecroq," Handbook of Exact String_Matching Algorithms", Published in King's college publication, Feb 2004.

[2] Ali Peiravi, "Application of string matching in Internet Security and Reliability", Marsland Press Journal of American Science, 6(1), pp. 25-33, 2010.

[3] Pei-fei Wu and Hai-juanShen,"The Research and Amelioration of Pattern-matching Algorithm in Intrusion Detection System", In the proc. of IEEE 14th International Conference on High Performance Computing and Communication & IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), pp. 1712-1715, 25-27 June 2012.

[4] Ramazan S. Aygün "structural-to-syntactic matching similar documents", Journal Knowledge and Information Systems, ACM Digital Library, Volume 16 Issue 3, pages 303-329, Aug 2008.

[5] Sanchez D., Martin-Bautista M.J., Blanco I. and Torre C.," Text Knowledge Mining: An Alternative to Text Data Mining", In the proc. of IEEE International Conference on Data Mining Workshops, ICDMW '08, pp. 664-672, 15-19Dec. 2008.

[6] Robert M. Horton,Ph.D. "Bioinformatics Algorithm Demonstrations in Microsoft Excel", California State University, Sacramento, 2004.

[7] Knuth D E, Morris Jr J. H and Pratt V. R," Fast pattern matching in strings", In the procd. Of SIAM J.Comput., Vol. 6, 1, pp. 323–350, 1977.

[8] Boyer R S and Moore J S,"A fast string searching algorithm", Communication of ACM 20, Vol. 10, pp. 762–772, 1977.

[9] Horspool R N,"Practical fast searching in strings", In proc. Of Software Practical Exp, Vol. 10, 6, pp. 501–506, 1980.

[10] Alfred v aho and Margaret j corasick,"efficient string matching: an aid to bibliographic search" communication of acm, vol. 18, June 1975.

[11] G. Navarro and M. Raffinot, "Fast and flexible string matching by combining bit-parallelism and suffix automata",ACM Journal. Experimental Algorithmics 1998.

[12] Hannu Peltola and Jorma Tarhio, Alternative Algorithms for Bit-Parallel String Matching, String Processing and Information Retrieval, Spire 2003Springer, LNCS 2857, pp. 80-93, 2003.

[13] Ricardo A. Baeza-Yates and Gaston H. Gonnet,"A New Approach to Text Searching", In Communications of the ACM, pp. 74-82, Oct 1992.

[14] Faro S. and Lecroq T,"The exact online string matching problem: A review of the most recent results", ACM Comput. Survey, Article 13, 42 pages, February 2013.

[15] Hannu Peltola and Jorma Tarhio," Variations of Forward-SBNDM", In the Proceedings of the Prague Stringology Conference, pp. 3–14, 2011.

[16] Longtao Hea, Binxing Fangaand Jie Sui," The wide window string matching algorithm" In the procd. Of Theoretical Computer Science of Elsevier, Vol. 332, pp. 391-404, 2005.

[17] L. Salmela, J. Tarhio, and J. Kytojoki, "Multi pattern string matching with q-grams", Journal of Experimental Algorithms, Volume 11, pp. 1-19, 2006.

[18] Changsheng Miao, Guiran Chang and Xingwei Wang," Filtering Based Multiple String Matching Algorithm Combining q-Grams and BNDM", In proc. Of Fourth International Conference on Genetic and Evolutionary Computing, 2010.

[19] Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhio," Tuning BNDM with q-Grams" In proc. of workshop on algorithm engineering and experiments SIAM USA, pp. 29-37, 2009.

