Journal of Applied Data Sciences
Vol. 3, No. 1, January 2022, pp. 15-23

ISSN 2723-6471

Knuth Morris Pratt String Matching Algorithm in Searching for Zakat Information and Social Activities

Fendi Riawan [1,*] , Taqwa Hariguna [2]

Universitas Amikom Purwokerto, Indonesia
[1] fendiriawan0102@gmail.com*; [2] taqwa@amikompurwokerto.ac.id
* corresponding author

(Received: November 18, 2021; Revised: December 20, 2021; Accepted: January 5, 2022; Available online: January 25, 2022)

Abstract

Algorithms are one of the components that need to be considered in the development of information systems. Determination of the algorithm is adjusted to the purpose of the system to be built. One algorithm that can be used is string matching. The string matching algorithm will play a role in searching for a string consisting of several characters (usually called a pattern). The method used in this research is string matching knuth morris pratt (KMP) which is used to search zakat information and social activities in the search engine system. KMP is a string matching algorithm with good performance. The results showed the performance of string matching using the KMP algorithm with 5 trials of input pattern on zakat information with execution times of 0.03 ms, 0.03 ms, 0.02 ms, 0.02 ms and 0.03 ms. And 5 times the input pattern experiment on social activities with execution time of 0.02 ms, 0.02 ms, 0.03 ms, 0.03 ms and 0.02 ms. Thus the average execution time of the KMP algorithm in string matching is 0.026 ms and 0.024 ms.

Keywords: String Matching Algorithm; Knuth Morris Pratt; Pattern; Search Engine

1. Introduction

In the industrial era 4.0, information systems are now growing which have been used in various circles to facilitate user activities and transactions. Algorithms are one of the components that need to be considered in the development of information systems. Determination of the algorithm is adjusted to the purpose of the system to be built. One algorithm that can be used is string matching. String Matching is the process of finding all occurrences of the query, which is then called a pattern into a longer string [1]. String matching algorithms (patterns) that have good performance are Knuth Morris Pratt (KMP) and Boyer Moore Algorithm. For string matching using the Brute Force algorithm, every time a pattern mismatch is found with the text, the pattern will be shifted one character to the right. While the KMP algorithm can maintain the information used in performing the number of shifts. The KMP algorithm uses this information to make further shifts, not just one character as in the Brute Force algorithm [2]. The optimization of the KMP algorithm will allow an increase in the execution speed of the string matching program [3].

The KMP algorithm has the advantage of finding matches on large files. The KMP algorithm searches for text in order from left to right at the beginning of the text and then shifts the word order to the end of the text [4]. The KMP algorithm performs matching with the forward matching performance flow [5]. The KMP algorithm basically works through two stages, namely KMP will process the pattern and text first to obtain the basic pattern and index value that will be used for information on calculating the number of shifts. And the second stage of the KMP algorithm will compare the character string and character pattern from left to right, from the beginning to the end of the pattern [6]. With the function and purpose of the Knuth Morris Pratt algorithm, the algorithm will be appropriate when used in

optimizing search results. Therefore, this research will implement the knuth morris pratt algorithm in website-based software engineering for optimizing search results in the study of finding information about zakat and social activities at LAZISMU Baturraden.

2. Knuth Morris Pratt (KMP) Method/Algorithm

Knuth Morris Pratt Algorithm (KMP) is an algorithm that is used in the search process where a string (in this case is referred to as a pattern) is found in a collection of other strings with a larger size [7]. The KMP (Knuth Morris Pratt) algorithm is a type of Exact String Matching Algorithm which is an exact match of strings with the arrangement of characters in the string being matched having the order or number of characters in the same string [8].
The KMP algorithm can shift effectively because it does preprocessing on the pattern so that a LPS component (Longest Proper Prefix Which Is Suffix) can be known, which is obtained based on the pattern sought before the main program is run. LPS is utilized for the largest shift that can be made to avoid unnecessary comparisons [9]. A match occurs when the characters in the text and the characters in the pattern being compared are the same, while the mismatch is the opposite [10].

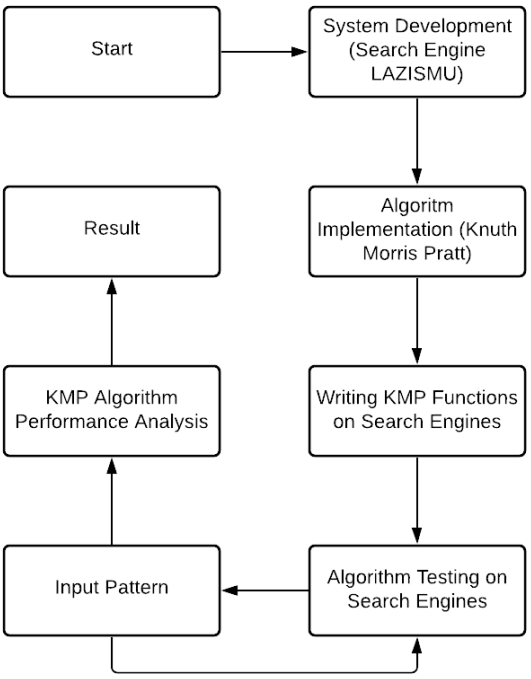


Figure. 1. KMP algorithm method

1) LAZISMU Search Engine System Development
At this stage the researchers designed a search engine system that was used as a medium in research [11]. The thing to do at this stage is to build a database and display the search engine.

2) Knuth Morris Pratt Algorithm Implementation
At this stage, the Knuth Morris Pratt algorithm is implemented as a search engine to obtain accurate and optimal search results [12].

3) Writing KMP Functions on Search Engines

In order for the Knuth Morris Pratt Algorithm to run well, it is necessary to write several functions, namely KMPSearch and preKMP [13].

4) Testing the Knuth Morris Pratt Algorithm on Search Engine Systems

At the testing stage, it will be tested regarding the implementation of the Knuth Morris Pratt algorithm in the search system (search engine) [14-17]. At this stage, the performance of the Knuth Morris Pratt algorithm will be known in obtaining optimal and accurate search results.

5) Input Pattern

In the stage of testing the performance of the KMP algorithm, the first step that must be done is to enter a pattern. Where in this system the pattern is a search query entered by the user on the search engine system [18]. The input pattern will continue to be performed every time the KMP algorithm is tested [19].

6) Knuth Morris Pratt Algorithm Performance Analysis

After the pattern input process is carried out, an analysis phase will be carried out on the search performance carried out by the KMP algorithm for pattern matching against the text in the database [20]. And it can be seen as the result of the performance of the KMP algorithm.

3. Result

Optimization of search for zakat information and website-based social activities using the Knuth Morris Pratt algorithm. In the process of implementing the Knuth Morris Pratt algorithm for optimizing the search for zakat information and social activities, it consists of several stages, namely analysis, designing a search function system (search engine), implementing the Knuth Morris Pratt algorithm on the system, and testing the Knuth Morris Pratt algorithm on the search engine function in system.

1) LAZISMU Search Engine System Development

Before implementing the Knuth Morris Pratt (KMP) algorithm in optimizing the search for zakat information and social activities, a search engine system is needed which will later become a medium for implementing the KMP algorithm.

2) Knuth Morris Pratt Algorithm Implementation

After the design stage of the search engine function system is complete, then enter the implementation stage of the Knuth Morris Pratt algorithm. The KMP algorithm will play a role in the search for zakat information and social activities optimally in the search engine system. Where the search process is carried out by matching the pattern entered by the user with the existing text pattern in the database.

3) Writing KMP Functions on Search Engines

The Knuth Morris Pratt Algorithm consists of several functions to carry out the process of matching and shifting patterns to the text in the database. Some of the functions used in the KMP Algorithm in the implementation of search engines for zakat information are as follows:

ISSN 2723-6471

Function preKMP

```
function preKMP($pattern){
      $i = 0;
      $j = $lompat[0] = -1;
      while($i<count($pattern)){
        while($j>-1 && $pattern[$i]!=$pattern[$j]){
          $j = $lompat[$j];
        }
        $i++;
        $j++;
        if(isset($pattern[$i])&&isset($pattern[$j])){
          if($pattern[$i]==$pattern[$j]){
            $lompat[$i]=$lompat[$j];
          }else{
            $lompat[$i]=$j;
          }
        }
      }
return $lompat;
   }
```

Figure. 2. Function preKMP

The preKMP function will determine the shift in the matching process. Where the match will start from the first index in the pattern against the text stored in the database. The mismatch between the pattern position index and the text will be used as information for shifting. The preKMP function has a pattern variable parameter with the declaration that the text index value is 0 and the pattern index value will be filled with the jump variable value when it is indexed 0.

As long as the condition of the text index value is less than the pattern index value count and as long as the pattern index value is greater than -1 and the pattern index is not the same as the text index, information will be obtained regarding the number of shift jumps that will be carried out with the value based on the location of the index value when it occurs. incompatibility.

Function KMPSearch

```
function KMPSearch($p,$t){
      $hasil = array();
      // pattern dan text dijadikan array
      $pattern = str_split($p);
      $text    = str_split($t);
      // hitung tabel lompatan dengan preKMP()
      $lompat = $this->preKMP($pattern);
      //print_r($lompat);
      $i = $j = 0;
      $num=0;
      while($j<count($text)){
        if(isset($pattern[$i]) && isset($lompat[$i])){
          while($i>-1 && $pattern[$i]!=$text[$j]){
            $i = $lompat[$i];
          }
        }else{
          $i = 0;
}
$j++;
        if($i>=count($pattern)){
          $hasil[$num++]=$j-count($pattern);
          if(isset($lompat[$i])){
            $i = $lompat[$i];
          }
        }
      }
      return $hasil;
   }
```

Figure. 3. Function KMPSearch

The KMPSearch function will return the last character index of the pattern found in the string. The KMPSearch function performs branching with conditions as long as the pattern index value (j) is less than the enumeration of the number of indexes in the text, then checks through the isset() command on the pattern and jump variables whether they have been set or not, where the isset function will return false if both variables (pattern and jump) contains null and returns true if both variables (pattern and jump) have been defined.

And as long as the text index value is not the same as the pattern index value, the pattern index value will be filled in by the value of the jump variable and outside of these conditions (the text index matches the pattern index) then the index value i is 0. And the matching continues with iterations on the index values i++ and j++. And if during the condition that the text index value is greater than the number of pattern index counts, then the result variable value is the text index with the number of pattern index counts.

4) Testing the Knuth Morris Pratt Algorithm on Search Engine Systems

After the implementation stage and writing functions on the KMP Algorithm, then the testing stage is carried out on the application of the search engine system. The process of testing the KMP Algorithm is carried out through several steps as follows:

Input Pattern

The first step in the KMP algorithm testing process that must be done is to enter a pattern. Where pattern is a search query entered by the user. Pattern will be used in the process of matching and shifting the existing text in the database. The pattern input process will be carried out during the KMP Algorithm testing process.

Knuth Morris Pratt Algorithm Performance Analysis

In the Knuth Morris Pratt algorithm, the initial process carried out is inputting the pattern, where the pattern in question is the search query desired by the user. Furthermore, the KMP algorithm will initialize the existing pattern, such as the length of the pattern and determine the index of the pattern and text (data in the database). And then shifting and matching will be done from the pattern to the text.

Table. 1. KMP Algorithm Shift Process

Text :	b	a	c	k	b	o	n	e
	0	1						
Pattern	b	o	n	e				
			0					
Iteration 2			b	o	n	e		
				0				
Iteration 3				b	o	n	e	
					0	1	2	3
Iteration 4					b	o	n	e

4. Discussion

In the implementation of the Knuth Morris Pratt Algorithm, it displays the search process by matching a string that has been matched between the inputted pattern and the description field from the information and activity table contained in your typical database. The following are the results of searching for information on zakat and social activities using the KMP algorithm, which can be seen in the following table.

Table. 2. KMP Execution Performance on Zakat Information

No	*Pattern*	*Number of descriptions found*	*KMP Search Function Execution Time (ms)*
1.	Zakat	12	0.03
2.	Sedekah	23	0.03
3.	Kikir	4	0.02
4.	Islam	4	0.02
5.	Memberi	4	0.03
		Amount	0.13
		Average	0.026

From table 3.2 KMP Execution Performance on Zakat Information, it can be seen that the average execution time of the KMP algorithm is 0.026 ms with the amount of text data in the information table database as many as 99 data which are matched according to the pattern.

Table. 3. KMP Execution Performance on Social Activities

No	*Pattern*	*Number of descriptions found*	*KMP Search Function Execution Time (ms)*
1.	UMKM	1	0.02
2.	Beasiswa	1	0.02
3.	Lazismu Baturraden	2	0.03
4.	Kajian	6	0.03
5.	Ahad	6	0.02
		Amount	0.12
		Average	0.024

From table 3.3 KMP Execution Performance on Social Activities, it can be seen that the average execution time of the KMP algorithm is 0.024 ms with the amount of text data in the activity table database as many as 8 data which are matched according to the pattern entered by the user.

Thus the results of the search performance of the KMP algorithm function shown by the table with the amount of zakat information data in the database as much as 99 and data on social activities in the database as much as 8 can be presented in the form of a bar chart of the performance of the KMP algorithm as follows:

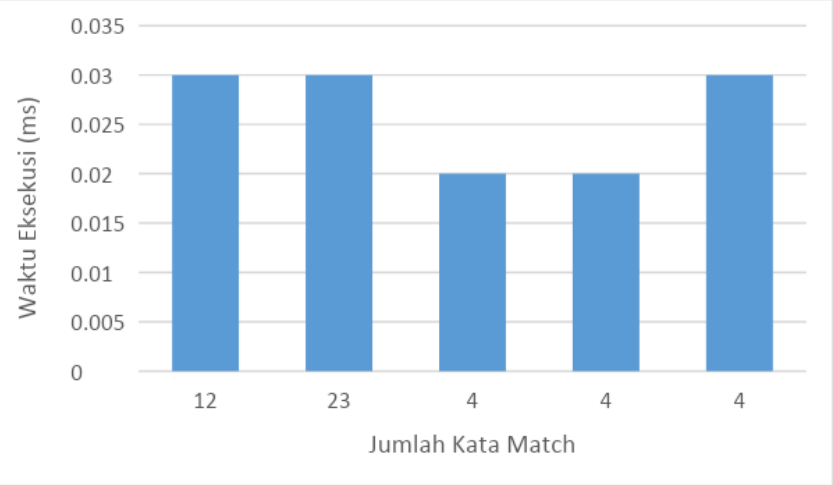


Figure. 4. KMP Algorithm Performance Diagram on Information

Figure 4 shows the performance of the KMP algorithm on the information table seen through the execution time of matching the pattern with the text in the information database. Where the average execution time is 0.026 ms against 99 data.

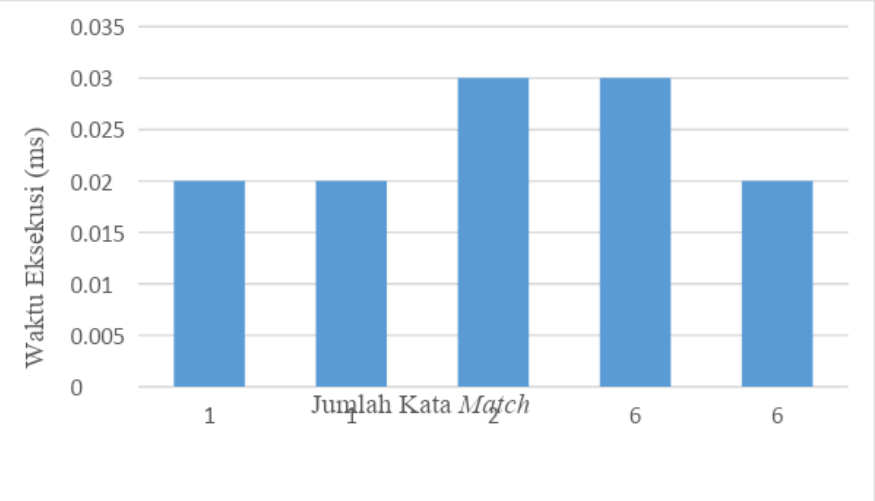


Figure. 5. KMP Algorithm Performance Diagram on Activities

Figure 5 shows the performance of the KMP algorithm on the activity table as seen through the execution time of matching the pattern with the text in the activity database. Where the average execution time is 0.024 ms for 8 data.

5. Conclusion

Based on the results of the analysis and discussion that has been carried out in the implementation of the Knuth Morris Pratt (KMP) Algorithm on search engines for zakat information and social activities, the following conclusions can be drawn: The implementation of the Knuth Morris Pratt Algorithm (KMP) on search engines has good and fast performance in the pattern matching process with text in the database and the Knuth Morris Pratt Algorithm (KMP) will display search results in the form of data that has the same word as the pattern and data containing the word like the characters in the pattern. The Knuth Morris Pratt (KMP) algorithm has the right search results according to the search pattern entered by the user. The search results are text that has the same word as pattern and text that contains words like pattern. The Knuth Morris Pratt (KMP) algorithm has a good performance with the performance of match execution time and search results on the information table an average of 0.026 ms

with search execution time data of 0.03 ms, 0.03 ms, 0.02 ms, 0.02 ms and 0.03 ms. And the execution time of matching and searching results on the activity table averaged 0.024 ms with data on search execution times of 0.02 ms, 0.02 ms, 0.03 ms, 0.03 and 0.02 ms.

References

[1] A. R. Hakim, K. Nasution, O. K. Sulaiman, and M. Z. Siambaton, "Perancangan Aplikasi Skripsi Online Menggunakan Algoritma String Matching Knuth Morris-Pratt Pada Fakultas Teknik Universitas Islam Sumatera Utara," J. Ilmu Komput. dan Inform., vol. 03, no. 2598–6341, pp. 98–107, 2019.

[2] M. Syarif, "Implementasi Algoritma String Matching Dalam Pencarian Surah Dan Ayat Dalam Al-Quran Berbasis Web," Indones. J. Netw. Secur., vol. 6, no. 2, pp. 70–76, 2017.

[3] A. B. Vavrenyuk, V. V Makarov, and V. A. Shurygin, "The substring search algorithm in the biotechnological processes simulation," Int. J. Pure Appl. Math., vol. 118, no. 5, pp. 701–709, 2018.

[4] T. H. Sa'diah, "Implementasi Algoritma Knuth-Morris-Pratt Pada Fungsi Pencarian Judul Tugas Akhir Repository," J. Komputasi, vol. 14, no. 1, pp. 115–125, 2017.

[5] M. R. Hossen, M. S. Azam, and H. K. Rana, "Performance Evaluation of Various DNA Pattern Matching Algorithms Using Different Genome Datasets," Accel. world's Res., vol. 3, no. 1, 2018, doi: 10.5281/zenodo.2580651.

[6] R. Y. Tsarev, A. S. Chernigovskiy, E. A. Tsareva, V. V. Brezitskaya, A. Y. Nikiforov, and N. A. Smirnov, "Combined string searching algorithm based on knuth-morris- pratt and boyer-moore algorithms," IOP Conf. Ser. Mater. Sci. Eng., vol. 122, no. 1, 2016, doi: 10.1088/1757-899X/122/1/012034.

[7] Nuraini and B. Firmansyah, "IMPLEMENTASI ALGORITMA KNUTH MORRIS PRATH UNTUK KAMUS TERJEMAHAN DIGITAL ACEH – BAHASA INDONESIA BERBASIS WEB," J. Nas. Inform., vol. 1, no. 1, pp. 66–75, 2020.

[8] M. M. Y. Daeli and R. K. Hondro, "Perancangan Aplikasi Pencarian Kata dengan Kombinasi Algoritma Knuth Morris Pratt dan Algoritma Boyer Moore," Maj. Ilm. INTI, vol. XII, no. 2, pp. 271–275, 2017, [Online]. Available: https://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/380/362.

[9] A. F. Qur'ani, "Penerapan Algoritma String Matching Knuth-Morris- Pratt Untuk Mencari Modifikasi Pada Kode," Makal. IF2211 Strateg. Algoritm. Inst. Teknol. Bandung, 2021.

[10] W. Astuti, "Analisis String Matching Pada Judul Skripsi Dengan Algoritma Knuth-Morris Pratt (Kmp)," Ilk. J. Ilm., vol. 9, no. 2, pp. 167–172, 2017, doi: 10.33096/ilkom.v9i2.136.167-172.

[11] C. B. G. Maldonado, M. S. Penas, and M. V. L. Lopez, "Negative Selection and Knuth Morris Pratt Algorithm for Anomaly Detection," IEEE Lat. Am. Trans., vol. 14, no. 3, pp. 1473–1479, 2016, doi: 10.1109/TLA.2016.7459637.

[12] K.-J. Lin, Y.-H. Huang, and C.-Y. Lin, "Efficient Parallel Knuth-Morris-Pratt Algorithm for Multi-GPUs with CUDA BT - Advances in Intelligent Systems and Applications - Volume 2," in IOP Conference Series: Earth and Environmental Science, 2013, pp. 543–552.

[13] R. Rahim, I. Zulkarnain, and H. Jaya, "A review: Search visualization with Knuth Morris Pratt algorithm," IOP Conf. Ser. Mater. Sci. Eng., vol. 237, no. 1, pp. 1–6, 2017, doi: 10.1088/1757-899X/237/1/012026.

[14] R. Y. Tsarev, A. S. Chernigovskiy, E. A. Tsareva, V. V. Brezitskaya, A. Y. Nikiforov, and N. A. Smirnov, "Combined string searching algorithm based on knuth-morris- pratt and boyer-moore algorithms," IOP Conf. Ser. Mater. Sci. Eng., vol. 122, no. 1, 2016, doi: 10.1088/1757-899X/122/1/012034.

[15] S. Aygün, E. O. Güneş, and L. Kouhalvandi, "Python based parallel application of Knuth-Morris-Pratt algorithm," in 2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), 2016, pp. 1–5, doi: 10.1109/AIEEE.2016.7821801.

[16] L. S. Riza, M. I. Firmansyah, H. Siregar, D. Budiana, and A. Rosales-Pérez, "Determining strategies on playing badminton using the Knuth-Morris-Pratt algorithm," Telkomnika (Telecommunication Comput. Electron. Control., vol. 16, no. 6, pp. 2763–2770, 2018, doi: 10.12928/TELKOMNIKA.v16i6.11554.

[17] D. Shapira and A. Daptardar, "Adapting the Knuth–Morris–Pratt algorithm for pattern matching in Huffman encoded texts," Inf. Process. Manag., vol. 42, no. 2, pp. 429–439, 2006, doi: https://doi.org/10.1016/j.ipm.2005.02.003.

[18] U. Ependi and N. Oktaviani, "Abstract Keyword Searching with Knuth Morris Pratt Algorithm," Sci. J. Informatics, vol. 4, no. 2, pp. 150–157, 2017, doi: 10.15294/sji.v4i2.9797.

[19] P. Gawrychowski, A. Jeż, and Ł. Jeż, "Validating the Knuth-Morris-Pratt Failure Function, Fast and Online," Theory Comput. Syst., vol. 54, no. 2, pp. 337–372, 2014, doi: 10.1007/s00224-013-9522-8.

[20] L. S. Riza, A. B. Rachmat, Munir, T. Hidayat, and S. Nazir, "Genomic repeat detection using the Knuth-Morris-Pratt algorithm on R high-performance-computing package," Int. J. Adv. Soft Comput. its Appl., vol. 11, no. 1, pp. 94–111, 2019.