JOURNAL of COMPUTER SCIENCE

Back-Navigation String Matching Algorithm (BSMA)

Matthias Daniel[1], Adiela Samuel R[2]. Oriji Ikechi B[3].

[1]*Science/ Computer Science/ Rivers State University, Port Harcourt, P.M.B. 5080, Nigeria*

[2]*Science/ Computer Science/ Rivers State University, Port Harcourt, P.M.B. 5080, Nigeria*

[3]*Science/Computer Science/University of Port Harcourt, Port Harcourt, Nigeria*

Email: matthias.daniel@ust.edu.ng samueladiela3@gmail.com, kchoriji@gmail.com

Abstract

This research took a comparative analysis on string matching algorithms. The study focused on developing an efficient algorithm (Back-navigation String matching algorithm) which will be used for large documents. The algorithm whilst compared to the existing system introduced a pattern of search which was done backwardly, from the last character to the first. The existing system considered more number of shifts which was slow and has a bad character shift. The research aimed at developing an efficient string matching for large documents sorting. The research methodology adopted for this project research is the verification and validation methodology which perform its test in a reverse manner so the software developer can at each stage review its step. The proposed system was executed and produced a result which compared with the existing system was termed efficiency. The system introduces a faster means of searching which starts form the last character to the first. The developed system which is the efficient string matching algorithm was analyzed and displayed a faster means of searching documents and is termed efficient because of its computing speed of 2 milliseconds while the naïve algorithm which ran with the computing speed of 154 milliseconds for a total number of 5000 characters.

Keywords: Algorithm, Automation, Comparative, Methodology, Navigation, String Matching

1. INTRODUCTION

In every organization, information is essential because it is the bedrock for an effective communication. Information is to an organization what blood is to human body. Information is processed through the use of a computer system. There is also urgency that proper means of retrieving or searching for information should diffuse within the organization. Information in this context can be defined as processed data that is meant to be used by an approved user which can be arranged and searched for in a sequential order, alphabetical order, numerical order or chronologically order than ascending or descending form.

Computers are virtually used in all areas of life, it is difficult to see any aspect of the society that has not been affected by some form of computerization and before its advent manual method were used in data processing and also the human brain was put to use. This manual method was error oriented and time wasting and the use of computer in data processing has gone a long way in increasing information credibility. Computers are able to process data quickly, making available information; ensure efficiency and accuracy in searching. It is of paramount demand that no decision be taken without proper consultation of the computer data processing centre. Therefore, any decision taken in any organization without computer analysis is prone to error. A real application of computer is the searching of strings with a given algorithm, often known as automation.

The need for an efficient means of finding several strings or information is also of necessity because when information fails to reach any part of an organization it becomes a constraint, thereby limiting adequate information flow. Information requires qualities such as timeliness, appropriateness and understanding.

The introduction of an efficient string matching algorithm will help in eliminating time-consumption and error, hence the need for the automaton that is, the string matching is necessary and is aimed at the automation of an algorithm for the searching and retrieval of strings or information hence, eliminating time delay constraints and error. The need for string matching can also be applied to the searching of a large amount of information that are stored in linear files which quantity tends to double every month, for this purpose an efficient algorithm is introduced for speed and reliability. It is an important component which helps in an information retrieval system.

String matching is a technique used to discover pattern from the specified input string and which introduces algorithms which are used to find the matches between the pattern that is, the set of characters or strings which is intended to be searched and a specified string which can aid the efficiency and the responsiveness of searching a string or information [1-2]. This matching algorithm introduces algorithm that will help in searching and retrieving patterns or strings in a linear file, with a view of performing a comparative test to determine how efficient one is over the other with respect to searching strings. Each string matching type introduces its own algorithm of which if followed accordingly produces its own output.

String matching has greatly influenced the field of computer science, and also other areas which includes string matching in bioinformatics that is, the application of information technology and computer science to biological problems, string matching in plagiarism detection which it is used in comparing texts and detecting the similarities between them, the basis of the similarities declares whether it is an original work or taken from somewhere, it is also applied in an intrusion detection system where data packets that contains intrusion related keywords are found by applying string matching strategy. String matching is a problem area in computer science which find the positions of text where a given pattern occurs allowing a limited number of errors in the matches.

The very basic and first conventional string matching strategy is the Brute Force Algorithm which considers all possible cases and taking shifts only one place to right even match or mismatch condition occurs anywhere. This algorithm also known as Naïve's approach.

Kleene proved the equivalence between finite automaton and regular expression which could be used to solve the string matching problems [3]. Avoiding numerous comparisons in brute force algorithm, in 1970 Morris and Pratt algorithm was proposed which has linear behavior. This algorithm is based on preprocessing of pattern and character from left to right and if mismatch occurs. It skips some character based on pre-processing phase. In 1977 Knuth Morris Pratt introduced an algorithm having a choice of improvements in Morris and Pratt algorithm. Knuth Morris Pratt has same time complexity as Morris and Pratt algorithm but searching performance found is much better than Morris and Pratt algorithm.

In 1977 Boyer Moore also proposed algorithm which compares character from right to left [4-5]. There are so many multiple pattern string matching algorithms has already been proposed in past decades such as: 1975 Aho-Corasick algorithm was presented by Alfred V. Aho and Magaret J. Corasick, which constructs automata for patterns in pre-processing phase. Commentz Walter proposed an algorithm which has based on Aho-Corasick and Boyer-Moore algorithm, Rabin Karp algorithm is also used to search multiple patterns.

In August, 1990, Daniel Sunday published a paper entitled: "A very fast substring search algorithm" which he actually presented three algorithms all of which he claims to outperform the Boyer-Moore algorithm [6-8]. They are quick search, maximal shift and optimal mismatch.

The Knutt-Morris-Pratt algorithm was conceived in 1970 by Donald Knuth and Vaughan Pratt and independently by James H. Morris. This algorithm presents a means of retrieving documents during one search through the text, to identify all positions where an existing match within the pattern exists [9].

The Boyer Moore string matching algorithm [10] pre-processes the string being searched for in the pattern but not the string searched for in the text. The Boyer Moore algorithm uses information gathered during the pre-process step to skip sections of the text, resulting in a lower constant factor than many other string search algorithms.

Rabin Karp string matching algorithm can be used to look for several different patterns in a string at the same time while still maintaining linear complexity [11]. It is easily done when all the patterns are of the same length and uses the hashing to find any one of a set of pattern strings in text. He Rabin Karp algorithm exploits a hash function to speed up the search

2. MATERIALS AND METHODS

A research methodology is a systematic programming approach of well-defined procedure that should be followed in carrying out a thorough research project or the analysis of the principles of methods, rules and postulates employed by a discipline. The methodology adopted for this research is the v model.

The methodology adopted analyzes the inefficiency in sorting documents by comparing some existing algorithm to determine the algorithm with the highest efficiency (Table 1). From the understanding of the existing algorithms the logical string matching algorithm was considered as the most efficient but a problem was identified which is comparing all character of the Text with the pattern from left to right once the first character of the pattern matches any character in the Text without carrying out a proper evaluation to reduce the possibility of a mismatch [8], hereby reducing efficiency and also the search halts immediately the targeted pattern is achieved.

2.1 Comparative Analysis of the Analyzed Existing String Matching Algorithms

The comparative analysis of the surveyed string matching algorithms with bias to their efficiency, time complexity, preprocessing is summarized in Table 1 below. The efficiency of string matching algorithm varies with some criteria [9]. Though some algorithms are actually more complex than others, some are also faster than others. *Table 1* considers a comparative analysis of the naïve string matching algorithm, approximate string matching algorithm, Rabin Karp string matching algorithm and Boyer Moore string matching algorithm. This would be achieved using some parameters such as the preprocessing time, time complexity, approach and their key ideas.

TABLE 1: Comparative analysis of string matching algorithm

Algorithm	Preprocessing time	Time complexity	Approach	Key ideas	Matching Time
Naïve string matching algorithm	0	0((n-m +1)m)	Linear search	Searching with all alphabet	0((n-m +1)m)
Approximate string matching algorithm	0(m)	0((k+ A)m)/log n	Heuristics based	Search through approximately	0((k+A)m)/log n
Boyer Moore string matching algorithm	0(nm + A)	0(n)	Heuristics based	Bad character and good suffix heuristics to determine the shift distance	0(n)
Rabin Karp string matching algorithm	0(m)	0((n-m +1)m)	Hashing based	Compare the text and pattern from their hash function	0((n-m +1)m)

The software design using this methodology is an enhancement of the bottleneck of the existing which introduces a different dimension of search in order to enhance the efficiency of the algorithm. After the conceptualization of a design, a module which includes just the searching of the text alone was tested, after a successful test of the module it was ascertain that the software design of the Back-Navigation String Matching Algorithm can proceed which eventually lead to the codification phase.

After the codification phase for the search, the software was also tested and after a success it proceeds to the codification of the sorting phase and also tested to verify that the search and the sorting codes were a success, after that the code was integrated an integration test was performed.

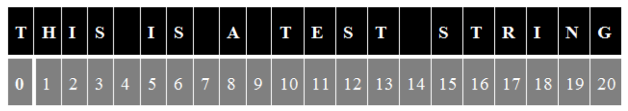
2.2 Design of the Proposed System

The proposed system is a Back Navigation String Matching Algorithm for large document sorting. Several pattern matching algorithms have been developed with a view to enhance the Searching processes by minimizing the number of comparisons performed. The design presents a new method for string matching in large document. The basic idea of the BSMA is to compare the characters of the Pattern to the characters of the Text if and only if the first and last character of the Pattern corresponds with the computed first character and matched character found in the Text respectively.

The BSMA for large document sorting produces the minimum shift and frequency of comparisons during the search process. If there are successive characters which are not in the pattern string they are ignored. For example, if the pattern is "*TEST*" and the Text string is "*THIS IS A TEST STRING*".

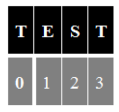
BSMA NOTATIONS

T = Text String
P = Pattern
n = Length of the Pattern
m = Position of the second character of the Pattern (if any).
i = Position of the last character in the Text String.



T	H	I	S		I	S		A		T	E	S	T		S	T	R	I	N	G
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

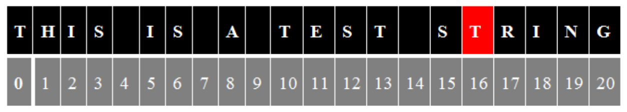
PATTERN



T	E	S	T
0	1	2	3

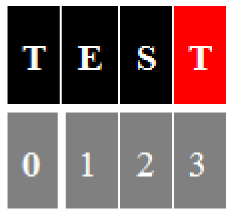
The last character of the pattern ("T" in position 3 of the pattern) is used to match characters of the Text String from the Right to left ("G" in position 20 to "T" in position 0 of the Text String), If a match is found (as seen in position 16 of the Text String).

TEXT STRING



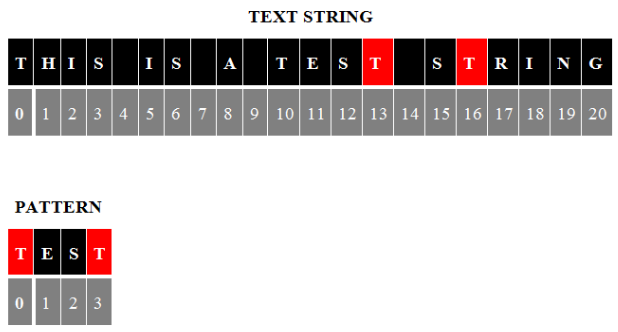
T	H	I	S		I	S		A		T	E	S	T		S	T	R	I	N	G
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

PATTERN

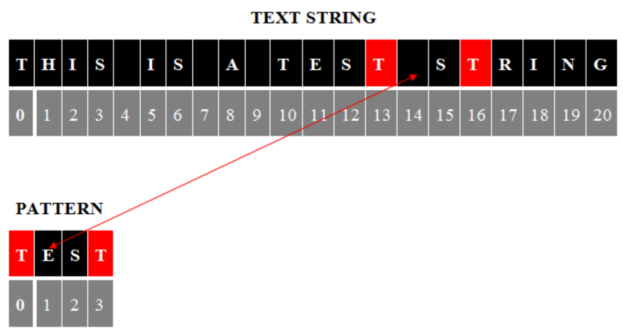


T	E	S	T
0	1	2	3

Then an evaluation is carried out before the respective characters of the pattern are been compared. The BSMA evaluation checks if the character at position zero "0" of the pattern ("T" at position 0 of the pattern) matches the character at position (i - (n-m)) i.e T[(i - (n-m)] (From the Example T[16-(4-1)] = "T in position 13 of the Text String".



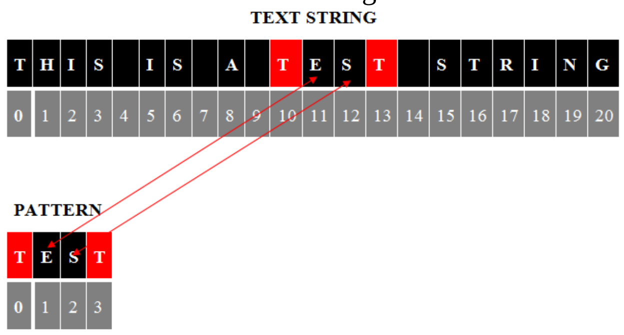
Since the character at Position "13" of the Text String matches the Character in position "0" of the pattern as seen in the example, hence the BSMA evaluation is successful as there's a high possibility that the pattern might correspond to the characters in the marked interval (i.e [13, 16] of the Text String). Since the BSMA evaluation produced a positive output hence the BSMA starts a character by character match within the marked interval of the Text String from left to Right.



But again as seen in the above example if the character at position "1" of the Pattern does not match the character at position "14" of the Text String then a mismatch occurs (i.e the pattern does not exist in the marked portion of the Text String therefore no need of searching further).

When a mismatch occurs during the BSMA comparison process, the value of "i" which holds the current position in the Text String then the value of "i" is decremented by 1 until the value of "i" equates to Zero "0".

Once the value of "i" is decremented the next occurrence of the last character of the pattern i.e "T" is been searched in the Text String until all occurrences of the pattern are located in the Text String.



From the example above since the characters within the marked portion matches the pattern then the position (i-(n-m)) i.e "position 10" is reported.

2.3 Algorithm of the Proposed System

The algorithm for the proposed system is deduced from the equations below;
Given that pattern characters P[0…(n-1)] matches text characters T[i…0] the BSMA Evaluation finds the closed intervals B[(i-(n-m)),i] such that the characters count of B[(i-(n-m)),i]= n assuming that P[0…(n-1)]= T[(i-(n-m)),i] for all characters within the LHS and RHS of the above expression.
Where:

i = the position of the algorithm in the Text String "T"
n = The Length of the Pattern "P" i.e Length(P)
B [lbB,upB]= The bounded interval found within the Text String "T"
LbB = The lower Bound of the interval B
upB = The upper bound of the interval B
m = Denotes the second character of the pattern P i.e m=1
n-1 = Position of the last Character of the Pattern P

B[(i-(n-m)),i]= n …………………………… Equ (1)
Let the LowerBound of the interval B = *lbB* and the UpperBound of the interval B = *upB*
Therefore:
$\Rightarrow$lbB= i-(n-m)
$\Rightarrow$upB= i
Hence Equ(1) can be written as:

B[lbB,upB]= n ………………………………….. Equ (2)
Since (n-1) denotes the last character in the pattern P[0…(n-1)] therefore From Equ(2)

upB-lbB= n-1 …………………….……….Equ (3)
Substituting the values of lbB and upB from Equ(1) into Equ(3)

i-(i-(n-m))= n-1 ……………………….…..…Equ (4)
Since the value of m=1 which denotes the second character of the pattern thus Equ (4) can be written as:

i-(i-(n-1))= n-1 ………………..…………..…Equ (5)
From Equ(5):
$\Rightarrow$ i-(i-(n-1))= n-1
$\Rightarrow$ i-i+(n-1)= n-1
$\Rightarrow$ n-1= n-1

Since the LHS = RHS this shows that a match is found within the interval
B[lbB,upB]$\in$ T[i… 0] and B[lbB,upB]= P[0…(n-1)]
Since the BSMA Evaluation holds in Equ(5) hence the lbB i.e i-(n-m) of the interval B[lbB,upB] is reported as the position where match occurred.

2.4 Pseudo code of the Proposed System

The Back-Navigation String Matching algorithm is given in pseudo code below as the procedure BSMA.

```
BSMA(T,P)
n ←Length(P)
i ←(Length(T)-1)
While i ≥0
    do if T[i]=P[n-1]  and (i-(n-1) )≥0
        m ←1
        while m <(n-1)
                do if T[i-(n-m)] ≠P[m-1]
                then "Break Exit Loop"
            then m ←m+1
            if m=n
                then return(i-(n-1) )
    then i←(i+1)
```

Where:

T = Text String

P = Pattern

n = Length of the Pattern

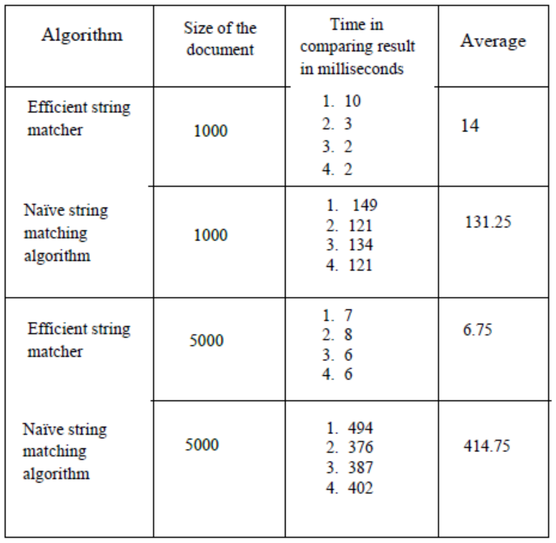
m = Position of the second character of the Pattern (if any with regards to single character patterns).

i = Position of the last character in the Text String

3. RESULTS AND DISCUSSION

The Back-navigation string matching algorithm was compared with the existing system in milliseconds using four different search results and dividing the result with the total number of times being searched. The table below shows the compared results.

TABLE 3: Time comparison of the proposed and existing system



Algorithm	Size of the document	Time in comparing result in milliseconds	Average
Efficient string matcher	1000	1. 10 2. 3 3. 2 4. 2	14
Naïve string matching algorithm	1000	1. 149 2. 121 3. 134 4. 121	131.25
Efficient string matcher	5000	1. 7 2. 8 3. 6 4. 6	6.75
Naïve string matching algorithm	5000	1. 494 2. 376 3. 387 4. 402	414.75

From Table 3 above, documents of different sizes were analyzed using the naïve and efficient string matching algorithm (proposed algorithm). The analysis involved executing both algorithms four (4) consecutive times each and an overall average time was obtained. For the average time obtained from both algorithms in the cases of documents of different sizes, it was observed that the efficient string matching algorithm (proposed algorithm) had a lesser running time due to the reduced number of comparison it uses. Therefore, the efficient string matching algorithm (proposed algorithm) is faster compared to the naïve string matching algorithm.

4. CONCLUSION AND FUTURE WORK

The study presented and analyzed detailed and an experimental result obtained from several string matching algorithms. The study examined different string matching algorithm and concentrated on developing a Back-navigation String Matching Algorithm (BSMA) which can be most practically suited for large document. The developed system which is the Back navigation string matching algorithm was analyzed and displayed a faster means of searching documents and is termed efficient because of its computing speed of 2 milliseconds while the naïve algorithm which ran with the computing speed of 154 milliseconds for a total number of 5000 characters. From the experimental results the efficient algorithm performs efficiently amongst.

5. REFERENCES

[1] Matthias D., Memoh Hosea C., and Taylor Onate E.. Neural Network: Application to Pattern Recognition Considering Training Time. International Journal of Information Technology & Systems, Volume 1; Number 2: ISSN: 2277-9825, 2013.

[2] Denga, G.I. Time space optimal string matching. Journal of Computer and System Science 45(5) 123-196, 1956.

[3] Kleene, M. Online serial exact string searching in Pattern Matching Algorithms Apostolico and Galiled Oxford University Press, 1956.

[4] Moore, B. Boyer-Moore approach to approximate string matching. 2nd Scand. Workshop on Algorithm Theory (SWAT), SLNCS 447, pp. 348-359, 1977.

[5] Donald, N. A variation on the Boyer Moore algorithm, theoretical Computer Science, Futura Publishing Company, Inc. New York, (1977).

[6] Sunday, P. Parallel String Matching Algorithm D.P Publication Shepherds Bush Green, London, 1990.

[7] Donald, N. The complexity of pattern matching for a random string. SIAM Journal on Computing, 4(34), 1970.

[8] Moore, J. S. A fast string searching algorithm Communications of the ACM, 20(10):761–772, 1977.

[9] Holmes, H. P. Fast string matching with k differences, journal of computer sciences and system science, 20 (5) Preliminary version FOCS'8537:63-78, 1949.

[10] Clifford, F.I. (1997). A Very Fast Substring Search Algorithm. Communications ACM 33,132-142.

[11] Karp, J. Fastest pattern matching in strings, Journal of Algorithms 4(45), 12-56, 1987.

[12] Ali, A. J. Advanced String Searching Technique, Science Research Association Inc. Chicago, Henley-no –Thomas Sydney, Toronto, A Maxwell Program Publishing Company. U.S.A, 2010.

[13] Thompson, P. Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development, 249-260, 1968.