String Matching Algorithms Using Bit Parallelism

Gulfishan Firdose Ahmed

Department of Computer Science and Engineering,
Maulana Azad National Institute of Technology
Bhopal-462051

Nilay Khare

Department of Computer Science and Engineering,
Maulana Azad National Institute of Technology
Bhopal-462051

Abstract— String matching will be done through character based and bit parallelism. Bit parallelism is faster than the other character based algorithms. Bit parallelism algorithms play a significant role in the current scenario. Bit operations in current processors are fast. In this paper, some of the bit parallelism algorithms are described with their comparative analysis such as Shift-AND, Shift-OR, Shift-OR with q-grams, BNDM, SBNDM, TNDM, etc.

Index Terms— Shift-OR, Shift-OR with q-grams, Shift-AND, BNDM, SBNDM, TNDM etc.

I. INTRODUCTION

String matching is often used in different areas such as text editors, virus scanning, Bioinformatics, digital libraries and web search engines. String matching, generally including exact string matching and regular expression matching is used in many applications. Bit parallelism is the most important techniques in the field of computer science. In the current years, bit parallelism plays an important role in string matching due to the length of pattern size that can be processed in parallel. Bit parallelism is done by creating bit vectors of the pattern characters, and then the matching is performed with the help of bit operations in parallel. Transformation into bits results in faster results as they can be performed in parallel. Bit parallelism although performs better as compared to other character based or non bit parallel algorithms, but it imposes a limitation on the pattern size. The Shift-And and the Shift-Or algorithms are strongly related to each other.

Algorithm based on the simulation of the non-deterministic factor automaton, is known as Backward Non-deterministic Matching algorithm and it is very efficient for short patterns [1]. The problem of multi-patterns string matching has been solved in the past by bit-parallel strings matching algorithms such as shift-OR and Backward non-deterministic DAWG matching (BNDM) [2].

II LITERATURE SURVEY AND METHODOLOGY

Bit-Parallelism— Bit-level parallelism is a form of parallel computing, based on very-large-scale integration (VLSI) technology. Bit-parallelism is the technique of packing several values in a single computer word and updating them all in a single operation. The main scheme of the bit-parallel algorithms is that they store several data items into a single computer word and then update them using a single computer operation (e.g. bitwise operation. Bit-parallel algorithms are very efficient for approximate string matching algorithms. The bit-parallel algorithms are more efficient that other string matching algorithms for small and long patterns respectively [4]. Bit parallelism is done by creating bit vectors of the pattern of characters and then the matching takes place with the help of bit operations in parallel. Transformation into bits results in faster results as they can be performed in parallel [5].

A) Shift-AND Algorithm: This algorithm workings fast, is easy to implement and can be
easily generalized to the case of approximate searching and it is single pattern string matching algorithm. The very basic algorithm is the Shift-AND algorithm. The Algorithms that uses bit-wise operations is called bit-parallel algorithms. Shift-And algorithm, which is a bit easier to describe.

The Shift-And algorithm simulates the behaviour of the non-deterministic string matching automaton that recognizes the pattern P of length m from the given text [6]. The Shift-AND algorithm needs \( O(n) \) comparisons, which are only of bitwise operations.

### B) Shift-OR Algorithm

The Shift-OR algorithms is both single and multiple pattern matching algorithm. It uses Shift -OR method for matching the patterns. The bit vector is set in a manner that occurrence denotes “0” and non occurrence denotes “1” in corresponding position. It behaves like a filter performing approximate string matching. The pre-processing algorithm constructs B having one bit mask entry for each \( c \in \Sigma \) for

\[ 0 \leq i \leq m - 1, \] the mask B[c] has \( i^{th} \) bit set to 0 if \( P[i] = c \) otherwise it is 1. The bits are numbered from right to left and pattern is scanned from left to right. If the \( i^{th} \) bit of B[c] is zero. In the searching Phase, shift –OR algorithm uses, a bit mask D so that \( i^{th} \) bit of this mask is set to 0. For each text symbol c the state vector D is simplified by \( D \leftarrow (D \ll 1) \mid B[c] \), where \( D \ll 1 \) makes the state active and then looks for transition on that state. If after the \( i^{th} \), the \( (m - 1)^{th} \) bit of D is zero, then there is an occurrence of P with shift i.m. If \( m \leq n \), then the running time of the algorithm is \( O(n) \)[8].

Fig. 1 Shift-OR Example

### C) BNDM (Backward Non Deterministic Matching) Algorithm

In bit parallelism we are performing \((a \& b)\) instead of \((a = b)\). The Pattern size is less than or same as the word size of the computer. Simulation of BDM Algorithm based on Suffix Automaton. BDM moves window over the text. It maintains a set of position on the reverse pattern that is beginning positions of the factors of the text positioned in the window. The vector D keeps a list of positions in pattern where the factor begins. Window is shifted to the position where last factor is found. The formula to update D is : \( D \leftarrow (D \& B[t]) \) \(<< 1\). The main advantage of BNDM is it uses Bit Parallelism and therefore matching process is faster. It is faster than other character based string matching algorithms such BM. The main limitation of BNDM algorithms its worst case time complexity is \( O(nm) \). All the patterns are assumed to be less than or equal to the word size of the computer [2].
BNDM Algorithm

BNDM\((p = p_1p_2 ...p_m, T = t_1t_2 ...t_n)\)

/* Pre-Processing Phase I */
1. For \( i \in 1...m \) do
2. \( B[p_m - i + 1] \leftarrow B[p_m - i + 1] \ 0^{m-1} \)

/* Searching Phase II */
3. \( j \leftarrow 0 \)
4. While \( pos = n - m \) do
5. \( j \leftarrow m \)
6. \( D = i \)
7. \( while \ D! = 0^m \ do \)
8. \( D = D \land B[t_p + j] \)
9. \( j = j - 1 \)
10. If \( D \land 10^{m-1} \neq 0^m \) then,
11. If \( j > 0 \) then \( last = j \)
12. Else report an occurrence at \( pos + 1 \)
13. End of if
14. \( D = D \ll 1 \)
15. End of while
16. \( Pos = Pos + last \)
17. End of while loop

BNDM (Backward Nondeterministic DAWG Matching) is among the best string matching algorithms. It implements a bit-parallel simulation of a nondeterministic automaton. The BNMD Algorithm is based on the Suffix automata for performing string matching from the text. There are many variation of BNMD algorithm such as TNDM, SBNDM, \( BNMD_q \), etc.

### III VARIATIONS OF BNMD ALGORITHMS

**A) TNDM (Two way Non Deterministic Matching):** This algorithms is a single pattern string matching algorithms. It performs two way scanning. The overall working is same as BNMD algorithm but scanning is two way means both the side. If the Text Character aligned with the end of the pattern is a mismatch, TNDM will scan forward
looking for Suffix of the pattern. The Number of Examined Characters is less than simplified BNDM and therefore matching is faster [3].

B) SBNDM: Peltola and Tarhio presented, SBNDM, a simplified version of BNDM. SBNDM does not care of prefixes, but shifts the pattern simply past a mismatch. SBNDM is slightly faster than BNDM especially for short patterns.

SBNDM$_q$, which is a revised version of SBNDM with q-grams. The SBNDM algorithm differs from the original algorithm in the main loop where it skips the exploratory of longest prefixes. If $s$ is the current alignment position in the text and $j$ is the number of updates done in the window, then the algorithm simply sets $s + m - j + 1$ to be the start position of the next alignment [7].

C) BNDM$_q$ Algorithm with q-gram: The BNDM$_q$ Algorithms firstly reads g-grams before testing the state vector D. The BNDM with q-grams is also based on BDM algorithm and it can be also simulated by bit parallelism. It uses “m” bits of a computer word $D = d_{m-1} d_{m-2} d_{m-3} \ldots d_0$.

The BDM algorithm moves a window over the text. Each time the window is positioned at a new text position just after pos, it searches backwards the window $t_{pos} + 1 \ldots t_{pos} + m - 1$ using the DAWG automaton, until either “m” transitions are performed or the automaton cannot follow any transition. The algorithm first builds a table B which for each character stores a bit mask $b_{m-1} b_{m-2} b_{m-3} \ldots b_0$. The mask in $B[c]$ has the $i^{th}$ bit 1 if and only if $pi = c$. The state of the search is kept in a machine word $D = d_{m-1} d_{m-2} d_{m-3} \ldots d_0$, where $di$ is set whenever $p0p1...pi$ matches the end of the text read up to now. By using q-gram, we get more simple instruction flow when the q-gram is not present in the pattern, algorithm can be speeded up by a factor of $q$ [8].

IV COMPARATIVE ANALYSIS OF BIT PARALLEL ALGORITHMS

Bit-parallel algorithms are currently the fastest approximate string-matching algorithms for many relevant applications. Bit-parallelism based algorithms are known to be fastest because they use the intrinsic property of bit-parallelism inside the computer architecture. Bit parallelism takes advantage of this fact by packing several variables into a single computer word. BNDM, TNDM, SBNDM are the single pattern string matching algorithms and shift-OR are multi-pattern string matching. The Shift-OR, Shift-AND and BNDM algorithm uses bit parallelism concept and therefore matching process is faster as compared to other character based string matching algorithms. By adding, concept of q-grams in Shift – OR and BNDM algorithms. The BNDM$_q$ Algorithms is faster than BDM. firstly reads g-grams before testing the state vector D, it gives better performance in terms of searching speed and complexity. We analyse that bit parallel algorithms are faster algorithms compare to other character based string matching algorithms because of the concept of bit parallelism obviously bit wise operation such as “&”, “|” are faster than “=” operation. Bit operations in current processors are fast.

V CONCLUSION

This paper was presented, the various bit-parallel string matching algorithms. The shift-OR algorithms is a multiple pattern matching algorithm. It uses Shift OR method for matching the patterns and it is easiest method compare to BNDM algorithm. Shift-OR and BNDM are exact string matching algorithms. Both are based on bit-parallel simulation of a non-deterministic automaton. Shift-Or automaton maintains information about which pattern prefixes match a suffix of the scanned string and accepts when the string ends with the full pattern, while the BNDM automaton maintains information about which factors of the reverse pattern match the scanned string and accepts suffixes of the reverse pattern. Shift-OR makes one, long left-to-right scan, while BNDM makes many right-to-left scans and shifts to the right between scans. Both require an integer
alphabet. When pattern length “m” is at most the machine word size w, BNDM search time complexity is O(mn) in the worst case, O(n(logm)=m), in the average case (which is optimal), and O(n=m) in the best case, while the Shift-Or search time complexity is always O(n) (which is optimal in the worst case). Both need O(m +n ) time for pattern pre-processing. Both algorithms slow down when m > w. We have concluded that, Multiple Pattern Bit Parallel algorithm is an approximate string matching. This algorithms work on all equal size patterns. Bit parallel algorithms are a latest faster algorithm compare to other character based algorithms.

REFERENCES


