IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727
PP 32-35
www.iosrjournals.org

Study of String Matching Algorithm

Prof. I.V.Srinivas[1,] Moez Samnani, Mohammed Shafaat Shaikh[2]
[1]*(MCA, K.J SIMSR, India)*
[2]*(MCA, K.J SIMSR, India)*

Abstract: *In every text-editing application like notepad, MS Word, MS Excel, etc.there is a needof mechanism to find a particular pattern. The main objective of string or pattern searchingis to searchparticular pattern for position in a large body of text (e.g.from a book, a paragraph, a sentence, etc.). The purpose is to find occurrence of a text within another text. For example when we need to find some text in text editor it is a tough task to find that word or text manually. To make our task simple we simply press shortcut key Ctrl+F,which generates a pop up window where we can enter text which we need to search and it finds the occurrences of that particular text in the entire document. If matches found then it will highlight it all the occurrence of string which we are looking for otherwise it will display no matches found if there is zero occurrence of string.*
Keywords: *String matching algorithms, Algorithm, pattern matching, exact pattern matching, searching*

I. Introduction

The motivation behind this paper is to study about various algorithms for the String matching issue. These algorithms are utilized for attempting to find one, a few or all events of a characterized pattern in a larger text. The string matching problem has various applications in different regions. Initially, an adjusted and productive algorithm of this issue can help to upgrade the responsiveness of a text editing program. Different applications in information technology incorporates web search engines, spam filters, natural language processing, computational science (enquiry of specific example in DNA arrangement), and feature detection in digital images.

In general, pattern matching algorithm is used in text editing to find particular pattern in text editor, data compression, DNA sequence matching, spell checking, computer virus, signature matching, dictionary-based language translation, World Wide Web search engines and other computer application system. String matching algorithm is used to find matches between specified string or text T and pattern P, where T is longer or equal to P.It is either precisely matched or partially matched with the example, in view of this, string matching algorithms are classified as: Exact string matching algorithm and Approximate string matching algorithm.

Exact string matching algorithms are worried about the quantity of events of the pattern into a given text, while Approximate string matching algorithms are worried about the likeness rate between the pattern and the text or any part of the text.

String matching algorithm is additionally used to discover a particular pattern in DNA sequence. It additionally assumes the imperative part in computational science.

For string matching problem, we assume that the text is in array T [1...n] and n is length of text array and pattern is in array [1...m]and m is length of pattern (m<=n).

Text T

d	C	b	A	a	e	b	a	C	C	b	e	a

S7 $\longrightarrow$

a	C	C	b

Pattern P

An example of the string-matching problem, where we need to discover all events of the pattern P = accb in the text T = dcbaaebaccbea.Here S=7 implies that from 7 shifts we found the pattern.

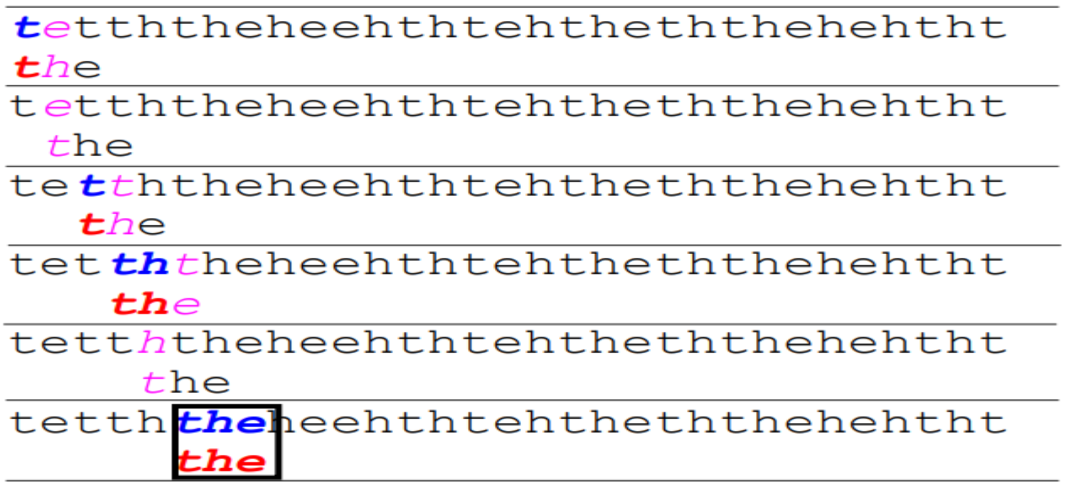
II. Algorithms

Various algorithms of string matching are[7]:

Algorithm	Time (Preprocessing)	Time (Matching)
Naïve Approach	0	O((n-m+1)*m)
KMP (Knuth-Morris-Pratt)	O(m)	O(n)
Rabin Karp	O(m)	O((n-m+1)*m)
Finite automata	O(m)	O(n)
Boyer-Moore	O(m+n)	O(mn)
Brute Force	O(mn)-Worst case	m (n-m+1)- Total number of comparisons

<u>Brute Force</u>

<u>Example:</u>


tetththeheehthtehtheththehehtht
the
tetththeheehthtehtheththehehtht
the
tetththeheehthtehtheththehehtht
the
tetththeheehthtehtheththehehtht
the
tetththeheehthtehtheththehehtht
the
tetththeheehthtehtheththehehtht
the

<u>RABIN KARP</u>

Richard M. Karp and Michael O. Rabin (1987) introduced RABIN KARP algorithm for pattern searching algorithm.

Rabin Karp uses O(n) processing time and its most pessimistic scenario running time is O((n-m+1)m). Its average running time is better.

A reasonable use of Rabin Karp algorithm is detecting plagiarism. Given source material, the algorithm can quickly look through a paper for instances of sentences from the source material, overlooking details such as case and punctuation.

The Rabin–Karp algorithm searches the pattern in text by using hash function. Hash function converts string into hash value using hash function. On the off chance that two strings are equivalent then its hash values are also equal. In KMP or naive approach substrings are compared, but in Rabin Karp algorithm string matching is decreased (almost) to register the hash estimation of the search pattern and after that searching for substrings of the input string with that hash value.

Theproblems with this approach are: So few hash values, some contrasting strings will have the similar hash value. On the off chance that the hash values match, the pattern and the substring may be different.

<u>NAIVE APPROACH</u>

Naïve approach is traditional approach it keeps on testing the entire possible pattern P [1…a] in text T [1...b].

Naïve approach algorithm discovers all the possible movement using loop to check the condition P[1..a]=T[C+1…C+a].

<u>KMP ALGORITHM</u>

KMP algorithm is first linear time searching algorithm which search particular pattern in text. This algorithm was discovered by Donald Knuth, Vaughan Pratt and James Morris in 1970 and distributed it in 1977. Running Time of KMP algorithm is O(m+n) in worst case.

<u>Working of KMP Algorithm</u>

Pattern “abxabcabcaby”

1)

a	b	X	a	B	c	a	b	C	a	b	y

a	b	C	a	B	y						

2)

a	b	X	a	B	c	a	b	C	a	b	y
		A	b	C	a	b	y				

3)

a	b	X	a	B	c	a	b	C	a	b	y
			a	B	c	a	b	Y			

4)

a	b	X	a	B	c	a	b	C	a	b	y
						a	b	C	a	b	y

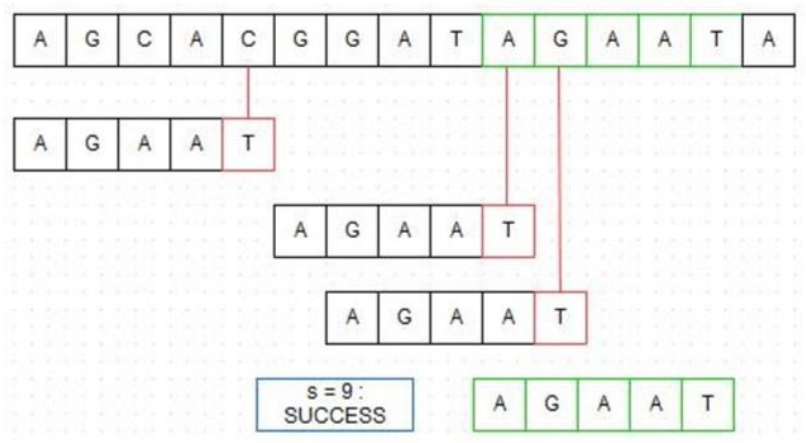
KMP Algorithm utilizes Partial Match Table also known as Failure function. This finds proper Suffix which is also proper Prefix of string.When mismatch occursin finding pattern I text then this partial match table is used to know where to start matching based on partial table.

Partial Match Table for pattern “abcaby”

Pattern	A	B	C	a	B	y
Index	0	1	2	3	4	5
Value	0	0	0	1	2	0

Boyce-Moore Algorithm

The essential thought behind this algorithm is that the equality is carried out from right to left. Due to this characteristic the algorithm can skip large number of characters than different algorithms. For instance if the primary character match of the text does not contains in the pattern P[0…m-1], then this algorithm skips m characters immediately. This algorithm additionally preprocesses the pattern to acquire a table that contains data to skip characters for each character of the pattern. Boyce-Moore algorithm likewise utilizes an additional table in light of the letter sets which contains the same number of entries as number of characters in the letter set.



Example of Boyce-Moore Algorithm [Source:Algorithms for String matching Marc GOU]

III. Conclusion

In this paper, various types of string matching algorithmswere examined with organic arrangementsfor example, DNA andProteins. From the contemplating, it is broke down that KMP algorithm is moderately less demanding to actualize on the grounds that it moves in forward direction onlyin the input sequence and it needs additional space.

RabinKarp algorithm is utilized to identifythe plagiarism and it requiresextra space for matching. Brute Force algorithm does notrequire preprocessing of the content, the issue with Brute Force algorithm is that it is moderate, it once in a while delivers effective outcome. The Boyer Moore algorithm is to a great degree quick on expansive sequences, it avoids bunches of pointless comparisons byfundamentallysearching pattern with respect to the content; its best case runningcomplexity is sub linear.

The outcomes demonstrate that Boyer-Moore is the best algorithm to tackle the string matching problem in regular cases, and Rabin-Karp is a decent option for some particular cases, for example when the pattern and the text are very small.

References

[1]. S.Viswanadha Raju and A.Vinaya Babu, “Efficient Parallel Pattern Matching Using Partition Method”, Proc. of PDCAT 2006, IEEE Computer society, 2006.

[2]. S.Viswanadha Raju and A.Vinaya Babu,“ Backend Engine For Parallel String Matching Using Boolean Matrix”, Proc.of PARALEC 2006, IEEE Computer society, 2006.

[3]. M M Rajashekharaiah, K, Madhubabu, Ch, SOmalaraju and Viswanadha raju. Parallel String Matching Algorithm Using Grid. International Journal of Distributed and Parallel Systems. 32012.3303.

[4]. Michael T. Goodrich and Roberto Tamassia, 2002. Algorithm Design. John Wiley and Sons, Inc.

[5]. P, Pandiselvam T, Marimuthu Raj and Lawrance A Comparative Study on String Matching Algorithm of Biological Sequences, https://www.researchgate.net/publication/259953827 , 2014

[6]. https://www.google.co.in/url?sa=t&source=web&rct=j&url=https://en.m.wikipedia.org/wiki/String_searching_algorithm&ved=0ahUKEwjxjpHo-O_XAhWIlJQKHUaYCk0QFghVMAQ&usg=AOvVaw2CFkFnWJbtdYwWZ63UL0eA

[7]. Searching for Patterns | Set 2 (KMP Algorithm) http://www.geeksforgeeks.org/searching-for-patterns-set-2-kmp-algorithm/