International Journal Of Information System & Technology
Vol. 1, No. 1, (2017), pp. 8-21

Implementation and Analysis Zhu-Takaoka Algorithm and Knuth-Morris-Pratt Algorithm for Dictionary of Computer Application Based on Android

Handrizal[1], Andri Budiman[2], Desy Rahayu Ardani[3]
[1] *AMIK Tunas Bangsa, Pematangsiantar, Sumatera Utara, Indonesia*
[2,3] *University of Sumatera Utara, Sumatera Utara, Indonesia*
handrizal_tanjung@yahoo.com, mandrib@usu.ac.id,
desy.rahayu.ardani@gmail.com

Abstract

The string matching algorithm is the one of the most important parts in the various processes related to data and text types, which is the word search on computer dictionary. Computers have a basic role in the field of education, especially in teaching and learning activities. So that the classical learning model, that is by using the book as learning resource can be boring. To make it easier for users who searching words, we made an offline dictionary application based on Android by applying Zhu-Takaoka algorithm and Knuth-Morris-Pratt algorithm. The performance of Zhu-Takaoka is doing a search starts from the end of pattern that is tailored to the text, but in Knuth-Morris-Pratt algorithm starts from the beginning of pattern till match which the pattern used is word searched. The result of this research indicates that the Zhu-Takaoka algorithm is faster than the Knuth-Morris-Pratt algorithm which showed the running time of each algorithm.

Keywords**:** *matching, string, Zhu-Takaoka, Knuth-Morris-Pratt, pattern, text*

1. Introduction

Over time, the development of computers and Information Technology from time to time continues to grow. Computers now have penetrated in all areas of life, so inevitably we have to deal with the computerized world. Just as we communicate, in the computer world also has its own term that many we do not know. Computers have a basic role in the field of education, especially in teaching and learning activities. So that the classical learning model, that is by using the book as learning resource can be boring. Computers can increase the interest of learners because learners prefer the new things especially those using equipment such as computers, so to increase the interest of learners, need to be made an application that can help in learning, one of which is with the creation of a digital dictionary application that can facilitate participants educated and can be taken anywhere.

2. Rudimentary

2.1. String Matching

Strings can be words, phrases, or sentences. String matching is a very important subject in the wider domain of text processing. The result of a string search in the document depends on the technique and the way string matching is used [1].
To perform this process required two types of conditions, they are:

1. The length of the string p should be less than equal to the length of the string t. In this case $p \leq t$. (p = pattern, t = text).
2. The length of the string which is used as the search source must be longer than p [2].

ISSN : 2580-7250

Figure 1. Text dan String [3]

2.2. String Matching Algorithm

String matching algorithm is a basic component in implementing various existing practical software. String matching is used to find one or more strings called patterns (strings that will be matched into text) in strings called text (inputed strings) [1].
The performances of string matching are:

1. Moves the text with in the shift table window which has same size as the pattern
2. Takes *window* on the beginning of the text
3. Compare by characters on window with characters on the pattern.
4. After matching (either result match or not match), shift to right on window. This procedure is performed repeatedly until the window is at the end of the text. This mechanism is called the sliding-window mechanism [1].

2.3. Zhu-Takaoka Algorithm

The Zhu and Takaoka (ZT) algorithm is a variation of the Boyer-Moore Algorithm having the same characteristics in string searching (from right to left) and using the good suffix [4]. These characteristics are divided into two phases: preprocessing phase and search phase. The difference between the Boyer-Moore Algorithm and the Zhu-Takaoka Algorithm lies in the determination stage of the bad character rule. In Boyer-Moore, bad characters consist of only one-dimensional arrays, while in Zhu-Takaoka modified into two-dimensional arrays [5].
The characteristics of Zhu-Takaoka algorithm are:

1. Development of Boyer-Moore algorithm
2. Using two-dimensional array to calculate the value of shift.
3. Matching from right to left

Zhu and Takaoka algorithms are designed which can do a shift by considering bad-character shifts for two consecutive text characters. During the search phase the comparison is done from right to left and when the window is positioned on the text factor y [j ... j + m-1] and the mismatch occurs between x [mk] and y [j + mk] while x [mk + 1 .. m-1] = y [j + mk + 1 .. j + m-1] the shift is done with bad-character shift for text characters y [j + m- 2] and y [j + m-1]. A well-shifted table equation is also used to calculate shifts [1].

In preproccesing the pattern P by calculating the fringe function (in other literature calling overlap function, failure function, etc.) indicating the largest possible s shift by using the comparison formed before the string search. The fringe function depends only on the characters in the pattern, and not on the characters in the text being searched. Therefore, we can perform the initial function calculation before the string search is performed.

2.4. Knuth-Morris-Pratt Algorithm (KMP)

KMP string searching algorithmsearches for occurrences of a "word" W within a main "text string" T by employing the observation that when a mismatch occurs, the word itself embodies sufficient information to determine where the next match could begin, thus bypassing re-examination of previously matched characters.[6].

This algorithm finds all occurrences of a pattern of length n in the text of length m with the time complexity of O (m + n). This algorithm requires only O (n) space of

internal memory if text is read from external file. All O's quantities do not depend on the size of the alphabet space [7]. Systematically, the steps of the Knuth-Morris-Pratt algorithm when matching strings are as follows:

1. The Knuth-Morris-Pratt algorithm begins to match the pattern at the beginning of the text.
2. From left to right, this algorithm will match the characters per character pattern, with characters in the corresponding text until one of the following conditions is met:
   a. The characters in the pattern and in the compared text do not match (mismatch).
   b. All the characters in the pattern match. Then the algorithm will notify the discovery of this position.
3. The algorithm then shifts the pattern according to the next table, and then counts step 2 until the pattern is at the end of the text.

2.5. Android

Android is an operating system for linux-based mobile devices that includes operating systems, middleware, and applications. Android provides an open platform for developers to create apps [8].

3. Research and Methodology

3.1. Problem Analysis

The problem of this research is to know the performance comparison of each string matching algorithm, the Knuth-Morris-Pratt algorithm and the Zhu-Takaoka algorithm based on the time difference of the running time in the dictionary of health term. The problems in this study were identified using Ishikawa diagrams. The problems in this study can generally be shown on the ishikawa diagram in Figure 2.

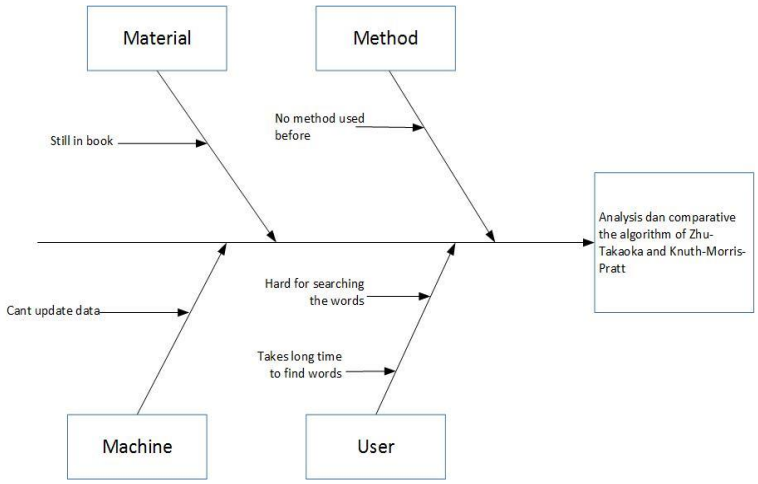

Figure 2. *Ishikawa* Diagram of Problem Analysis

In Figure 2. there are four categories in the comparative research of the Knuth-Morris-Pratt algorithm and the Zhu-Takaoka algorithm. The several causes of the four categories are as follows:

1. The user category has difficulties in searching words so that the time spent in searching takes time to be relatife long.
2. Categories of methods, there is no method to be used by comparing the complexity of two algorithms namely the Zhu-Takaoka and Knuth-Morris-Pratt algorithms.
3. Category of material, the dictionary in circulation is still in print form as the book has a shortage because it is made of paper that is easily torn and damaged,

4. Machine category is data that can not be updated because it still uses dictionary in the form of book.

3.2. Analysis Processing

Process analysis is designed with the aim to describe the conditions of all processes that occur and provide an overview of the parts of the system designed. The design of this system is made with several diagrams including Flowchart, Use Case Diagram, Activity Diagram, Sequence Diagram, and Class Diagram.

3.2.1. System Flowchart

The general system flowchart in this application is shown in Figure 3. where the flowchart describes how the system works from the beginning of the system is opened to completion. At the start of the search the user can immediately start the search by entering the word in the search box field. Then the word will be used as a pattern and then sent to the class KMP or ZT class, depending on what type of algorithm will be used by the user. Then the data in the database will be converted into data in the form of an arraylist to then be sent also on the class KMP or class ZT as text. The word entered then used as a pattern will be compared with data from the database that has been used as an arraylist as text one by one with what type of algorithm used by user. If the input pattern is found in the text, the text is displayed in the list form below the Search Box field, otherwise it is not displayed. Figure 3. Illustrates the search flowchart in this application:

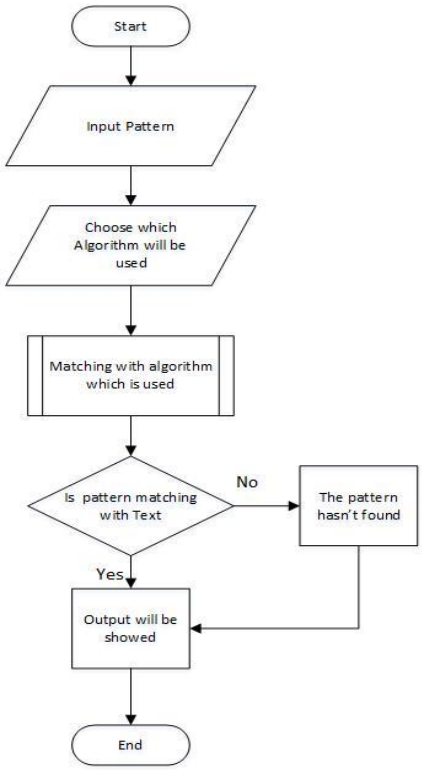

Figure 3. System Flowchart

3.2.2. Use Case Diagram

Use case diagram is the functionality of a system, so that customers or users of the system understand and understand the usefulness of the system to be built. In Figure 4. describes activities that can be performed by the user, where users can search the health term by determining the Knuth-Morris-Pratt algorithm or the Zhu-Takaoka algorithm to

be selected first. After that, the user inputs the word to be searched and will check the term is or does not exist in the health dictionary application. From the word search results will record the time and display the time (running time) and the number of words found to the system screen for each algorithm and if the search word is not found then the system will display notification (result not found).

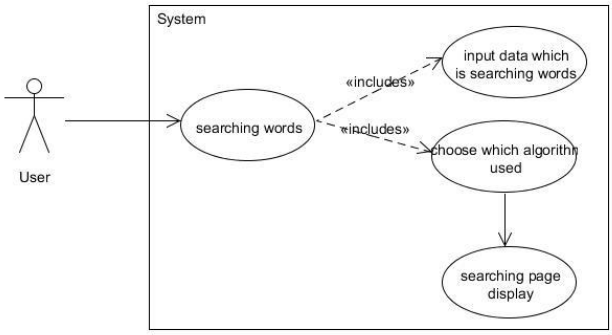

Figure 4. ***Use Case Diagram***

3.2.3. System Flowchart of Zhu-Takaoka Algorithm

In Figure 5. is a flowchart of the string search steps of the Zhu-Takaoka algorithm, in case of a mismatch it will be shifted based on the greatest value between bmGs [i] and ztbc [y [j + m-2]] [y [j + m -1]]. Whereas if a match is found then the shift is done by the value i of bmGs at position 0 or value i = 0.

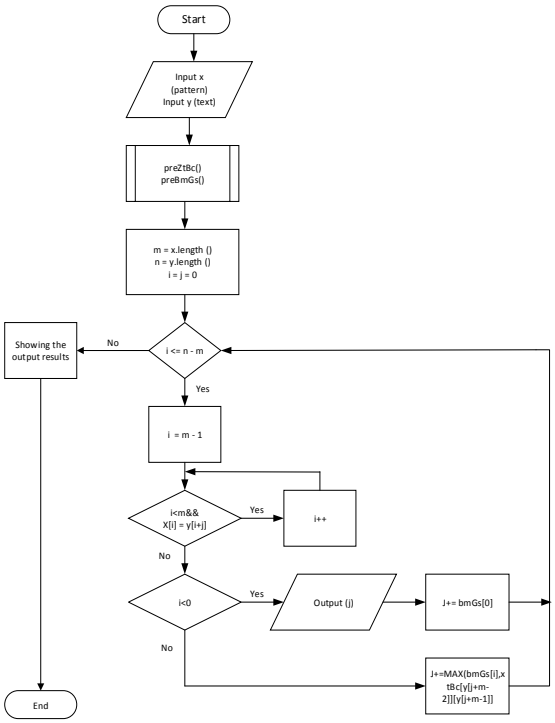

Figure 5. Flowchart of Zhu-Takaoka Algorithm

3.2.4. Flowchart of Knuth-Morris-Pratt Algorithm

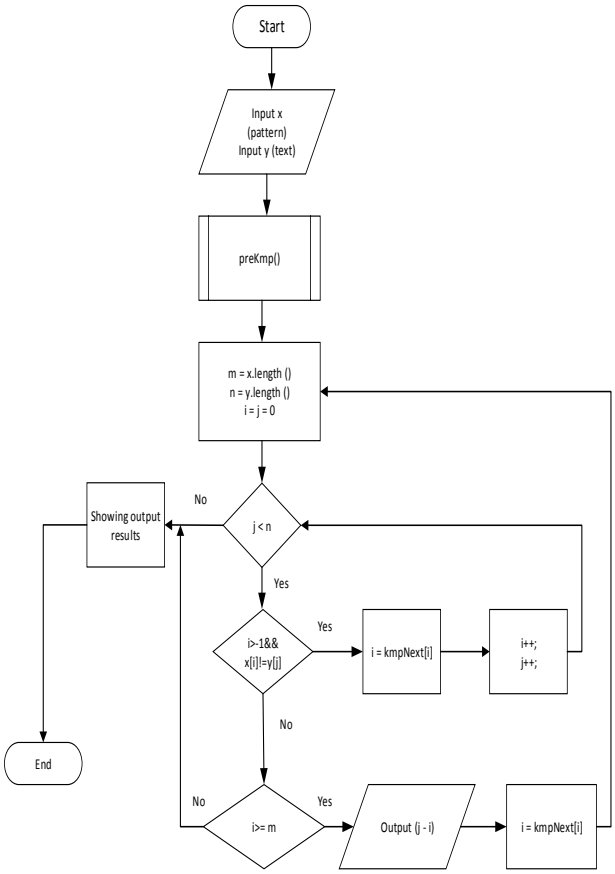

Figure 6. ***Flowchart of Knuth-Morris-Pratt Algorithm***

4. Results and Discussion

4.1. System Implementation

In the implementation process of this system there are 3 (three) activity page view, which is main activity main menu as welcome activity, and main activity of ZT algorithm and KMP algorithm as text search menu. While the buttons contained in the main menu as well as the search menu there is 1 button "search" is used to start a text search according to the word entered.

4.2. System Testing

System testing is done to ensure the system after it is built can run well in accordance with the analysis and design that has been designed. The main focus of this research is to examine how the system performs string search on health terms using the Zhu-Takaoka and Knuth-Morris-Pratt algorithms

4.2.1. Search String Test with Zhu-Takaoka Algorithm

The way to search the text is to input the word into the search engine and select the "ZT" algorithm by pressing the "Search" button then the system will automatically display the search results. This process is shown in Figure 7.

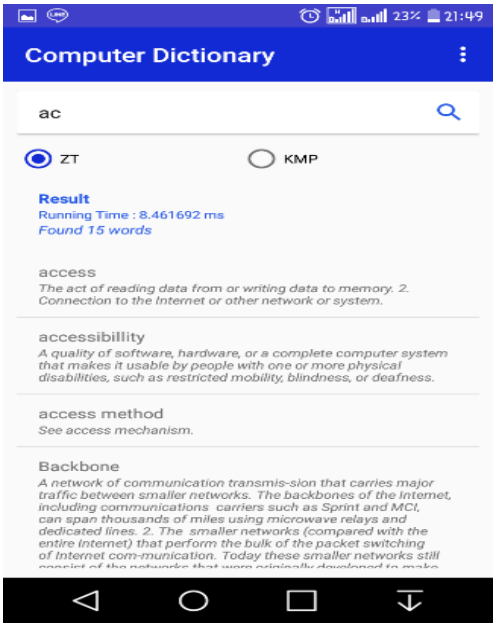

Figure 7. String Search in Zhu-Takaoka Algorithm Page Display 1

In Figure 7. the search results using the "ZT" algorithm with the "av" pattern resulted in running time of 32.425156 ms and the number of words found was 5 words.

4.2.2. Search String Test with Knuth-Morris-Pratt Algorithm

The way to search the text is to input the word into the search engine and select the "KMP" algorithm by pressing the "Search" button then the system will automatically display the search results. This process is shown in Figure 8.

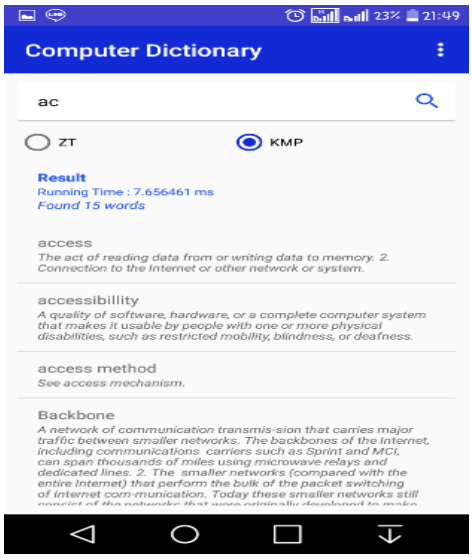

Figure 8. String Search in Knuth-Morris-Pratt Algorithm Page Display

In Figure 8. the search results using the "KMP" algorithm with the "av" pattern resulted in running time of 33.40446 ms and the number of words found was 5 words.

4.3. The Test Results

Test results from this research are running time of word search and word count found in Algorithm of Zhu-Takoka and Knuth-Morris-Pratt Algorithm done to different strings.

Table 1. Test Result of Zhu-Takaoka Algorithm and Knuth-Morris-Pratt Algorithm on Computer Dictionary Application

No	Pattern	*Running time (ms)*		Finding word result	Result
		ZT	KMP		
1	i	10.51	8.65	365	*Match*
2	in	10.00	12.62	57	*Match*
3	inf	8.97	10.41	4	*Match*
4	info	12.24	11.59	1	*Match*
5	infor	10.50	10.43	1	*Match*
6	inform	9.68	11.60	1	*Match*
7	informa	7.81	10.49	1	*Match*
8	informat	9.05	10.99	1	*Match*
9	informati	14.76	15.88	1	*Match*
10	informatii	10.28	9.84	0	*Mismatch*
Total		103.8	112.5		
Average		10.38	11.25		

The results of the test shown in Table 1 show that the total comparative results of the two algorithms is the Zhu-Takaoka algorithm has a low total Running Time value compared to the Knuth-Morris-Pratt algorithm, where the Zhu-Takoka algorithm is faster for word matching than the algorithm Knuth-Morris-Pratt.

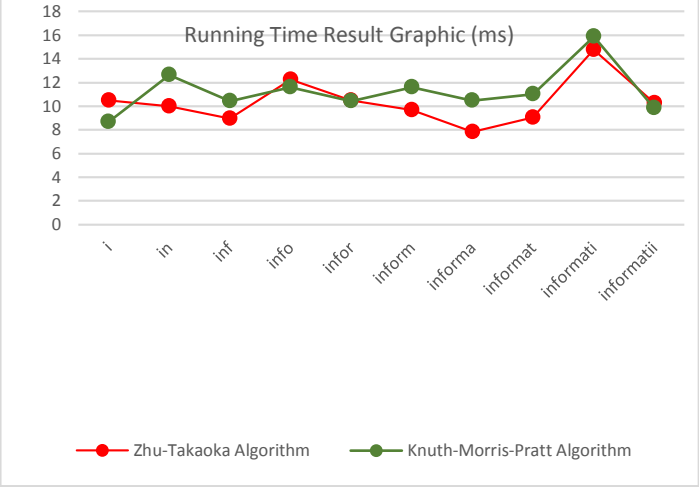

Figure 9. Comparison Chart Results Running Time Algorithm Zhu-Takaoka and Knuth-Morris-Pratt Algorithm

From the graph above it can be explained that the speed of each algorithm depends on the number of words found in the dictionary. In the Zhu-Takaoka (ZT) algorithm, the required running time will be lower than the Knuth-Morris-Pratt (KMP) algorithm based

on the length of the pattern searched, while the running time on the KMP algorithm will be higher if the input pattern is more than 1 text character.

4.4. The Complexity of Time

The complexity of time is done by seeking the big theta of the existing pseudocode. This pseudocode is part of a program that has been designed in accordance with the Zhu-Takaoka algorithm and the Knuth-Morris-Pratt algorithm. The calculation results from the time complexity of the two algorithms using the average case / big theta notation are presented in Table 2.

4.4.1. The Time Complexity of Knuth-Morris-Pratt Algorithm

Table 2. Time Complexity in Preprocessing of Knuth-Morris-Pratt Algorithm

Program Code	C	#	C#
Integer[] table = **new** Integer[W.**length**];	C_1	1	C_1
int pos = 2;	C_2	1	C_2
int cnd = 0;	C_2	1	C_2
table[0] = -1; table[1] = 0;	C_3	1	C_3
while(pos < W.**length**) {	C_4	m	C_4m
if(W[pos-1] == W[cnd]) {	C_5	m	C_5m
table[pos] = cnd;	C_3	m	C_3m
cnd += 1;	C_6	m	C_6m
pos += 1;	C_7	m	C_7m
} **else if**(cnd > 0)	C_8	m	C_8m
cnd = table[cnd];	C_6	m	C_6m
else {	C_9	m	C_9m
table[pos] = 0;	C_3	m	C_3m
pos += 1;	C_7	m	C_7m
}} **return** table;}	C_{10}	1	C_{10}

From the calculation of running time in the table then obtained:

$T(n) = (C_1 + 2C_2 + C_3 + 2C_4 + C_{10}) m^0 + (2C_3 + C_4 + C_5 + 2C_6 + 2C_7 + C_8 + C_9) m^1$

$T(n) = m^0 + m^1$

$T(n) = \theta(m)$

Table 3. Time Complexity of Knuth- Morris-Pratt Algorithm

Program Code	C	#	C#
char[] W = pattern.toCharArray();	C_1	1	C_1
char[] S = source.toCharArray();	C_1	1	C_1
if(W.length == 0)	C_2	1	C_2
return 0;	C_3	1	C_3
if(S.length == 0)	C_2	1	C_2
return -1;	C_3	1	C_3
int m = 0;	C_4	1	C_4
int i = 0;	C_4	1	C_4
Integer[] T = *createTable*(S);	C_5	1	C_5
while(m+i < S.length) {	C_6	n	C_6 n
if(W[i] == S[m+i]) {	C_2	n	C_2 n
if(i == W.length-1)	C_2	n	C_2 n

return m;	C_3	n	C_3 n
i += 1;	C_7	n	C_7 n
} else {	C_8	n	C_8 n
m = (m+i - T[i]);	C_9	n	C_9 n
if(T[i] > -1)	C_2	n	C_2 n
i = T[i];	C_7	n	C_7 n
else	C_8	n	C_8 n
i = 0; }}	C_7	n	C_7 n
return -1; }	C_3	1	C_3

$T(n) = (2C_1 + 2C_2 + 3C_3 + 2C_4 + C_5) n^0 + (2C_2 + C_3 + C_6 + 3C_7 + 2C_8 + C_9) n^1$

$T(n) = n^0 + n^1$

$T(n) = \theta(n)$

In the Knuth-Morris-Pratt algorithm, the preprocessing phase is T (n) = θ (m) and the search phase has T (n) = θ (n). Then the complexity of the Knuth-Morris-Pratt algorithm is T (n) = θ (m + n).

4.4.2. The Complexity Time of Zhu-Takaoka Algorithm

Table 4. Complexity of PreZtbc Function in Zhu-Takaoka Algorithm

Program Code	C	#	C#
int i, j, m = x.length,z= ztBc.length;	C_1	1	C_1
for (i = 0; i < z; ++i)	C_2	m	C_2m
for (j = 0; j < z; ++j)	C_2	mn	C_2mn
ztBc[i][j] = m;	C_3	mn^2	C_3mn^2
for (i = 0; i < z; ++i)	C_2	m	C_2m
ztBc[i][x[0]] = m - 1;	C_3	m	C_3m
for (i = 1; i < m - 1; ++i)	C_2	m	C_2m
ztBc[x[i - 1]][x[i]] = m -1-i; }	C_3	m	C_3m

$T(n) = (C_1 + 2C_2 m + 2 C_3 m + C_2 mn + 2C_3 mn^2$

$T(n) = \theta(mn^2)$

Table 5. Complexity of Suffixes Function in Zhu-Takaoka Algorithm

Program Code	C	#	C#
int f = 0, g, i, m = x.length;	C_1	1	C_1
suff[m - 1] = m;	C_2	1	C_2
g = m - 1;	C_3	1	C_3
for (i = m - 2; i >= 0; --i) {	C_4	m	C_4 m
if (i > g && suff[i + m - 1 - f] < i - g)	C_5	m	C_5 m
suff[i] = suff[i + m - 1 - f];	C_2	m	C_2 m
else {	C_6	m	C_6 m
if (i < g)	C_5	m	C_5 m
g = i;	C_3	m	C_3 m
f = i;	C_7	m	C_7 m

while (g >= 0 && x[g] == x[g + m - 1 - f])	C_8	m	C_8 m
--g;	C_9	m	C_9 m
suff[i] = f - g; }}}	C_2	m	C_2 m

$T(n) = (C_1 + C_2 + C_3)m^0 + (2C_2 + C_3 + C_4 + 2C_5 + C_6 + C_7 + C_8 + C_9) m^1$

$T(n) = \theta(m)$

Table 6. Complexity of PreBmGs Function in Zhu-Takaoka Algorithm

Program Code	C	#	C#
int i, j, m = x.length;	C_1	1	C_1
int[] suff = new int[m];	C_1	1	C_1
suffixes(x, suff);	C_2	1	C_2
for (i = 0; i < m; ++i)	C_3	m	C_3m
bmGs[i] = m;	C_4	m	C_4m
j = 0;	C_5	m	C_5m
for (i = m - 1; i >= 0; --i)	C_3	m	C_3m
if (suff[i] == i + 1)	C_6	m	C_6m
for (; j < m - 1 - i; ++j)	C_3	m^2	C_3m^2
if (bmGs[j] == m)	C_6	m^2	C_6m^2
bmGs[j] = m - 1 - i;	C_4	m^2	C_4m^2
for (i = 0; i <= m - 2; ++i)	C_3	m	C_3m
bmGs[m - 1 - suff[i]] = m - 1 - i; }	C_4	m	C_4m

From the calculation of running time table above:

$T(n) = (2 C_1 + C_2)m^0 + (3C_3 + 2C_4 + 2C_5 + C_6) m^1 + (C_3 + C_4 + C_6) m^2$

$= m^0 + m^1 + m^2$

$T(n) = \theta(m^2)$

Table 7. Complexity of BM Function Zhu-Takaoka Algorithm

Program Code	C	#	C#
int[][] ztBc = new int[256][256];	C_1	1	C_1
int[] bmGs = new int[m];	C_1	1	C_1
preZtBc(x, ztBc);	C_2	1	C_2
preBmGs(x, bmGs);	C_2	1	C_2
j = 0;	C_3	1	C_3
while (j <= n - m) {	C_4	m	C_4m
i = m - 1;	C_5	m	C_5m
while (i >= 0 && x[i] == y[i + j])	C_4	mn	C_4mn
--i;	C_6	m	C_6m
if (i < 0) {	C_7	m	C_7m
result.add(j);	C_8	1	C_8

j += bmGs[0];	C_9	m	C_9m
} else if(j + m - 2 >= 0) {	C_{10}	m	$C_{10}m$
j += Math.*max*(bmGs[i], ztBc[y[j + m - 2]][y[j + m - 1]]);	C_9	m	C_9m
} else	C_{11}	m	$C_{11}m$
j += bmGs[i];}	C_9	m	C_9m
return result;}	C_{12}	1	C_{12}

$$T(n) = (2C_1 + 2C_2 + C_3 + C_8 + C_{12})m^0 + (C_4 + C_4 n + C_5 + C_6 + C_7 + 2C_9 \; C_{10} + C_{11})m^1 + C_4 mn$$
$$= m^0 + m^1 + mn$$

$$T(n) = \theta(mn)$$

In the Zhu-Takaoka algorithm, the pre-process phase ztBc has T (n) = Θ (mn^2), Suffix T phase (n) = Θ (m), the pre-process phase of bmGs has T (n) = Θ (m^2), and the search phase has T (n) = Θ (mn). Then the complexity of the Zhu-Takaoka Algorithm is Θ (mn).

5. Conclution

From the design and manufacture to the program testing with Zhu-Takaoka algorithm and Knuth-Morris-Pratt algorithm, it can be obtained some conclusions and suggestions for further program development.

5.1. Conclution

From the research conducted, the conclusions that can be taken are:

1. The application of string matching algorithm that is Zhu-Takaoka algorithm and Knuth-Morris-Pratt algorithm is by doing the implementation into a dictionary application of health term so that it can be known to optimize the use of both algorithm by doing system test.
2. Comparative results in the study show that the Zhu-Takaoka algorithm is faster than the Knuth-Morris-Pratt algorithm seen based on the running time used in the word search process and depending on the number of words found with the average Zhu-Takaoka algorithm result and the Knuth-Morris-Pratt algorithm because the workings of these two algorithms are very different. In the Zhu-Takaoka algorithm the word search is performed by shifting characters based on ztBc and bmGs, while Knuth-Morris-Pratt performs the search process by using fringe functions.
3. The time complexity of the Zhu-Takaoka algorithm is Θ (mn), and the Knuth-Morris-Pratt algorithm is Θ (m + n), where to calculate the fringe functions, this algorithm takes time Θ (m) and for string searching requires time Θ (n).

5.2. Suggestions

The following are suggestions that can be used for the development phase of this system's research, among others:

1. It is suggested to further development to compare performance with other string matching algorithms such as Smith Waterman, Approxiamate string matching, and Rabin Karp algorithm to find out which string matching algorithm has the fastest time complexity.P
2. Further development can create dictionaries online so that data can be updated automatically on the user application without having to re-install the application.

References

[1] Charras, C. & Lecroq, T. 2004. Handbook of Exact String-Matching Algorithms. London: King's College Publications.

[2] Abu-Zaid, Ibrahim M. & Emad Kh. El-Rayyes. 2012. Parallel Search Using KMP Algorithm in Arabic String. Int. J. of Science and Technology. Vol.2 No.7 : pp. 2224-3577.
[3] Wayne,K. & Sedgewick, R. 2014. Algorithms Fourth Edition: Substring Searc.
[4] ZHU, R.F. and TAKAOKA, T. 1987. On Improving the Average Case of The Boyer-Moore String Matching Algorithm. Journal of Information Processing 10(3) pp.173 177.
[5] Sumit K. Aggarwal, 2005. Analysis of string-searching algorithms on biological sequence databases. Research Communications, Vol. 89, No. 2 pp. 368-374.
[6] Singla, Nimisha & Deepak Garg. 2012. String Matching Algorithms and Their Applicability in various Application, Int. J. of Soft Computing and Engineering. Vol.I pp. 2231-2307.
[7] R.Sadli, 2001. Optimization of sequence queries in database systems, Proceeding PODS '01 Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 71-81..
[8] Margaret B, 2011. Android: Changing the Mobile Landscape, IEEE pervasive Computing Vol. 10 No. 1 pp 4-7.

Authors

1st Author
Handrizal
Lecturer of AMIK Tunas Bangsa Pematangsiantar
handrizal_tanjung@yahoo.com

2nd Author
Andri Budiman
Lecturer Of University of Sumatera Utara, *Sumatera Utara, Indonesia*
mandrib@usu.ac.id

3rd Author
Desy Rahayu Ardani
University of Sumatera Utara, *Sumatera Utara, Indonesia*
desy.rahayu.ardani@gmail.com