Volume 4, Issue 3, March 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Rabin-Karp Algorithm with Hashing a String Matching tool

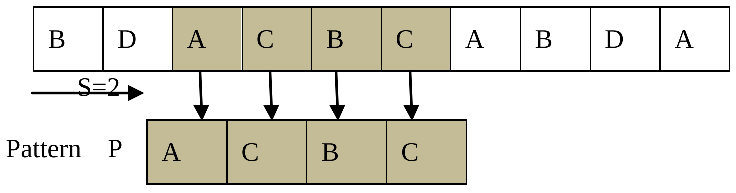
Ms Sunita, Ms Ritu Malik, Ms Mamta Gulia
S.E.S.B.P.S.M.V Khanpur Kalan
India

Abstract-String matching has always been an attainable problem for the developers. The problem of String Matching became vast if the length of the String to be matched is large. To solve this problem many String matching algorithm has been proposed by many of the developers. Rabin –Karp Algorithm is one of the strongest algorithms proposed for String Matching by Michael O. Rabin and Richard M. Karp in 1987. In this paper we will Study the Rabin Karp Algo that provides a better average running time by using the Hashing algorithm for String Matching. This paper will represent the Rabin Karp Algo. as a best tool for the Plagiarism. For a given source material Rabin-Karp can find out the multiple instance of pattern within the text ignoring details such as case and punctuation

Keywords—Rabin Karp, String, Text, Hashing, Prime, ASCII, Horner's Rule, Valid, Invalid, Spurious Hit Shift

I. INTRODUCTION

String Matching is the way to find out a particular pattern within a text. *Text* is a document that is being edited (it can be a large document file) and the pattern is the particular word that is searched by the user. For example to find out a particular pattern in a DNA Sequence. String matching can be a problem when we are searching for a particular pattern in a large size of text. So the *String matching problem* can be generalized in the terms as we suppose that Text is a large size of array T [1……..n] and pattern the user is looking for is the an array of P[1………..m] with length m≤ n. We again take that the *text* and the *pattern* all belongs to a set of finite alphabet Σ (set of all the alphabets). For example we may have Σ as(0,1,2……9) or (a,b,c,d,e,……….z). P (pattern) and T (text) are strings of character drawn from Σ. So in string matching we say that the pattern P occurs in the text T with shift s. *Shift* is the no. of character in text by leaving them pattern occur in the text(P occurs in the text beginning at the position S+1). Two types of Shift are there Valid Shift and invalid Shift. *Valid Shift:* if the pattern occurs in the text with shift S than it is a *valid shift* otherwise S is *an invalid shift*. String matching is the problem to find out all the Valid Shift in the text for which the pattern occurs in the text.
For Ex:-



(In this fig. the string matching problem the Text T= BDACBCABDA and the pattern P=ACBC. The goal is to find out all occurrence of the pattern P in the text with a valid shift. In this pattern occurs only once in the text with shift S=2. S=2 is said to be the valid shift). There are many numbers of approaches to find out all the possible pattern with the valid shift. One of the most basic approaches is the *Brute Force Approach* in this approach every character of the pattern is match against the text characters one by one. This approach is very slow and can be time consuming if we are finding pattern in a very large text file. So we need a better and faster approach to find out the pattern. This can be done by avoiding the comparison between the text character and the pattern character one by one. We'll try to compare the pattern and the text at once .So we need a good hash function with this we can find out the hash value of the pattern and compare it with hash value of all the possible substring of the text. For this we can use a better approach named as Rabin *Karp Algorithm for String matching.*

II. RABIN-KARP ALGORITHM FOR STRING MATCHING

Rabin-Karp Algorithm is a String matching algorithm created by Michael O. Rabin & Richard M. Karp in 1987. It uses Hashing to find out the possible pattern in the Text. . For text of length *n* and pattern p of combined length *m*, its average and best case running time is O (*n*+*m*) in space O (*p*), but its worst-case time is O (*nm*). In space O (m). Rabin Karp is a simple algo. That find out hash value of the length m pattern substring and then it find out hash value of all possible m length substring of the text. If Hash value of the pattern and Text substring match than string find out with shift s otherwise next substring value is matched to find out the string of length m. So in this way of finding out the pattern there is less number of comparisons as compare to *Brute Force Approach of String matching*. So there is only one comparison per text subsequence to find out the pattern. We assume that Σ= (0, 1, 2……9) each character is a decimal digit. Here we can take the ASCII code of all the character but must be careful for the multi-lingual. We can also assume that each character is a digit in radix-d notation as d= [Σ].We will find out the string with a given Pattern P[1…..m] of value p.

Text T[1……….n]. And denote the value of length m substring T[s+1……S+m] for shift s=0,1,2,….n-m. We can say that ts (substring of length m in the text)=p(pattern of the length m) if and only if T[s+1…..s+m]=P[1….m] with a valid shift s to compute P fast in θ(m) time we use

Horner's Rule:-

P=P[m] +10(p [m-1] +10(p [m-2]+……….10(p[2]+10p[1])

As *for ex.* We have pattern p [1….m] in ASCII format as P[31415] then we can compute P by Horner's rule P=5+10(1+10[4+10[1+10[3]]])

And to compute the remaining substring of length m in text represented by ts we can again apply the Horner's rule of Shifting

Ts+1= 10(ts-10^{m-1}T[s+1])+T[s+m+1]

For ex.=If m=5 and ts=23590 then we want to remove high order digit T[s+1]=2 and bring lower order digit 2 it is T[s+m+1]

T[s+1]=10(23590-10000*2)+2 = 35902.

By using this formula we can compute t1, t2, t3………t(n-m). T[s+1] can be computed from the ts in constant time by using the Horner's rule. Subtracting the 10^{m-1}T[s+1] removes the lower order digit from ts and by multiplying it with 10 shift it left one position and by adding T[s+m+1] bring lower order digit in T(s+1).In this equation 10^{m-1} is constant in the equation if it be computed in advance than overall string matching time can be reduced. This constant can be computed in θ (m) time. This is called the *Pre-processing time* of Rabin Karp Algo.

III. Use of Hash Function

Rabin-Karp Algo. try to speed up the string matching by testing the equality of pattern to the text substring using the Hash Function. Hash function is a function which converts every string into a numeric value. Two equal strings always return the same Hash value. In this way we can compute Hash value of Pattern we are looking for and then test to find out the equal hash value substring in the Text. First we will compute H (p) and then compute H (Si) and check if H(p)=H(Si) then Pattern found in the Text otherwise not.

However there can be the problem *First* There can be so many different substring that to keep the hash value small we have to assign some of the string same number. This means that if two substring's Hash value matches than it is not necessary that the substring actually match we have to check equality manually. But it usually doesn't happen so Rabin Karp Algo gives a better average search time.

To make it easy to compare two Hash values the range of hash function is set to sufficiently small nonnegative integer then two hash values can be compared easily in a single machine instruction. For this we take each string as nonnegative number but take the result as modulo K for some suitable modulus k. In general we take module K a sufficient prime no. so that 10q can be just fit in a single computer word so that all the necessary computation by the single computer instruction.

For ex:-. We have string CAR it can be represented by the nonnegative number as C=2, A=0, R=17 CAR=2+0+17=19 then we take k as 11(prime no.) CAR mod k=19 mod 11=6 we have to search value 6 in the text substring for an exact match. To work with modulo k we can compute Ts+1 as

Ts+1=(d(ts-T[s+1]h)+T[s+m+1])mod k where h= d^{m-1} d is the radix base;

Since there can be some problem like if ts=P(mod k) that doesn't means that ts=p

For example 31415mod13=10 and 67399mod13=10 but values 31415 ≠ 67399 this situation is called as *Spurious Hit* not a valid shift.

On the other hand if ts ≠p than we definitely have that p≠ts so it is an invalid shift. The testing for Spurious Hit and a valid hit can be done explicitly by checking the condition P [1………m]= T[s+1……….. S+m]. If we take K(prime no.) sufficiently large than the situation of Spurious Hit occur infrequently thus the cost of extra checking is low

IV. Algorithm

The idea previously defined are Implemented by this algorithm in this input are the text T and pattern P, radix d(which is typically taken Σ) and the prime no k is chosen

RABIN-KARP- MATCHER (T,P,d,K)

```
[1]  n ←length [T]
[2]  m ←length [P]
[3]  h ←$d^{m-1}$mod K
[4]  p ←0
[5]  $t_0$ ←0
[6]  for I ←1 to m        □ Pre-processing
[7]  do P ← (dp + P[i]) mod K
[8]  $t_0$ ← (d$t_0$ + T[i]) mod K
[9]  for s ← 0 to n-m
[10] Do if p = $t_s$
[11] then if P[1……m] = T[s+1……..s+m]
[12] then print "pattern occurs with shift s "
[13] if s <n-m
```

[14] then Ts+1=(d(t s-T[s+1]h)+T[s+m+1])mod k

V. WORKING OF RABIN- KARP ALGO.

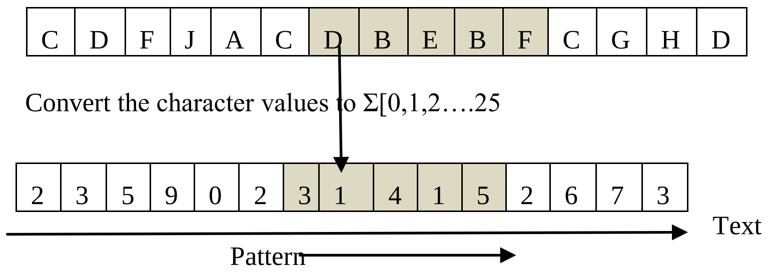
In the Rabin Karp algo all character are taken as Radix d interpretation. N is the length of overall text string we have and m is the length of pattern that the user is looking for. In line 3 h is initializing to higher order digit position of an m digit window. Line 4 and 5 initialize the pattern p and t_0 to 0 and then in the next lines from 6 to 8 it compute the value of pattern p and first substring of text as t_0 for the length 1 to m. it compute the pattern value as P[1…….m] mod K and t_0 as the value of T[1…….m] mod K for all the m length substring of Text T .line 9 to 14 check all the valid shift in the Text for the pattern. Line 9 check for all the possibilities of valid shift from 0 to n-m . Line 10 check if the module value for the Pattern and the Substring of Text Ts is same p = t_s. If it is true then it again explicitly checks for the possibility of Spurious Hit in line 11 by testing the condition P [1……m] = T[s+1……..s+m]. Then if it is found a valid shift in line 11 it is printed in line 12 that pattern occurs with valid shift in the text. Then if condition s<n-m found true in line 13 then for loop is executed again at least one more time to check the possibility of an extra valid pattern in the text. Then line 14 compute the value of t[s+1] mod K from the value of Ts mod K that has already been computed in line 7 and 8 in constant time by using the equation Ts+1=(d(t s-T[s+1]h)+T[s+m+1])mod k directly. .

This working can be represented with the help of this method

[1] Take the Text String that is to be matched for a pattern
[2] Convert the alphabetic string to be matched to the alphabet values for [A,B,C...................Z]=[0,1,2.....................25]
[3] Take the Pattern String to be matched
[4] Convert the Pattern String to the alphabets value same as step 2
[5] Choose a Prime number
[6] Calculate the mod of the Pattern String against the Prime number chosen in step 5
[7] Divide the Text String length to the every possible length of the Pattern String(if the length of the pattern string is 5 then divide the Text String to the every possible length of 5)
[8] Find out Mod of the every possible length of the Text String to the prime number
[9] Match the every possible mod value calculated in the previous step to the mod value of the Pattern String.
[10] Do the manual matching for all the matched mod results for the Spurious Hits and Valid Hits against the actual alphabetic Pattern String that you want to find out in the text String.

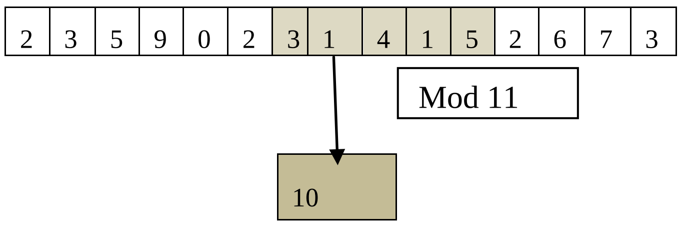
VI. EXAMPLE FOR RABIN- KARP ALGO

Suppose we have a Text String CDFJACDBEBFCGHD and we have pattern DBEBF that we are looking for. In this example we take that each character is represented by values [0, 1,2,…. 25] for all the alphabets [A,B,C…….Z]. We will take Prime number 11 to find out the modulo

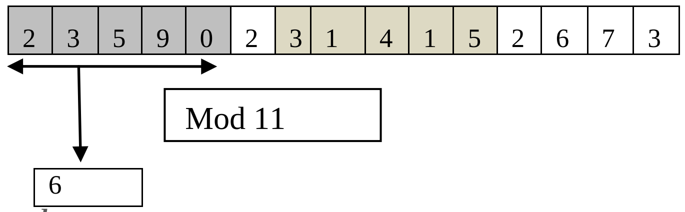


C D F J A C D B E B F C G H D
Convert the character values to Σ[0,1,2….25
2 3 5 9 0 2 3 1 4 1 5 2 6 7 3
Text
Pattern

Then find out the modulo 11 for the pattern string

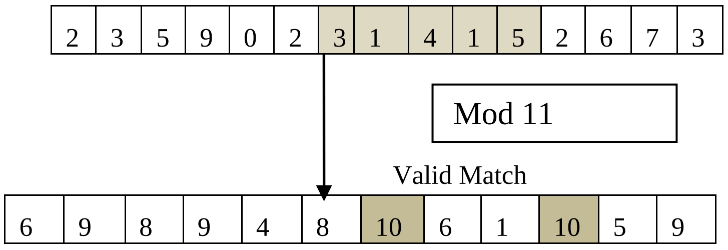


Then find out the Modulo 11 for the every possible substring of pattern length m in the Text in this example Text is of N=15 and pattern m=5 , prime no.=11



In this example the string whose modulo is find out showed in link pink colour in the same way modulo for the whole possible substring is find out.

23590 mod 11 = 6	31415 mod 11= 10
35902 mod 11 = 9	14152 mod 11= 6
59023 mod 11 = 8	41526 mod11 = 1
90231 mod 11 = 9	15267 mod 11= 10
02314 mod 11 = 4	52673 mod 11 = 5
23141 mod 11= 8	26739 mod 11 = 9



In this mod value for the spurious hit and valid hit are same these are shown shaded as
31415 mod 11 =10 and 15267 mod 11 =10
But our pattern is 31415 so another value 15267 is a *Spurious Hit* this can be handled by extra checking. In this way we can find out our exact pattern 31415 or DBEBF by a less computation.

VII. PERFORMANCE OF RABIN-KARP ALGO

Rabin Karp Algo take θ(m) pre-processing time in which it precompute the pattern modulo and Text substring modulo in a constant time . By which all the substring of Text can be computed in a very less time. Its worst case running time is θ(n-m+1). The Rabin Karp Algo explicitly verifies every valid shift this verification take time θ((n-m+1)m) because n-m+1 is a valid shift. In many application we expect few valid shifts in this the expected matching of the algorithm is O((n-m+1)+cm)= O(n+m) plus some extra time to process Spurious Hits.

VIII. ADVANTAGES/ DISADVANTAGES OF RABIN KARP ALGO

Advantages of Rabin-Karp Algo:-

[1] Rabin-Karp algo choose a random prime number that actually formalizes the intuition
[2] Good for Plagiarism because it can deal with multiple patter matching
[3] With a good hashing Algo it is quite effective to match pattern.
[4] Good Algorithm as it can find out pattern without actually caring for the detail such as the Punctuation mark

Disadvantages of Rabin-Karp Algo

[1] There are many other algo that are faster than O(n+m)
[2] It is practically very slow and also take extra space

IX. CONCLUSION

Rabin-Karp Algo is a great algorithm with a very simple algo using Hashing technique. In this paper we represent that If we use a strong Hashing algo and a careful approach for randomly selecting prime number Rabin Karp Algo can be used as a best tool for multiple pattern matching. This algo can be perfectly used for Plagiarism even for larger phrases. This algo provides a better average searching time than other single string matching algorithm. It can be used for pattern matching from source text with ignoring details such as punctuation and case with a pre-processing constant time and matching time and also with a very less computation effort.

ACKNOWLEDGMENT

We would like to express our deep gratitude to Assistant Professor Ms. Sunita,our research work supervisor, for her valuable and constructive suggestions during the planning and development of this research work. Her willingness to give her time so generously has been very much appreciated.

REFERENCES

[1] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001-09-01). "The Rabin–Karp algorithm". *Introduction to Algorithms* (2nd ed.). Cambridge, Massachusetts: MIT Press
[2] Karp, Richard M.; Rabin, Michael O. (March 1987). *Efficient randomized pattern-matching algorithms*
[3] En.wikipedia.org/wiki/Rabin-Karp_Algorithm
[4] WWW.Sci.Unich.it/~acciaro/Rabin-Karp.pdf
[5] WWW.stoimen.com/blog/2012/04/02/computer_algorithms-Rabin-Karp_String_Searching/
[6] WWW.CS.Utex.edu/~Plaxton/c/337/05f/slides/stringmatching-1.pdf
[7] People.Cedarville.edu/Employee/..Web/…/Rabin_Karp_matching.ppt