International Journal of Latest Trends in Engineering and Technology
Special Issue SACAIM 2017, pp. 362-365
e-ISSN:2278-621X

COMPARATIVE ANALYSIS ON EFFICIENCY OF MULTIPLESTRINGPATTERN MATCHING ALGORITHMS

Haritha B.K[1], Joyline Jessica Correa[2] & Ms.SuchethaVijayakumar[3]

Abstract- The main objectiveof the pattern discovery process is pattern matching, where it is already known and we want to figure out how often and where it takes place in a sequence. Multiple-pattern matching algorithms are the heart of many network intrusion detection systems signature matching engines. Simultaneously they allow these engines to rapidly search for many patterns with multiple input passing through such systems, but frequently consume most of the processing time. Thus, processing should be as fast as possible to ensure system scalability into networks of ever-increasing speeds. Simultaneously they must enforce security so that they are not susceptible to algorithmic complexity attacks.
We will implement multiple pattern matching with the following Algorithms:

- **Brute Force Algorithm**
- **Rabin–Karp string search algorithm**

Keywords- Execution Time, Text, Pattern,Window.

1. INTRODUCTION

Multiple pattern matching algorithms allows engines to quickly search number of patterns simultaneously. But often consume most of the processing time. Here it checks sequence of tokens for the presence of some patter. It takes the input pattern 'P' of length and text 'T' of length, where 'P' will be smaller than 'T'. This technique has two categories:

- Single pattern matching technique
- Multiple pattern matching technique

In single pattern matching it is required to find all the occurrences in the given input text 'T'. If the input text matches simultaneously the text is called multiple pattern matching. In pattern matching, considers that the text is examined through the window. By default, the window slides from left to right, but during the search it shifts accordingly to the rules and regulations of the algorithm.

When the window is at a particular position of the text, this checks whether the pattern occurs there or not. If there is a whole match, the position is displayed.

The main objective behind the pattern-matching algorithms is to reduce the total number ofcharacter comparisons and to reduce time required for both worst and average case analysis. The efficiency of algorithms is evaluated by their running times and the type of inputs they provided. The pattern matching algorithms are used across network security environments, Text Editors,Information Retrieval etc.

2. TEXT PATTERN MATCHING ALGORITHM

2.1 The Brute Force Algorithm

The simplest approach for string matching problem is - The Brute ForceAlgorithm which is also known as Naive Algorithm. This algorithm follows linear search approach. As shown in the Figure1this algorithm tries to match the first letter "T" of the Textand the first letter "i" of the Pattern and checks whether these two letters are equal. If so, then check second letters of the text that is "h" and pattern "s". If it is not equal, then move first letter "i" of the pattern to the second letter of the text. Then check these two letters. When we find a match, return its starting location and continuous the same till nth position of text.

Example:

Let the Text (T) be,
THIS IS TEXT
and the Pattern (P) be,
I

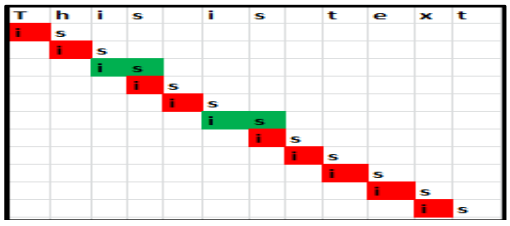

Figure1. Example for Brute-Force Algorithm

[1] Aloysius Institute of Management and Information Technology, Mangalore, Karnataka, India.
[2] Aloysius Institute of Management and Information Technology, Mangalore, Karnataka, India.
[3] Asst. Professor, Department of MSc. ST, AIMIT, Beeri, Mangalore, Karnataka, India.

Implementation:

```
void bruteForceMatcher(String text, String pattern){
int n=text.length();
int m=pattern.length(),j,flag=0;
for(int i=0;i<=(n-m);i++){

    j=0;
    while((j<m)&&(text.charAt(i+j)==pattern.charAt(j))){
        j++;
    if(j==m)
        System.out.println("Pattern starts at position"+(i+1));
    flag=1; //match at i
    }
 }
if (flag==0)//no match
System.out.println("Pattern not found");
}
```

2.2. The Karp-Rabin Algorithm

This algorithm exploits a hash function to speed up the search. It calculates a hash value for the pattern to be searched and each subsequence of text to be compared.Then both the hash values are compared. If the hash values are not equal then the algorithm will estimate the hash value for next character sequence. Suppose if the hash values are equal then the algorithm will do the brute force comparison with the pattern and the character sequence for which the hash value matched as shown in Figure2.

Example:

```
Hash value of "ABCA" is 65
Hash value of "A" 65
1)ABCA
  A
        65=65          1 comparison made
2) ABCB
    A
        65≠66          1 comparison made
3) ABCA
     A
        65≠67          1 comparison made
3) ABCA
      A
        65=65          2 comparison made
```

Figure2. Example for Karp-Rabin Algorithm

Implementation:

```
public int patternSearch(char[] text,char[] pattern){
    int m=pattern.length;
    int n=text.length;
    long patternHash=createHash(pattern, m-1);
    long textHash=createHash(text,m-1);
    for(i=1;i<=n-m+1;i++){
        if(patternHash==textHash && checkEqual(text,i-1,i+m-2,pattern,0,m-1)){
            flag=1;
            System.out.println("Pattern starts at position"+(i));
        }
        if(i<n-m+1){
            textHash=recalculateHash(text,i-1,i+m-1,textHash,m);
        }
    }
      if (flag==0){
            System.out.println("Pattern not found");
         }
    return flag;
}
```

3. COMPARATIVE ANALYSIS ON STRING PATTERN MATCHING ALGORITHM

In this paper, we analysed selected Multiple pattern string matching algorithms on the basis of Execution time and search type. Each algorithm has certain advantages and disadvantages.

3.1 Observations on Algorithms Used For Matching:

1) Brute-Force String Search Algorithm:

The "naive" approach is easy to understand and implement but it can be too slow in some cases. If the length of the text is 'n' and the length of the pattern is 'm', in the worst case it may take (n * m) iterations to complete the task.The main advantage of Brute Force Algorithm is that wide applicability, simplicity, yields reasonable algorithms forsome important problems (e.g., matrix multiplication, sorting, searching, string matching).Weakness of the brute-force algorithms are unacceptably slow, not as constructive as some other design technique. It takes much time as it search linearly.

2) Rabin Karp String Search Algorithm:

It is a string searching algorithm that uses hashing to find any one of a set of pattern strings in a text. This algorithm works well in many practical cases such as it's good for plagiarism, it can deal with multiple pattern matching, but can exhibit relatively long running times on certain examples, such as searching for a pattern string of 10,000 "A"s followed by a single "B" in a search string of 10 million "A"s.

3.2 Analysis

Algorithm	Length of I/P No. of executions	Nature of I/P									Findings
		AU			AL			M			
		1	2	3	1	2	3	1	2	3	
Brute Force (BFA)	Short	6	4	3	5	4	3	5	3	2	BFA is good for short strings and search is faster for lower case than uppercase letters.
	Long	8	5	3	8	7	5	9	7	6	
Karp Rabin (KRA)	Short	6	5	4	4	3	2	4	3	2	KRA results are similar to BFA but and search is faster for mixed letters compared to BFA.
	Long	8	6	5	8	7	6	9	6	5	
AU- All Uppercase		AL- All Lowercase						M- Mixed			

Table1.Table of findings for execution time (in seconds) based on various inputs for different algorithms.

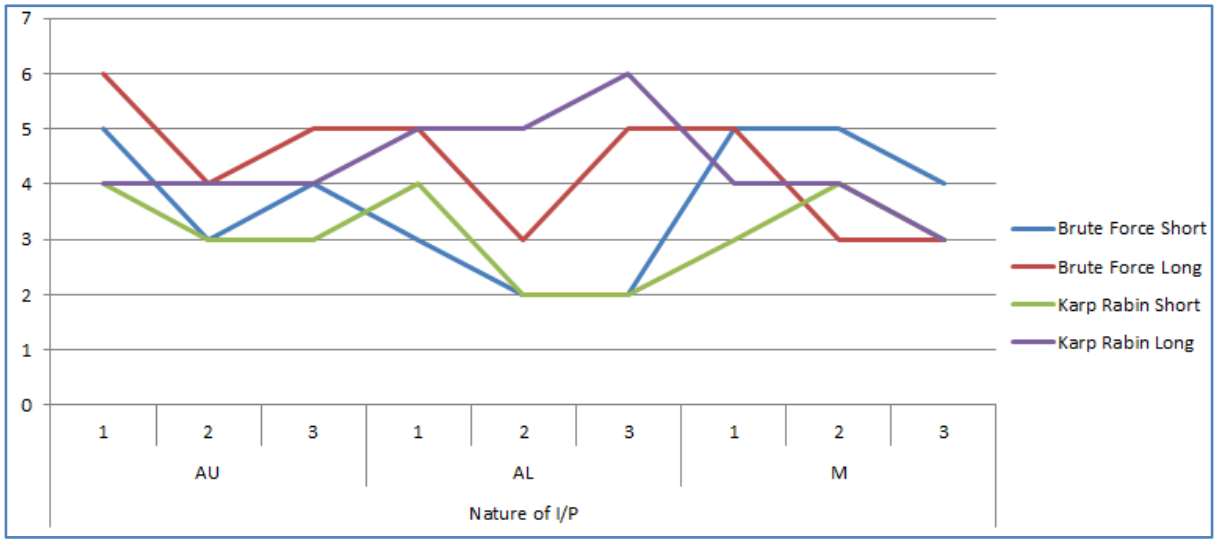

Figure 2: Graph showing comparative analysis of algorithms

By looking at the performance of various algorithms in Table 1, we can conclude that the best algorithm in majority of the cases is Rabin Karp.When the pattern is long and mixed,its results are a lot better than the naive solution and it is definitely the best choice in the situations where Boyce-Moore is not adapted. TheBrute-Force Algorithm obtains similar results, but the performance is very slow when the pattern is very large. The Brute-Force could be a good choice if the length of pattern is very short, Where Rabin-Karp obtains very good results in these tests

4. CONCLUSION

We have presented theMultiple Pattern String Matching Algorithms. In this study there are various scenarios for improving the performance of multiple pattern matching algorithm .Where Comparison of proposed algorithm is made. To answer the question: Which algorithm is the best? We implemented the two algorithms using programming language (JAVA), and after conducting different testscomparisons and the attempts with these implementations, the answer is that Rabin Karp is the Best Algorithm. Not in all cases, but in the practical cases, Rabin Karp algorithm is extremely fast on mixed text that is why we can consider this algorithm as the best one.

5. REFERENCES:

[1] "Analysis of Multiple String Pattern Matching Algorithms" Akinul Islam Jony, International Journal of Advanced Computer Science and Information Technology (IJACSIT) Vol. 3, No. 4, 2014, Page: 344-353, ISSN: 2296-1739

[2] " Multiple Pattern String Matching Methodologies: A Comparative Analysis" Zeeshan Ahmed Khan , R.K PateriyaInternational Journal of Scientific and Research Publications, Volume 2, Issue 7, July 2012 1 ISSN 2250-3153

[3] GeorgyGimel'farb, "String matching Algorithms", COMPSCI 369 Computational Science AkhtarRasool Amrita Tiwari et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012,3394 – 3397

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms, Third edition. The MIT Press, 2009.

[5] Algorithms for String matching,Marc GOU, July 30, 2014

[6] https://www.cs.purdue.edu/homes/ayg/CS251/slides/chap11.pdf

[7] http://www.geeksforgeeks.org/pattern-searching-set-7-boyer-moore-algorithm-bad-character-heuristic/

[8] http://www.stoimen.com/blog/2012/03/27/computer-algorithms-brute-force-string-matching/

[9] https://ccsl.carleton.ca/people/theses/Kelly_Master_Thesis_06.pdf

[10] http://www.ijltet.org/journal/148125788537.pdf