COMPLAINT REPORTING AND PATTERN SEARCHING USING KMP ALGORITHM

Anchana Ashok [1] , Sumy Roslin Joseph [2] , Anjana Vinod [3] and Dr. Juby Mathew [4]

[123] PG Scholars of Amal Jyothi College of Engineering, Kanjirappally, Kerala.
[4] Associate Professor of Amal Jyothi College of Engineering, Kanjirappally, Kerala.

Abstract :The string-searching algorithms, also called string-matching algorithms, are a significant class of string algorithms. The objective of these algorithms is to discover a place where one or a few strings (likewise called patterns) are found inside a larger string or text. Nowadays there are numerous algorithms accessible for string looking with changed seeking speed. The larger the move of the pattern with respect to the string if there should be an occurrence of pattern and string characters' confound is, the higher is the algorithm running speed. This paper offers an algorithm, which has been created based on the Kunth-Morris-Pratt string looking algorithm. This algorithm depends on different fundamental standards of pattern matching. Knuth-Morris-Pratt algorithm depends on forwarding pattern and the pattern is completely contrasted and the selected text window (STW) of the text string and show the beginning index position.

Keywords: KMP algorithm, searching, pattern matching.

I. INTRODUCTION

In this day and age, we need a quick algorithm with least mistakes for tackling the problems. Pattern matching strategy is a constant issue. There exist various kinds of information in web application issues, for instance, content records, picture documents, sound documents, and video documents seeking. For searching various kinds of information web crawler is required and every search algorithm is utilized by every search engine for dealing with various sorts of data. This paper gives an altered adaptation of the KMP algorithm for text matchingPandey, etal[1]. This is a territory of expanding research enthusiasm for the segments of database and data retrieval. The time complexity of the KMP algorithm is O(n) in the worst case.

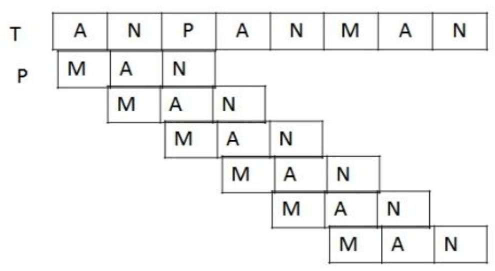
Centralized Public Grievance Reporting And Monitoring System (CPGRAMS) is an online web-empowered stage dependent on web innovation which essentially intends to empower accommodation of complaints by the citizens from anyplace and whenever (24x7) premise to Departments who investigate and make a move for the quick and ideal revealing of these complaints. This Portal went for furnishing the natives with a stage for revealing of their complaints. On the off chance that you have any complaint against any Government association/others in the state, you may hold up your complaint here which will go to the State Government worried for prom.pt review. Absence of paper developments improves the speed which was never conceived in manual mode by any. stretch of the imagination.

II. LITERATURE REVIEW

A. Single pattern string matching algorithms:

1) *Naive string matching algorithm*:Rasool, etal[4], it has no pre-preparing stage, needs consistent additional space.It is otherwise called the Brute Force algorithm. It generally moves the window by precisely one position to the right. It requires 2n expected text characters' comparisons. It discovers every single legitimate move using a loop that checks the condition P[1....m]=T[s+1........s+m] for every one of then-m+1 potential estimations of s. Consider the example:

T=ANPANMAN
P=MAN



igure
1: Naive String Matching Example

Theworst case running time is O((n-m+1)m). The running time of Naive String Matching algorithm is equivalent to its coordinating time since there is no preprocessing.

2) *Rabin Karp String Matching Algorithm*:Shabaz,etal [2] utilizes the hashing function in his method. It works in two stages i.e. pre-processing stage (time complexity O(m)),matching stage(time complexity average O(n+m), worst O((n-m+1) m)). *Rabin Karp* algorithm is used to discover a numeric pattern Pat from a given text T.Firstly it separates pattern Pat with a predefined prime number q to calculate the remainder of the pattern. On the off chance, it takes the first m characters from text T at first move s to compute the remainder of m characters from text T. If the remainder of the pattern and remainder of the text T are equal, only then a comparison between them is needed. We will repeat the process for the next set of characters from the text for all possible shifts which are from s=0 to n-m. Along these lines, as indicated by this, two numbers n1 and n2 must be equivalent if REM (n1/q) = REM(n2/q). After division, there are three cases:-

Cases	Condition	Result
Successful hit	REM(n1)=REM(n2)	n1=n2
Spurious hit	REM(n1)=REM(n2)	n1≠n2
Unsuccessful hit	REM(n1) ≠REM(n2)	n1≠n2

Table I: Three cases

3) *Boyer-Moore String Matching Algorithm*:Shabaz, etal[2],the Boyer Moore algorithm (BM) was put forward by R.S.Boyer and J.C.Moore in 1977. The BM algorithm filters the characters of the pattern from right to left start with the furthest right one and plays out the examinations from right to left. If there should be an occurrence of a confound (or a total match of the entire pattern) it utilizes two pre-computed functions to move the window to the right, that is a good-suffix shift and the bad-character shift.It works in two stages: Preprocessing stage in O(M+|∑ |) time complexity, Matching stage in Ω(n/m), O(n) time complexity. There are 3n text character comparisons in the worst case while looking for a non-discontinuous pattern.

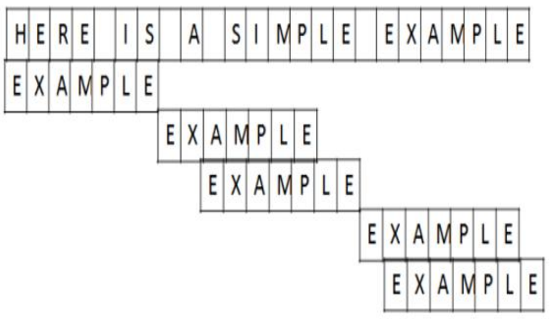


Figure 2: Boyer Moore String Example

III. METHODOLOGY

KMP algorithm is utilized to discover a "Pattern" in a "Text". Pandey, etal[1]compares character by character from left to right. However, at whatever point a mismatch happens, it utilizes a pre-processed table called "Prefix Table" to skip characters examination while matching.A few times prefix table is otherwise called LPS Table. Here LPS means "Longest Proper Prefix which is likewise Suffix.

Steps for Creating LPS Table (Prefix Table)

Step 1: Declare a 1-d array (LPS[size]) with the size equal to the length of the Pattern.

Step 2: Define variables **i & j**. Set i = 0,j = 1 and LPS[0] = 0.

Step3 : Compare Pattern[i] and Pattern[j].

Step 4: If a match occursthen set **LPS[j]= i+1** and increment both i & j values by one. Goto to Step 3.

Step 5: If both are not matched then check the value of the variable 'i'. If it is '0' then set **LPS[j] = 0** and increment 'j' by one, if it is not '0' then assign **i=LPS[i-1]**. Goto Step 3.

Step 6- Repeat the above steps until all the values of LPS[] are filled.

We utilize the LPS table to choose what number of characters are to be skipped for comparison when a mismatch has found. At the point when a mismatch happens, check the LPS estimation of the previous character of the mismatched character in the pattern. On the off chance that it is '0' at that point begin looking at the first character of the pattern with the next character to the mismatched character in the text. On the off chance that it isn't '0' at that point begin looking at the character which is at an index value equivalent to the LPS value of the previous character to the crisscrossed character in pattern with the mismatched character in the Text.

KMP (Knuth Morris Pratt) ALGORITHM

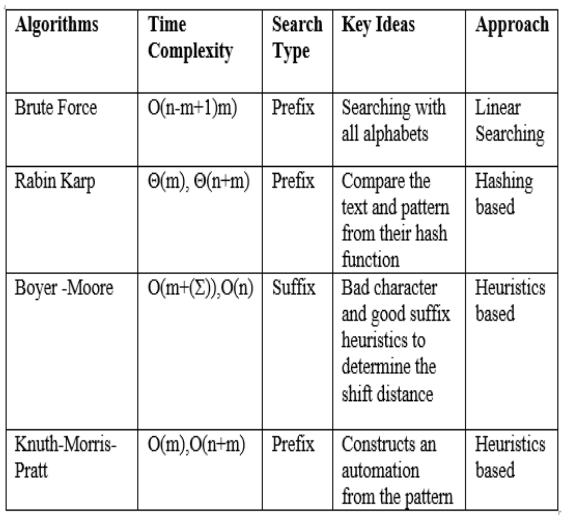
```
KMPsearch(P,T)
1. m –> Pattern
2. n–>Text
3. int c[]
4. j = 0
5. COMPUTELPSARRAY(PAT ,M,C)
6. i = 0
7. while(i < n)
 8. if(pat[j] == txt[i])
 9. j + +
10.i + +
11. if(j == m)
12. printf("found pattern at index %d",i − j)
13. j = c[j − 1]
14. else if(i < n&& pat[j]! = txt[i])
15. if(j! = 0)
16. j = c[j − 1];
17. else
18. i = i + 1;
```

COMPUTELPSARRAY (PAT,M,C)

```
1.  m = p.length
2.  $let\ [1…m]\ beanewarray$
3.  $j$ = 0, = $j$ + 1,$c$[0] = 0
4. $whil(i<m − 1)$
{
 while(p[i]! = p[j])
 {
c[i] = 0;
 i + +;
 }
if(p[i] == p[j])
{
c[i] = j + 1;
i + +;
 j + +;
}
 }
 5.  while(i == m − 1)
{
int k;
while(p[i]! = p[j])
{
k = c[i − 1];
j = c[k];
 }
if(p[i] == p[j])
{
j = j + 1;
c[i] = j;
}
}
6.  return c;
```

IV. RESULT

Pattern searching is an important problem in the area of computer science. When we do look for a string in a word document or program or database, to get the search results searching algorithms are utilized.The implementation of this algorithm in a portal like CPGRAMS has a large use in case of time.The execution time of getting the result of a query from a database is greater when compared to the KMP algorithm.



Algorithms	Time Complexity	Search Type	Key Ideas	Approach
Brute Force	O(n-m+1)m)	Prefix	Searching with all alphabets	Linear Searching
Rabin Karp	Θ(m), Θ(n+m)	Prefix	Compare the text and pattern from their hash function	Hashing based
Boyer -Moore	O(m+(Σ)),O(n)	Suffix	Bad character and good suffix heuristics to determine the shift distance	Heuristics based
Knuth-Morris-Pratt	O(m),O(n+m)	Prefix	Constructs an automation from the pattern	Heuristics based

Table II: Comparative Analysis

V. CONCLUSION

String matching algorithms have a major role in this CPGRAMS Portal. Many persons are functioning on software and hardware levels to make arrangements searching quicker. By approximate best algorithms in various algorithms in various claims is determined. The modified algorithms give compact complexity and also compact calculation time. The procedure allotted to various requests may not be the best optimum algorithm but better than the all-purpose algorithms. It has been well-known that many applications use Boyer Moore, KMP algorithm for their operational functionality and other uses the basics of these algorithms for their functionalities as the KMP algorithm has less time complexity.

VI. FUTURE SCOPE

In the real world KMP algorithm is used in those applications where pattern matching is done in long strings, whose symbols are taken from an alphabet with little cardinality. A relevant example is the DNA alphabet, which consists of only 4 symbols (A, C, G, T).Spell Checkers, Intrusion Detection System, Search Engines, Plagiarism Detection, Bioinformatics, Digital Forensics, and Information Retrieval Systems, etc are some of its applications.

REFERENCES

[1] Garima Pandey,Nitin Arora, Mamta Martolia. “A Novel String Matching Algorithm and Comparison with KMP Algorithm” in International Journal of Computer Applications (0975 – 8887) Volume 179 – No.3, December 2017.

[2] *Er. Mohammad Shabaz and Er. Neha Kumari."Vance-Rabin Karp Algorithm For String Matching”in International Journal of Current Research Vol. 9, Issue, 09, pp.57572-57574, September 2017.

[3]Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition. The MIT Press, 2009.
[4] Akhtar Rasool, Amrita Tiwari, Gunjan Singla, Nilay Khare. String Matching Methodologies: A Comparative Analysis, Vol. 3 (2), 2012.