World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:6, 2008

RB-Matcher: String Matching Technique

Rajender Singh Chillar, Barjesh Kochar

***Abstract*—**All Text processing systems allow their users to search a pattern of string from a given text. String matching is fundamental to database and text processing applications. Every text editor must contain a mechanism to search the current document for arbitrary strings. Spelling checkers scan an input text for words in the dictionary and reject any strings that do not match. We store our information in data bases so that later on we can retrieve the same and this retrieval can be done by using various string matching algorithms. This paper is describing a new string matching algorithm for various applications. A new algorithm has been designed with the help of Rabin Karp Matcher, to improve string matching process.

***Keywords*—**Algorithm, Complexity, Matching-patterns, Pattern, Rabin-Karp, String, text-processing.

I. Introduction

STRING-MATCHING is a technique to find out pattern from given text. Let Σ be an *alphabet*, a nonempty finite set. Elements of Σ are called *symbols* or *characters*. A string (or word) over Σ is any finite sequence of characters from Σ. For example, if Σ = {a,b}, then *abab* is a string over Σ. String-matching is a very important topic in the domain of text processing.[2] String-matching consists in finding one, or more generally, all the occurrences of a string (more generally called a *pattern*) in a *text*. The pattern is denoted by P=P [0 ... *m*-1]; its length is equal to *m*. The text is denoted by T=T [0 ... *n*-1]; its length is equal to *n*. Both strings are building over a finite set of character called an alphabet denoted by Σ. [3]

RABIN KARP matcher is one of the most effective string matching algorithms. To find a numeric pattern 'P' from a given text 'T'. It firstly divides the pattern with a predefined prime number 'q' to calculate the remainder of pattern P. Then it takes the first m characters (where m is the length of pattern P) from text T at first shift s to compute remainder of m characters from text T. If the remainder of the Pattern and the remainder of the text T are equal only then we compare the text with the pattern otherwise there is no need for the comparison.[1] The reason is that if the remainders of the two numbers are not equal then these numbers cannot be equal in any case. We will repeat the process for next set of characters from text for all the possible shifts which are from s=0 to n-m(where n denotes the length of text and m denotes the length of P). So according to this two number n1 and n2 can only be equal if

$$\text{REM}(n1/q) = \text{REM}(n2/q) \quad [1]$$

After division we will be having three cases :-

Case 1:
Successful hit: - In this case if REM (n1) = REM(n2) and also characters of n1 matches with characters of n2.
Case 2:
Spurious hit: - In this case REM (n1) = REM (n2) but characters of n1 are not equal to characters of n2.
Case 3:
If REM (n1) is not equal to REM (n2), then no need to compare n1 and n2.

Ex-
For a given text T, pattern P and prime number q

T=234567899797797976534356678886756456890
97554534343424545475655454

P=667888

q=11

so to find out this pattern from the given text T we will take equal number of characters from text as in pattern and divide these characters with predefined number q and also divide the pattern with the same predefined number q. Now compare their remainders to decide whether to compare the text with pattern or not.
Rem (Text) =234567/11=3
Rem (Pattern) =667888/11=1
As both the remainders are not equal so there is no need to compare text with pattern. Now move on to next set of characters from text and repeat the procedure. [1]. If remainders match then only we compare the part of text to the pattern otherwise there is no need to perform the comparison. We will maintain three variables Successful Hit, Spurious Hit and Unsuccessful Hit.

Rabin Karp Matcher Algorithm

```
Rabin_Matcher (T,P,d,q)
{
    n =Length (T)
  m= Length (P)
  t0=0
  p=0
  h=d^(m-1) mod q
```

Dr. Rajender Singh Chillar is Reader, Formal HOD (CS) with Maharishi Dayanand University, India (phone: +919416277507; e-mail: chillar01@rediffmail.com).

Barjesh Kochar is working as HOD-IT with GNIM, New Delhi, India (phone:+919212505801; e-mail: barjeshkochar@gmail.com)

Garima Singh (Jr. Author) is pursuing B.Tech studying in GGSIPU University, New Delhi, India (e-mail: garima_20_04@yahoo.co.in).

Kanwaldeep Singh (Jr. Author) is pursuing B.Tech studying in GGSIPU University, New Delhi, India (e-mail: kawal_deep87@yahoo.co.in).

```
For i=1 to m
{  p = (dp+P[i]) mod q
   t0 =(d t0 + T[i] ) mod q
}
For s =0  to n-m
{  If ts=p
   {  then
      {  if P[1….m]=T[s+1…….s+m]
            then print pattern matches at shift 's'
      }
   }
    if s<= n-m
      ts+1= (d(ts-h*T[s+1]) + T[s+1+m] ) mod q
}
}[1]
```

II. Improved String Matching Algorithm

A. Theory

As we can see, spurious hit is an extra burden on algorithm which increases its time complexity because we have to compare text with pattern and won't be able to get pattern at that shift so to avoid this extra matching, RB_Matcher says that along with remainders compare the quotients also.

REM(n1/q)=REM(n2/q) and
QUOTIENT (n1/q)= QUOTIENT (n2/q)

So, according to this method along with calculation of remainder, we will also find out quotient and if both remainder and quotient of text matches with pattern then it is successful hit otherwise it is an unsuccessful hit and then there is no need to compare it. That means there is no extra computation of spurious hits if both are same then pattern found else pattern not found. Please, leave two blank lines between successive sections as here.

B. Algorithms

The modified algorithm is as follows:-

```
RB_ Matcher (T,P,d,q)
{
n =Length (T)
m= Length (P)
t0=0
p=0
Q=0
pq=0
h=dm-1mod q
For i=1 to m
{
   p = (dp+P[i]) mod q
   t0 =(d t0 + T[i] ) mod q
}
pq= P[1…..m] DIV q
For s =0 to n-m
{
   Q=T{s+1…….s+ m] DIV q
   If (ts = p and Q = pq)
   {
      then print pattern matches at shift 's'
   }
   if s<= n-m
      ts+1= ( d(ts-h*T[s+1]) + T[s+1+m] ) mod q
}
} [3]
```

III. Improvements

Rabin Karp matcher algorithm was computing remainder on the basis of which it was conducting whether the pattern has been found in the text or not. So there was an extra computation when processing for the spurious hits. But in the case of Modified RB matcher there is no chance of spurious hits because it always gives one solution i.e. in case of successful hits.

A. Comparisons in terms of Time Complexity

To compare the previous work with the new one we applied both these algorithms on a lot of Fictitious Data & the results shows that Modified RB_matcher algorithm is having less Time complexity as compared to Rabin – Karp Matcher. The worst case time complexity of Rabin-Karp matcher is O (n-m+1) m). While the worst case time complexity of Modified RB Matcher is O (nm+ 1) (This Time complexity can be further improved if q=m) where n denotes the total characters in Text T and m denotes total characters in Pattern P.[9]

Graphs:-

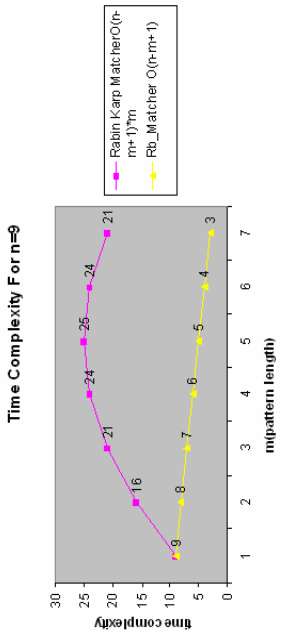

Fig. 1 shows Time complexities of Rabin Karp and RB Matcher if n=9

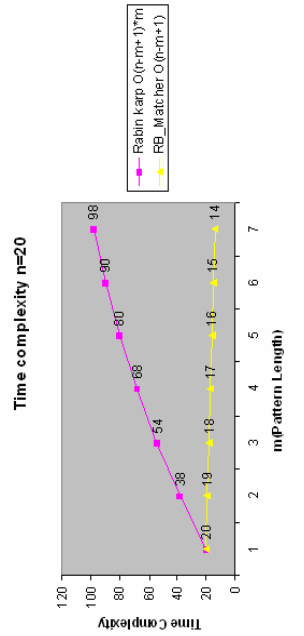

Fig. 2 shows Time complexities of Rabin Karp and RB Matcher if n=20

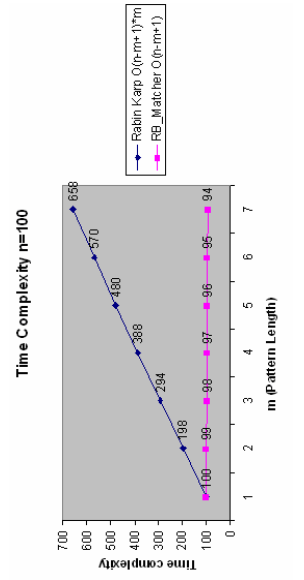

Fig. 3 shows Time complexities of Rabin Karp and B Matcher if n=100

B. Comparison in terms of Example

Rabin Karp Example: Text= 14412217356431121441, to find Pattern P= 1441, q = 11, Remainder of Pattern 'P' is p=0 Rest cases are Unsuccessful Hits. So in Rabin Karp to find out Pattern 'P' we encounter Spurious Hits which is extra processing.

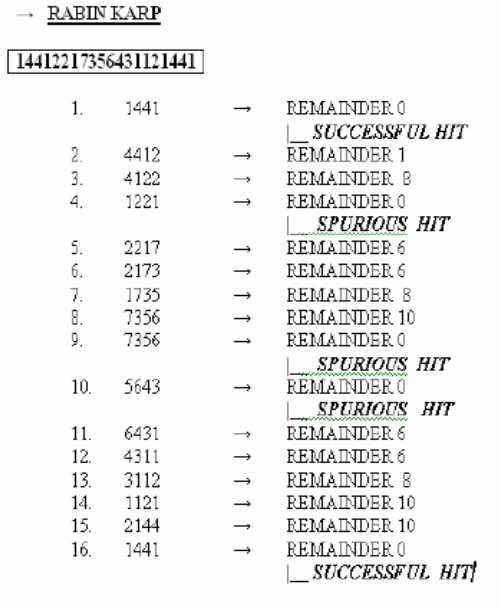

Fig. 4 Example Rabin Karp

Modified Rabin Karp Example:

1441 →REMAINDER = 0 & Q =131 ▶ *Successful Hit*
4412 →REMAINDER = 1 x
1221 →REMAINDER = 0 & Q = 121 x
2217 →REMAINDER = 6 x
2173 →REMAINDER = 6 x
1735 →REMAINDER = 8 x
7356 →REMAINDER = 10 x
3564 →REMAINDER 0 & Q = 324 x
5643 →REMAINDER 0 & Q = 513 x
6431 →REMAINDER = 7 x
4311 →REMAINDER = 10 x
3112 →REMAINDER = 10 x
1121 →REMAINDER = 10 x
2144 →REMAINDER = 10 x
1441 →REMAINDER = 0 & Q =131 ▶ *Successful Hit*

(Q denotes Quotient)

In this algorithm comparison of pattern and Text will always lead to successful hits.

C. Test Cases

We will some study test cases for the modified algorithms complexity.

TABLE I FOR MODIFIED RABIN KARP MATCHER (CONSTANT TEXT LENGTH)

S No	Length of text (n)	Length of pattern (m)	VALUE OF q	Complexity (n-m+1)
1	100	10	2	91
			3	91
			5	91
			7	91
			11	91
			13	91

			17	91
			19	91
			23	91
2	100	15	2	86
			3	86
			5	86
			7	86
			11	86
			13	86
			17	86
			19	86
			23	86
3	100	20	2	81
			3	81
			5	81
			7	81
			11	81
			13	81
			17	81
			19	81
			23	81
4	100	50	2	49
			3	49
			5	49
			7	49
			11	49
			13	49
			17	49
			19	49
			23	49

TABLE II FOR MODIFIED RABIN KARP (FOR CONSTANT PATTERN LENGTH)

S No	Length of text (n)	Length of pattern (m)	VALUE OF q	Complexity (n-m+1)
1	100	10	2	91
			3	91
			5	91
			7	91
			11	91
			13	91
			17	91
			19	91
			23	91
2	200	10	2	191
			3	191
			5	191
			7	191
			11	191
			13	191
			17	191
			19	191
			23	191
3	300	10	2	291
			3	291
			5	291
			7	291
			11	291
			13	291
			17	291
			19	291
			23	291
4	400	10	2	391
			3	391
			5	391
			7	391
			11	391
			13	391
			17	391
			19	391
			23	391

IV. CONCLUSION

With the above, we concluded that any numeric pattern can be found out from the given Text 'T' by following Modified RB matcher in an effective & efficient way. We also invite other Research Scholars to work on same and find out some better way to find out string from the given text.T.

REFERENCES

[1] Algorithm design and analysis by T.H Coreman .
[2] Algorithm design by Aho ulman and Hopcrafft
[3] www.algodesign.com
[4] Richard M.Karp. An Introduction to Randomized Algorithms. Discrete Applied mathematics,34:165-201,1991