ISSN:0975-9646

Shivendra Kumar Pandey et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4732-4735

A Study on String Matching Methodologies

Shivendra Kumar Pandey, Neeraj Kumar Dubey, Sonam Sharma

Department of Computer Application and Research
National Institute of Technical Teachers Training and Research
Bhopal-462002, MP, India

Abstract— **String matching is the technique of finding the occurrences of a character pattern in a given string. In this paper, we provided an overview of different pattern matching algorithm and also proposed a algorithm For pattern matching .In this paper we have evaluated several pattern matching algorithms such as Naive String matching algorithm , brute force algorithm , Rabin-Karp algorithm , and K-M-P algorithm , and we proposed an algorithm for pattern matching . Our proposed algorithm works in linear time, if the number of occurrences of the pattern in a string is very less .We also compared the matching performance of these algorithms. It is observed that the performance of string matching algorithm is based on the selection of algorithms used for matching.**

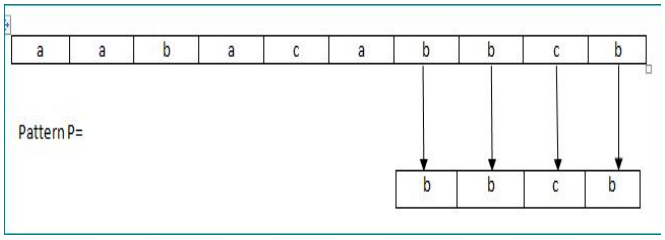
Keywords— **String Matching , Rabin Karp Algorithm, KMP, our purposed Algorithm , comparison of String Matching algorithms.**

1. INTRODUCTION

String matching is a technique to find out the pattern in a given string . let ∑ be a set of alphabet . the member of set ∑ called character or symbol . For Example let ∑={a, b ,c} , then aabbcaa is a String over ∑ .The pattern denoted by P[1..m] and Text denoted by T[1…n].If pattern P occurs in Text T with sift 's' then we call pattern occur with sift 's' and 's' is a valid shift otherwise it is invalid shift. The string matching is a technique of finding all valid presence of the pattern P in Text T.

EXAMPLE-> Let ∑= {a , b , c} and Text T= a a b a c a b b b c b . And pattern P = b b c b now

Text T=



From above figure the value of valid shift s = 6.

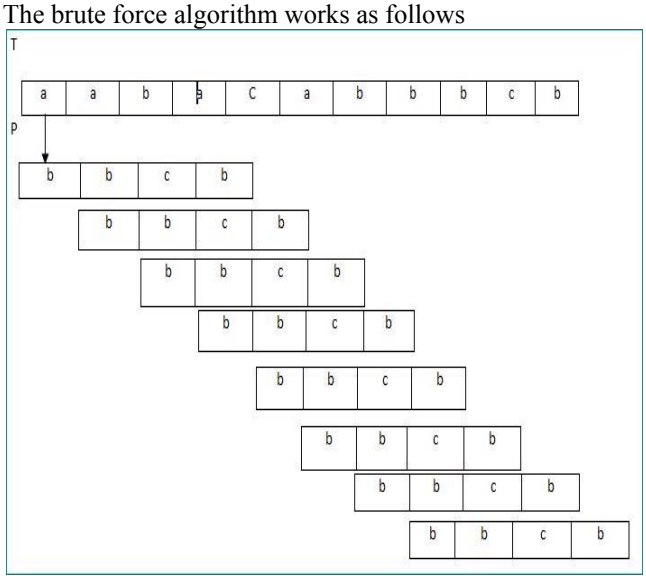
2. NAÏVE STRING MATCHING ALGORITHM

Naïve String matching algorithm also known as brute force matching . In this technique pattern matched by text character by character and slide right one character at a time. this technique doesn't require any preprocessing time and takes constant extra space .expected character comparison required in this technique is 2n. in this technique, it checks for valid shifts by using for loop which checks whether P[1…m] = T[s+1..s+m] for each of the n-m+1 possible value of s

Consider the following example

Let ∑= {a , b , c} and Text T= a a b a c a b b b c b . And pattern P= b b c b

The brute force algorithm works as follows



Worst case time complexity of Naïve String matching algorithm is O((n-m+1)m). Matching time of Naïve string matching algorithm equals to its running since it doesn't require any preprocessing time

3. RABIN-KARP STRING MATCHING ALGORITHM

This algorithm works in two phases in the first phase it pre-process and in a second phase it performs matching. This algorithm uses hashing technique. Rabin-Karp algorithm is used for finding the numeric pattern in a given text T. in this algorithm first it divides patterns P by a predefined prime number q and calculates remainder of the pattern. In next step it calculates the remainder of first m character of text T. If the remainder of first m characters and remainder of the pattern P are equal, then only it perform matching .otherwise no need of comparison. This process will be repeated from s =0 to s= n-m. Time complexity of Rabin-Karp algorithm in first phase (pre-processing phase) is O(m) and in matching phase is O((n-m+1)m). Suppose we have two number A and B then there are following possibility

If remainder of A =REM(A/q) and remainder of B=REM(B/q) are equal

1>Then successful hit occur if REM(A/q) and REM(B/q) are equal and A=B.

2>Spurious hit if REM(A/q) and REM(B/q) are equal but A!=b.
3>Unsuccessful hit REM(A/q) and REM(B/q) are not equal and A!=b.
Example. For a given Text T pattern P and Prime number q.
T=2345678997977979765343566788867564568909755453434342454547565 5454

P= 667888
q=11

REM(Text) =234567/11 =3
REM(P) = 667888/11 =1
REM(Text) ≠ REM(P)
Now move on to the next set of characters from text and repeat the procedure.

4. KNUTH-MORRIS-PRATT ALGORITHM

Knuth Morris and Pratt developed an algorithm for pattern matching which takes linear time O (n) for finding patterns. This algorithm works in two phases. In the first phase it runs an algorithm which is called a preprocessing algorithm and in this phase it calculates prefix function. In the second phase this algorithm runs KMP matcher algorithm.

A. The prefix function ,π

The prefix pattern , π for a pattern contains the idea about how the pattern matches against the shifts of itself. Prefix function is used for avoiding useless shifts .

B. The KMP Matcher

In this phase , it has String 'S', pattern 'P' and prefix function π as input . It finds the occurrence of pattern in a string and returns number of shifts of pattern after which occurrence is found.

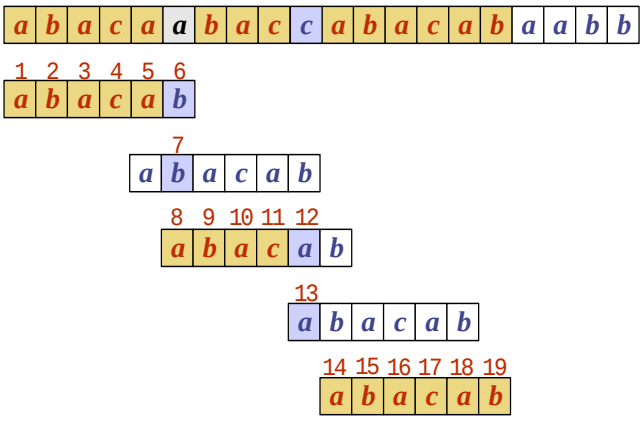
Example:

Let a pattern P = abacab and a string
S= abacaabaccabacabaabb

Prefix function,π

I	0	1	2	3	4	5
P[i]	a	b	A	c	a	b
Π[i]	0	0	1	0	1	2

KMP Matcher:



5. OUR PROPOSED ALGORITHM:

In this algorithm we work in two phases. First phase is the preprocessing phase and second phase is matching phase .In preprocessing phase, we only find some index values of substrings of length m from text T and the second phase matches these substrings with actual pattern P. We tried to show the working of two phases by following two algorithms.

A. Phase 1 preprocessing phase

In the first phase that is preprocessing phase we will find the some index values from each substring of length m from text T which matches with first and last character of pattern P and insert these index values into a the queue Q . The approximate running time of pre-processing phase is O(n-m).

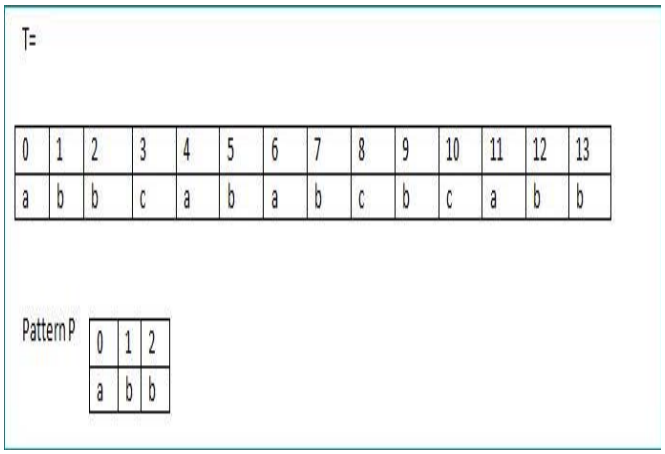
```
1.  n =  length S;
2.  m = length P;
3.  X ←P[0],y <-P[m-1];
4.  For(i=0 to  n-m);
5.     If(S[i]==x   &&  S[m+i-1]==y)
6.      Insert I into the queue  Q
7.  Return Q
```

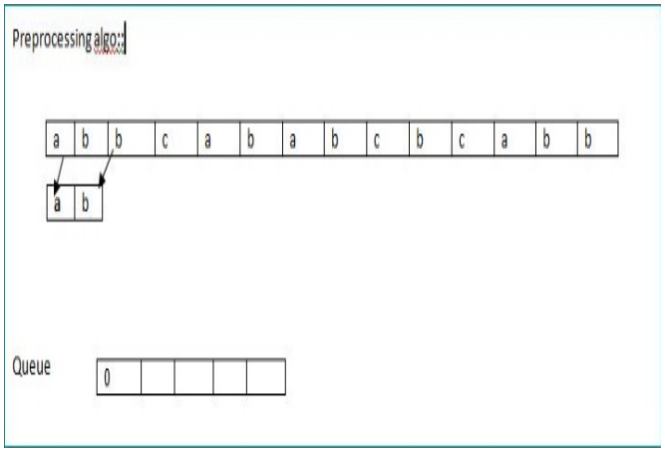
B. Phase 2 Matching phase->

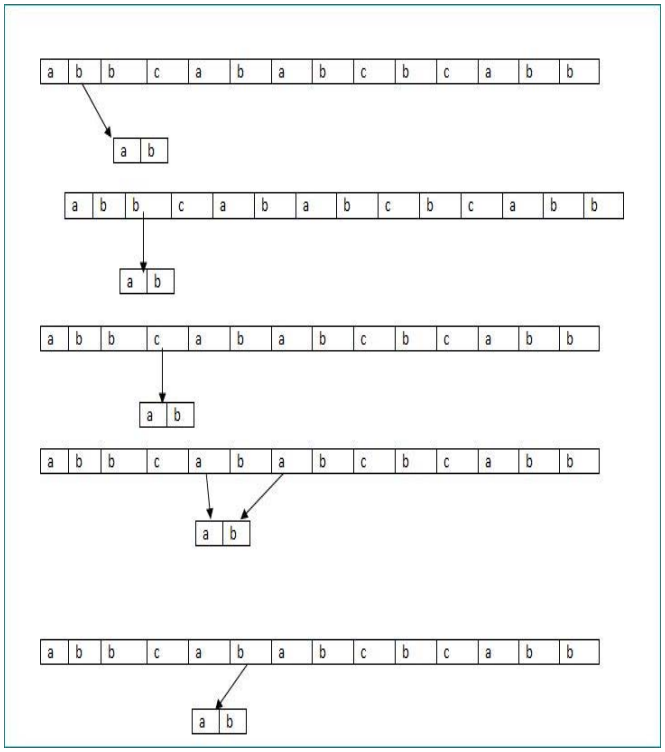
Matching phase totally depends on the size of the queue. In this phase, we find the substrings of length m on the basis of index value And compare the pattern with the substrings of size m. if a match occurs, then we called it is successful hit otherwise it will be spurious hit . The approximate running time of matching algorithm O (lm) where l is the size of the queue and m is the length of pattern. Following algorithm shows the matching phase

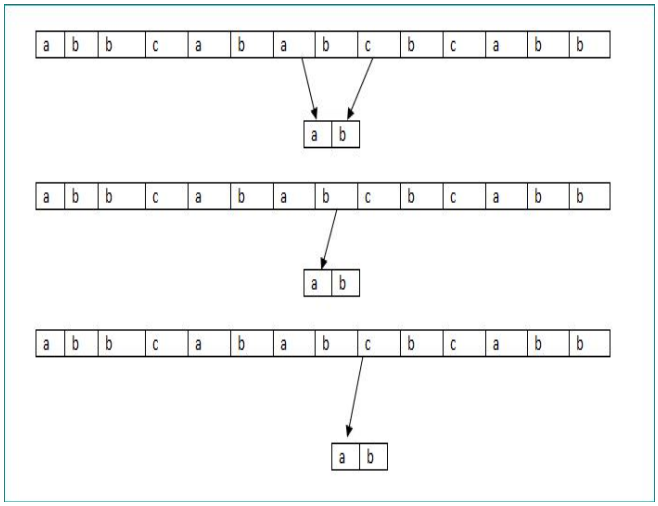
```
1.  While(Q!= empty)
2.  {
3.   I= dequeue()
4.  K =I;
5.  Count==0;
6.  For(j=0 to m-1)
7.  {
8.  If(P[j]==S[k])
9.  {
10.       K=k+1;
11.       Count = count+1;
12. }  // end of if condition
13. Else
14.          Goto 1
15. }
16. If(count==m)
17. {
18. Print("pattern occur at location " i);
19. }
20. }  // end of while loop
21. Exit
```

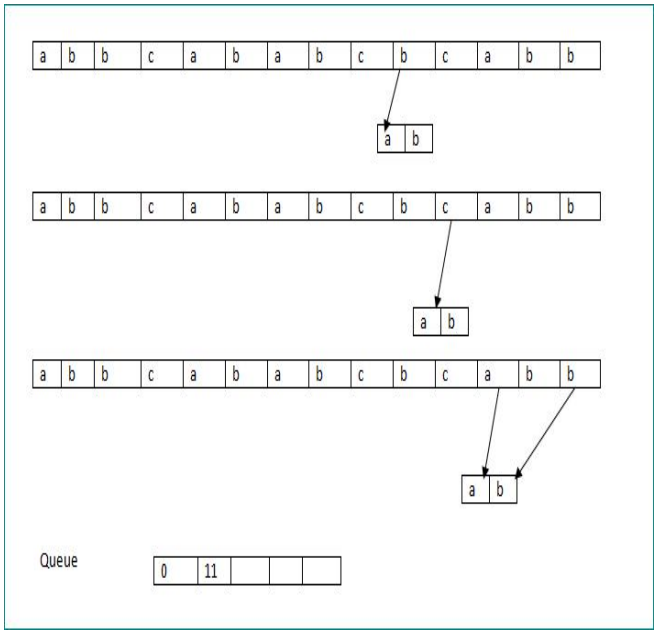
Example
Let we have a given text T=a b b c a b a b c b c a b b, Pattern P= a b b and the queue Q.

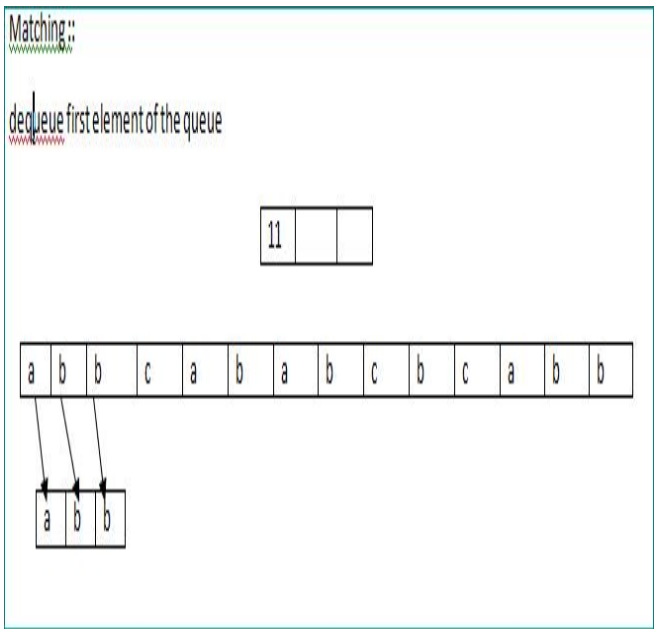




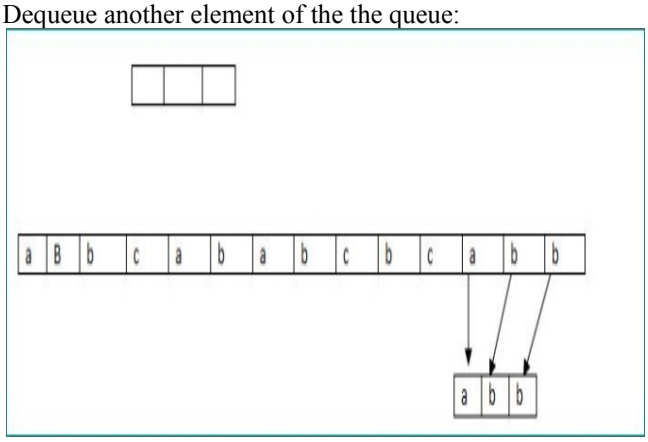








Dequeue another element of the the queue:



6. COMPARISON:

Algorithms	Time complexity for preprocessing	Time complexity for Matching
Brute force algorithm	No preprocessing is required	O((n-m+1)*m)
Rabin karp	O(m)	O(n+m)
Knuth-morris-pratt	O(m)	O(n+m)
Our proposed algorithm	O(n-m)	O(l*m)

7. CONCLUSIONS:

This paper evaluates the performance of some existing algorithm with our purposed pattern matching algorithm. Our proposed algorithm is a linear time matching algorithm. It could be better to other algorithms in some constraints .the running time of this algorithm depends on size of the queue i.e. the number of index value in the queue which shows that the number of expected patterns in text T. if there is only one element in the queue then maximum time taken by matching phase would be O(m) , where m is the size of pattern.

REFERENCES

1.http://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string_search_algorithm.
2. http://en.wikipedia.org/wiki/String_matching.
3. http://www.digitalforensicssolutions.com/Scalpel.
4.Introduction to algorithm ,corman
5. International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-I, Issue-6, January 2012
6. Xinyan Zha and Sartaj Sahni "Multipattern String Matching On A GPU",IEEE,2011,pp. 277-282.
7. Nathan Tuck, Timothy Sherwood, Brad Calder, George Varghese "Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection" IEEE INFOCOM 2004