Searching Process with Raita Algorithm and its Application

Robbi Rahim*, 1, Ansari Saleh Ahmar2, Dahlan Abdullah3, Dedy Hartama4, Darmawan Napitupulu5, Andysah Putera Utama Siahaan6, Muhammad Noor Hasan Siregar6, Nurliana Nasution7, Siti Sundari8 and S Sriadhi9

1School of Computer Engineering and Communication, Universiti Malaysia Perlis, Malaysia
2Departement of Statistics, Universitas Negeri Makassar, Indonesia
3Department of Informatics, Universitas Malikussaleh, Aceh, Indonesia
4Department of Information System, STIKOM Tunas Bangsa, Pematang Siantar, Indonesia
5Research Center for Quality System and Testing Technology, Indonesian Institute of Sciences, Indonesia
6Department of Management, Universitas Graha Nusantara, Indonesia
7Department of Informatics, Universitas Lancang Kuning, Pekanbaru, Indonesia
8Department of Informatics, Universitas Harapan Medan, Medan, Indonesia
9Department of Electrical Engineering, Universitas Negeri Medan, Indonesia

E-mail: *usurobbi85@zoho.com

Abstract. Searching is a common process performed by many computer users, Raita algorithm is one algorithm that can be used to match and find information in accordance with the patterns entered. Raita algorithm applied to the file search application using java programming language and the results obtained from the testing process of the file search quickly and with accurate results and support many data types.

1. Introduction
Searching[1 - 4] on the computer is a common process performed by computer users, before computer technology develops like this searching process is perform on the archive organized with the number documents that much and also searching process not easy to do and require a lot of time[5], with the presence of computer technology[5 - 7] right now it makes easier for human to search files because it’s already in digital form and also indexed by the computer[8 - 9]. A result from searching process will appear in a few seconds due to the speed of the computer and amount of data. The searching process on the computer can be achieved using available tools on the operating system[6], [10], the length of time searching process on the operating system depends on the process of indexing from operating system to speed up searching process, Raita algorithm [11 – 12] were applied for searching a file using a keyword that represents name of the file from operating system.

The results of this paper is applied Raita algorithm in a fast file search application using java language. R. Rahim, et all[1], [2], [13] explains the search process can be done using many algorithms such as Knuth Morris Pratt to search for a particular word or file, or also by using double hashing techniques to make the search process faster by creating the main table and overflow table, based on
previous experiment from other researcher the searching process can be done for various type of data you want. Raita algorithm is not a new algorithm, Patel [11] with his finding describe a different result of many searching algorithm with advantage and weakness, based of his result this paper is to prove that Raita algorithm is much more fastest than usual algorithm by implement in searching file. The Raita algorithm implemented in the file search application is supposed to be able to search file much faster than using ordinary file searching, one of the reasons that allows the search process faster because Raita algorithm has a procedure of matching and searching simultaneously and this makes the process time more faster.

2. Methodology
The RAITA algorithm has a good pattern in searching [11], [12], especially in mid-word character checks from given pattern by comparing the last character of both patterns (m-1) with the second character from length of the character. The middle character of the pattern is compared twice for the corresponding character of the text. This procedure is repeated until does not reach n-m + 1.
An analyzing RAITA algorithm for searching a file can be assumption as text searching, see an example process below.

P(Text) = ROBBIRAHIMKEREN
K (Pattern) = RAHIM

| i | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| a | R | A | H | I | M |

Based on the words to be tested made table BmBc by using equation formula: \( m - 2 \) which serves as pattern matching boundary \( m - 1 - i \).

\[ m = K = 5 \]
\[ m = \text{Pattern Length} \]
\[ m = P \]
\[ m = 5 - 2 = 3 \]

BmBc (a) = m - 1 - i
1) 5 - 1 - i (5 - 1 - 0) = 4, the value is put on index to 0 with character that is R
2) 5 - 1 - i (5 - 1 - 1) = 3, the value is placed on index to 1 with character that is A
3) 5 - 1 - i (5 - 1 - 2) = 2, the value is placed on the second index with the character of H
4) 5 - 1 - i (5 - 1 - 3) = 1, the value is placed on index to 3 with character that is I
5) 5 - 1 - i (5 - 1 - 4) = 0, the value is placed on index to 4 with character that is M

Based on the calculation of BmBc above obtained the following results:

| i | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| a | R | A | H | I | M |
| BmBc | 4 | 3 | 2 | 1 | 0 | 5 |

The value of m is 5 in accordance with the length of the pattern, because the alphabet that is not in the table then is initialized with the sign (*) and then the value corresponds to the length of the pattern. The searching process can be viewed step by step in tables 3 and table 4 by similar the character pattern, table 3 are the first table form Raita process.
Table 3. Text and sentence pattern phase 1.

| Text  | R | O | B | B | I | R | A | H | I | M | K | E | R | E | N |
|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pattern | R | A | H | I | M | R | A | H | I | M | K | E | R | E | N |

In the first stage the character I pattern does not match the M text, since the character M does not have BmBc, and then the pattern will advance as much as 5 characters. Next phase can see in table 4 below:

Table 4. Text and sentence pattern phase 1.

| Text  | R | O | B | B | I | R | A | H | I | M | K | E | R | E | N |
|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pattern | R | A | H | I | M | R | A | H | I | M | K | E | R | E | N |

In the second stage the character M pattern corresponds to the M text, since the M character exists in the BmBc table with the shift, then the process stops performed and is tested, the test is performed for all characters ranging from 0 - 4 according to the character position values in the BmBc table and all corresponding characters then the search process stops because the word has been found. Raita algorithm process above then applied to the application created by using Java programming language with Windows 8.1 platform and also can be applied to any operating system that provides java library, below is pseudo code from Raita algorithm applied to application.

```java
if (pattern.length == 1) {
    final int nLimit = Math.min(text.length, textEnd);
    for (int n = textStart; n < nLimit; n++) {
        if (text[n] == pattern[0])
            return n;
    }
    return -1;
}
else if (pattern.length == 2) {
    final int nLimit = Math.min(text.length, textEnd) - 1;
    for (int n = textStart; n < nLimit; n++) {
        if (text[n] == pattern[0]) {
            if (text[n + 1] == pattern[1])
                return n;
        }
    }
    return -1;
}
CharIntMap m = (CharIntMap) processed;
int i, j, k, mMinusOne;
char last, first;
i = pattern.length - 1;
mMinusOne = 1 - 1;
last = pattern[i];
first = pattern[0];
```
i += textStart;

while (i < textEnd) {
    if (text[i] == last && text[i - (pattern.length - 1)] == first) {
        k = i - 1;
        j = mMinusOne;

        while (k > -1 && j > -1 && text[k] == pattern[j]) {
            --k;
            --j;
        }

        if (j == -1) {
            return k + 1;
        }
    }

    i += m.get(text[i]);
}

return -1;

3. Result and Discussion

Searching application with Raita algorithm designed by using java language, and the trial of searching file are using keyword that represent a file name, figure 1 are the application interface when searching process already finish.

![Search result](image)

**Figure 1.** Search result.

Searching on a folder with name *makalah* containing 8.132 files with size 5.32 GB, searching process using Raita algorithm with character *a* and PDF file type obtained total files as 423
pdf files and it takes 3,650 millisecond. Implementation Raita algorithm for searching file were perform as much 5 times with same type and different keywords, Table 5 below is the result of the testing performed.

```
| No | Number of Files | Raita Application (Second) |
|----|-----------------|----------------------------|
| 1  | 432             | 3.650 ms                   |
| 2  | 519             | 4.121 ms                   |
| 3  | 812             | 7.712 ms                   |
| 4  | 1012            | 14.812 ms                  |
| 5  | 1200            | 56.139 ms                  |
```

Based on testing done with varying results, the search time is quite fast and with more files found will require a lot of time than usual process.

4. **Conclusion**

File search application by applying Raita algorithm to find file on operating system got good result with fast enough time, application only done limited to matching and searching process by using Raita algorithm and result obtained not compared with search by using windows search or also with the use of other algorithms, future development makes it possible to implement multiple algorithms in one search process and compare the results.

**References**

[1] R. Rahim, I. Zulkarnain, and H. Jaya, “A review: search visualization with Knuth Morris Pratt algorithm,” in *IOP Conference Series: Materials Science and Engineering*, 2017, vol. 237, no. 1, p. 12026.

[2] R. Rahim, I. Zulkarnain, and H. Jaya, “Double hashing technique in closed hashing search process,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 237, no. 1, p. 12027, Sep. 2017.

[3] S. J. Putra and I. Khalil, “Context for the intelligent search of information,” in *2017 5th International Conference on Cyber and IT Service Management (CITSM)*, 2017, pp. 1–4.

[4] R. Rahim, S. Nurarif, M. Ramadhan, S. Aisyah, and W. Purba, “Comparison Searching Process of Linear, Binary and Interpolation Algorithm,” *J. Phys. Conf. Ser.*, vol. 930, no. 1, p. 12007, Dec. 2017.

[5] M. Andrews, “Searching the internet,” *IEEE Softw.*, vol. 29, no. 2, pp. 13–16, 2012.

[6] J. G. Reed and P. M. Baxter, “Computer searching,” *Library use: Handbook for psychology (3rd ed.)*. pp. 27–37, 2003.

[7] G. G. S. Fiates, J. E. A. Fiates, F. A. R. Serra, and M. P. Ferreira, “Innovation environment in small technology-based companies,” *J. Technol. Manag. Innov.*, vol. 5, no. 3, pp. 81–95, 2010.

[8] R. Rahim, D. Hartama, H. Nurdiyanto, A. S. Ahmar, D. Abdullah, and D. Napitupulu, “Keylogger Application to Monitoring Users Activity with Exact String Matching Algorithm,” *J. Phys. Conf. Ser.*, vol. 954, no. 1, p. 12008, 2018.

[9] R. Rahim *et al.*, “Block Architecture Problem with Depth First Search Solution and Its Application,” *J. Phys. Conf. Ser.*, vol. 954, no. 1, p. 12006, 2018.

[10] R. Rahim, A. S. Ahmar, A. P. Ardyanti, and D. Noftiansyah, “Visual Approach of Searching Process using Boyer-Moore Algorithm,” *J. Phys. Conf. Ser.*, vol. 930, no. 1, p. 12001, Dec. 2017.

[11] U. Patel and M. Thakkar, “An Efficient Exact Single Pattern Matching Algorithm,” vol. 3, no. 5, pp. 1955–1958, 2014.

[12] H. N. Verma and R. Singh, “A fast string matching algorithm,” vol. 2, no. 6, pp. 1877–1883.
2011.

[13] R. Rahim, Nurjamiyah, and A. R. Dewi, “Data Collision Prevention with Overflow Hashing Technique in Closed Hash Searching Process,” *J. Phys. Conf. Ser.*, vol. 930, no. 1, p. 12012, Dec. 2017.