Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Perbandingan Algoritma Raita Dan Apostolico Crochemore Untuk Pencarian Data Perpustakaan Menggunakan Metode Eksponensial

Herman Krismon Simbolon

Ilmu Komputer Dan Teknologi Informasi, Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: Hermansimbolin10@gmail.com

Abstrak-Algoritma *Raita* dan *Apostolico Crochemore* merupakan algoritma pencocokan *string* yang dapat digunakan untuk mempermudah proses pencarian suatu *serch engine*. Kedua algoritma tersebut memiliki karakteristik pencarian yang berbeda untuk menemukan hasil pencocokan *string*. Algoritma yang memiliki kecepatan lebih optimal dalam melakukan pencocokan *string* tentunya akan lebih untuk digunakan.Dalam penelitian ini penulis melakukan perbandingan antara kedua algoritma tersebut menggunakan Metode Eksponensial agar mudah diketauhi metode yang lebih optimal dalam melakukan pencarian diantara kedua algoritma tersebut.Studi kasus dalam pencocokan *string* yang digunakan untuk membandingkan kedua metode tersebut yaitu dalam proses pencarian data perpustakaan yang akan penulis bangun menggunakan *Visual Basic Net* 2008.

Kata Kunci : Algoritma *Raita*; *Apostolico Crochemore*; Metode Eksponensial; Data Perpustakaan

Abstract-The Raita and Apostolico Crochemore algorithms are string matching algorithms that can be used to simplify the search process for a search engine. The two algorithms have different search characteristics to find string matching results. Algorithms that have more optimal speed in string matching will certainly be more useful. In this study, the authors compare the two algorithms using the Exponential Method so that it is easy to find out which method is more optimal in searching between the two algorithms. A case study in string matching is used to compare the two methods, namely in the process of searching for library data that the author will build using Visual Basic Net 2008.

Keywords : Raita Algorithm; Apostolico Crochemore; Exponential Methods; Library Data

1. PENDAHULUAN

Perpustakaan adalah sebuah ruangan, bagian sebuah gedung, ataupun gedung itu sendiri yang digunakan untuk menyimpan buku dan terbitan lainya yang biasanya disimpan menurut tata susunan tertentu untuk digunakan pembaca, bukan untuk dijual. Perpustakaan bertujuan memberikan layanan kepada pemustaka, meningkatkan kegemaran membaca, serta memperluas wawasan dan pengetahuan untuk mencerdaskan kehidupan bangsa.

Permasalahan yang umum terjadi pada proses pencarian data perpustakaan yaitu banyaknya waktu yang dibutuhkan untuk menemukan data yang ingin dicari karena faktor jumlah data yang sangat banyak, sehingga prosedur pencarian data pepustakaan menjadi tidak efesien. Oleh karena itu dibutuhkan sistem berbasis komputer yang dapat membantu pencari data perpustakaan untuk mencari bukutu yang diinginkan, yaitu aplikasi perpustakaan yang di dalamnya terdapat fitur pencarian dengan penerapan String Matching.

String Matching merupakan bagian penting dari sebuah proses pencocokan string di dalam fitur pencarian sebuah aplikasi. Terdapat banyak algoritma yang dapat digunkan dalam proses string matching, diantaranya yaitu algoritma Horspool, Zhu Takaoka, Raita, Apostolico Crochemore, Binary Search, KnuthMorris-Pratt, Berry Ravindran, dan lain-lain [1][2].

Algoritma String Matching yang akan diteliti dalam penelitian ini adalah algoritma Raita dan Apostolico Crochemore. Pemilihan algoritma Raita dan Apostolico Crochemore berdasarkan hasil penelitian dari peneliti terdahulu yang menunjukkan bahwa kedua algoritma ini memiliki kecepatan yang baik dalam melakukan pencocokan string. Pada penelitian sebelumnya algoritma Raita dan Apostolico Crochemore telah dibandingkan dengan algoritma lainnya dan dinyatakan lebih baik.

Pada tahun 2017 Dedi Rudi Bawanto dan Nidia Rosmawanti melakukan penelitian tentang perbandingan algoritma Binary Search dan Raita dalam pencarian data. Hasil penelitian menunjukkan bahwa algoritma Raita lebih cepat dari algoritma algoritma Binary Search [3]. Pada tahun 2019 Sri Anggreni T melakukan penelitian tentang perbandingan algoritma Raita dan algoritma Berry Ravindran dalam pencocokan string pada aplikasi kamus Indonesia-Korea. Hasil penelitian menunjukkan bahwa algoritma Raita lebih cepat dari algoritma Berry Ravindran [4]. Pada tahun 2019 Mamta Culkari Puding, Dkk. melakukan penelitian tentang perbandingan algoritma Raita dan algoritma Horspool pada aplikasi istilah psikologi berbasis android. Hasil penelitian menunjukkan bahwa algoritma Raita lebih cepat dibandingkan algoritma Horspool [5]. Kemudian pada tahun 2017 Husnil Khotimah Siregar juga melakukan penelitian tentang perbandingan algoritma Knuth Morris Pratt dan algoritma Apostolico Crochemore. Hasil penelitian menunjukkan bahwa Apostolico Crochemore lebih baik dari Knuth Morris Pratt [6].

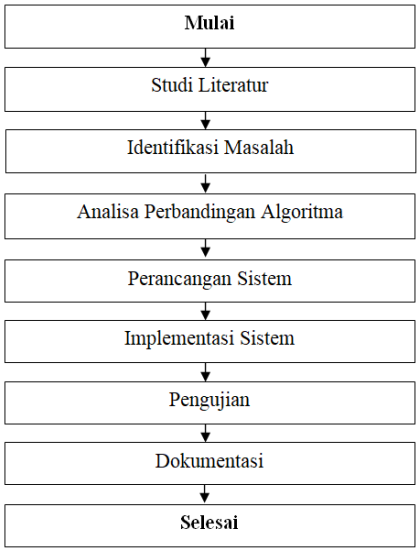
Berdasarkan hasil penelitian terdahulu yang telah diuraikan di atas maka pada penelitian ini penulis melakukan perbandingan algoritma Raita dan Apostolico Crochemore untuk mengetahui algoritma yang lebih baik untuk diterapkan dalam pencarian data perpustakaan. Dalam melakukan perbandingan antara kedua algoritma tersebut penulis menggunakan metode eksponensial agar mudah diketahui selisih waktu yang diperlukan untuk memberikan hasil pencarian.

Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

2. METODOLOGI PENELITIAN

2.1 Kerangka Kerja Penelitian

Kerangka kerja penelitian merupakan desain penelitian yang dirancang dengan tahapan yang jelas dan dapat menggambarkan alur penelitian yang akan dilakukan secara maksimal. Tujuan dari pembuatan kerangka kerja penelitian yaitu untu mempermudah penulis dalam menyelesaikan proses penelitian sesuai dengan target yang ingin diperoleh. Adapun kerangka kerja penelitian yang di gunakan oleh penulis dalam penelitian ini dapat dilihat pada gambar 1 di bawah ini :



Gambar 1. Kerangka Kerja Penelitian

Adapun keterangan dari kerangka kerja yang terdapat pada Gambar 1 di atas adalah sebagai berikut :

1. Studi Literatur
   Pada tahap ini penulis melakukan pengutipan kajian putaka terkait topik penelitian yang akan diteliti dalam penelitian ini, yaitu berupa kajian teoritis yang behubungan dengan perpustakaan, string matching, algoritma Raita, algoritma Apostolico-Crochemore, dan penelitian terkait yang dipublikasikan maksimal 5 tahun terakhir.
2. Identifikasi Masalah
   Pada tahap ini penulis melakukan identifikasi masalah pencocokan string dalam proses pencarian data perpustakaan dan bagaimana melakukan perbandingan antara algoritma Raita dengan algoritma Apostolico-Crochemore untuk mengetahui algoritma yang lebih baik.
3. Analisa Perbandingan Algoritma
   Pada tahap ini penulis melakukan analisa perbandingan algoritma Raita dengan algoritma Apostolico-Crochemore yang dilengkapi dengan penjelasan terkait tahapannya.
4. Perancangan Sistem
   Pada tahap ini penulis melakukan perancangan sistem menggunakan unified modeling language dan design interface system.
5. Impementasi Sistem
   Pada tahap ini penulis menerapkan algoritma Raita dan algoritma Apostolico-Crochemore ke dalam sistem.
6. Pengujian
   Pada tahap ini penulis melakukan pengujian terhadap hasil perbandingan algoritma Raita dengan algoritma Apostolico-Crochemore pada sistem yang telah dihasilkan.
7. Dokumentasi
   Pada tahap ini penulis melakukan pembuatan dokumentasi sistem mulai dari tahap awal hingga tahap pengujian sistem, untuk selanjutnya dibuat dalam bentuk laporan penelitian (skripsi).

2.2 Perpustakaan

Perpustakaan adalah sebuah ruangan, bagian sebuah gedung, ataupun gedung itu sendiri yang digunakan untuk menyimpan buku dan terbitan lainya yang biasanya disimpan menurut tata susunan tertentu untuk digunakan pembaca, bukan untuk dijual. Perpustakaan bertujuan memberikan layanan kepada pemustaka, meningkatkan kegemaran membaca, serta memperluas wawasan dan pengetahuan untuk mencerdaskan kehidupan bangsa". Sedangkan perpustakaan sekolah menurut Badan Standardisasi Nasional bertujuan "menyediakan pusat sumber belajar sehingga dapat membantu pengembangan dan peningkatan minat baca, literasi informasi, bakat serta kemampuan peserta didik [7].

2.3 Algoritma String Matching

String Matching merupakan suatu algoritma yang dapat di manfaatkan dengan tujuan meminimalisir waktu yang dibutuhkan dalam proses pencarian kata. *String Mathing* terbagi dari dua bagian, yaitu *Exact Matching* dan *Heuristic* atau *Statistical Matching*. Berdasakan klarifikasinya arah pergerakan untuk melakukan pencocokan *string*, algoritma *String Matching* dapat melakukan pencocokan *string* dari arah kiri ke kanan, kanan ke kiri, dan dari kedua arah.

Algoritma String Matching adalah algoritma untuk melakukan pencarian semua kemunculan string pendek P[0..n-1] yang disebut pattern di string yang lebih panjang T[0..m-1] yang disebut teks[8]. Algoritma String Matching merupakan suatu metode yang digunakan untuk menemukan suatu keakuratan atau hasil dari satu atau beberapa pola teks yang diberikan. String merupakan pokok bahasan yang penting dalam ilmu komputer karena teks merupakan adalah bentuk utama dari pertukaran informasi antar manusia, misalnya pada literatur, karya ilmiah, halaman web dan sebagainya[9]. Algoritma String Matching memiliki 3 (tiga) prinsip kerja untuk melakukan pencocokan string, yaitu :

1. Memindai teks dengan bantuan sebuah window yang ukurannya sama dengan panjang pattern.
2. Menempatkan window pada awal teks.
3. Membandingkan karakter pada window dengan karakter dari pattern.

Setelah pencocokan (baik hasilnya cocok atau tidak cocok), dilakukan shift ke kanan pada window. Prosedur ini dilakukan berulang-ulang sampai window berada pada akhir teks. Mekanisme ini disebut mekanisme Sliding-Window[10]. Secara garis besar teknik *String Matching* dalam pencocokan *string* dibedakan menjadi 2, yaitu :

1. *Exact String Matching*
   Exact String Matching merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama. Bagian algoritma ini bermanfaat jika pengguna ingin mencari *string* dalam dokumen yang sama persis dengan *string* masukan. Beberapa contoh algoritma *Exact String Matching* antara lain:
   a. *Brute Force*
      Brute Force membandingkan karakter per karakter sampai ditemukannya pola yang dicari dari awal *string* sampai dengan akhir *string*
   b. *Knuth-Morris-Pratt*
      Knuth-Morris-Pratt membandingkan karakter per karakter sampai ditemukannya pola yang dicari dari awal string sampai dengan akhir *string* dari arah kiri ke arah kanan
   c. *Boyer Moore*
      Boyer Moore melakukan pencocokan karakter dari kanan ke kiri.
2. *Inexact String Matching*
   Inexact String Matching merupakan pencocokan *string* secara samar, maksudnya pencocokan *string* dimana *string* yang dicocokkan memiliki kemiripan dimana keduanya memiliki susunan karakter yang berbeda tetapi *string-string* tersebut memiliki kemiripan baik kemiripan tekstual atau penulisan atau kemiripan ucapan (*Phonetic String Matching*). Metode *Inexact String Matching* diarahkan untuk mencari nilai dari beberapa *string* yang mendekati dan tidak hanya menghasilkan cocok atau tidak cocok[11].

2.4 Algoritma Raita

Raita adalah alagoritma pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama[12]. *Raita* merancang sebuah algoritma dengan membandingkan karakter yang terakhir dari pola yang diawali dari karakter paling kanan dari *window*. Jika terjadi kecocokan, kemudian karakter pertama dari pola teks paling kiri dari *window* juga dibandingkan. Jika terjadi kecocokan, maka akan dibandingkan karakter tengah pola dengan karakter teks tengah *window*. Pada akhirnya, jika mereka benar-benar cocok, maka algoritma membandingkan karakter lain mulai dari pola karakter kedua ke karakter kedua terakhir, dan akan membandingkan dengan karakter tengah lagi[13].

Prosedur *Raita* dalam melakukan pencocokan *string* terdiri dari 5 (lima) tahapan. Berikut tahapan prosedur *Raita* dalam melakukan pencocolan *string* :

1. Buat tabel pergeseran pola yang dicari sebagai kata yang akan dicari pada teks.
2. Jika dalam proses pembandingan terjadi ketidakcocokan antara pasangan karakter pada akhir pola dengan karakter teks, pergeseran dilakukan sesuai nilai karakter teks pada tabel *BmBc*
3. Jika dalam proses pembandingan akhir pola terjadi ketidakcocokan lagi maka karakter akan digeser lagi sesuai tabel *BmBc*
4. Jika karakter akhir pola dengan karakter pada teks yang sedang dibandingkan cocok, maka posisi karakter pada pola dan teks akan memiliki nilai nol (0), dan dilanjutkan pencocokan pada karakter awal pola. Jika cocok maka dilanjutkan pencocokan dengan karakter tengah pola.
5. Jika akhir, awal dan tengah pola telah cocok. Pencocokan dilanjutkan dengan bagian kanan dari awal karakter pada pola, jika cocok maka dicocokkan pada bagian kanan tengah pola[13].

2.5 Algoritma *Apostolico Crochemore*

Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Algoritma *Apostolico-Crochemore* adalah salah satu jenis algoritma pencarian *string* dari banyak jenis algoritma pencarian *string*. Algoritma *Apostolico-Crochemore* adalah algoritma sederhana yang melakukan perbandingan karakter pada teks sebanyak 3n/2 pada kasus terburuknya. Algoritma *Apostolico-Crochemore* terdiri dari dua fase, yaitu fase proses awal (*Preprocessing*) dan fase pencarian *string*. Pada fase proses awal Algoritma *Apostolico-Crochemore* dilakukan fungsi pinggiran untuk menentukan jumlah langkah pergeseran *pattern* terbesar dengan menggunakan perbandingan sebelum pencarian *string*. Pada fase pencarian *string* dilakukan perbandingan *pattern* pada teks. Algoritma *Apostolico-Crochemore* memiliki cara kerja yang mirip dengan algoritma *Knuth-Morris-Path (KMP)*. Fungsi pinggiran (*Border Function*) yang digunakan mirip seperti fungsi pinggiran pada algoritma *Knuth-Morris-Path (KMP)* dan proses pencariannya sama-sama dimulai dari kiri ke kanan. Akan tetapi, tahapan pencarian (Proses Urutan Pembandingan) berbeda dengan algoritma *KMP* [14].

2.5.1 Fase Proses Awal

Fase proses awal (*Preprocessing*) terhadap *pattern* x adalah dengan menghitung fungsi pinggiran yang mengindikasikan pergeseran x terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum fase pencarian *string*. Dengan adanya fungsi pinggiran ini dapat dicegah pergeseran yang tidak berguna, seperti hal nya pada algoritma *Brute Force*. Fungsi pinggiran hanya bergantung pada karakter-karakter di dalam *pattern*, dan bukan pada karakter-karakter di dalam teks. Fungsi pinggiran pada algoritma *Apostolico-Crochemore* adalah *kmpNext*[14].

2.5.2 FasePencarian String

Proses pencarian (Pencocokan) *pattern* dengan teks pada Algoritma *Apostolico-Crochemore* adalah sebagai berikut. Misalkan :
x adalah *string* dari *pattern* yang akan dicari
y adalah teks yang akan dicocokan dengan *pattern*
m adalah panjang x
n adalah panjang y

I= 0 jika x adalah karakter tunggal yang dipangkatkan ($x = c^m$ dengan c di dalam S) dan l adalah posisi karakter pertama dari x yang berbeda dari x[0] ($x = a^l$buuntuk a, b di dalam S , u di dalam S^* , dan a ¹ b). Setiap pembandingan dilakukan dengan posisi yang berpola dengan urutan sebagai berikut : l , l +1, ..., m-2, m-1, 0, 1, ..., l -1.
Selama fase pencarian, kita mempertimbangkan (i, j, k)di mana:
- jendela karakter diposisikan pada teks y[j .. j+m-1]
- 0 £ k £ l dan x[0 .. k-1] = y[j .. j+k-1]
- l £i < m dan x[l ..i-1] = y[j+ l .. i+j-1]

Inisialisasi awal dari (i, j, k) adalah (l , 0, 0).

Komputasi (Perhitungan) untuk menentukan (i, j, k) berikutnya mempertimbangkan tiga kasus yang bergantung pada nilai i. Ketiga kasus tersebut sebagai berikut :

1. i = l
   Jika x[i] = y[i+j] maka (i, j, k) berikutnya adalah (i+1, j, k).
   Jika x[i] ≠ y[i+j] maka (i, j, k) berikutnya adalah (l , j+1, max{0, k-1}).
2. l < i < m Jika x[i] = y[i+j] maka (i, j, k) berikutnya adalah (i+1, j, k).
   Jika x[i] ≠ y[i+j] maka ada dua kasus yang muncul yang bergantung pada nilai kmpNext[i] :
   - kmpNext[i] ≤ l maka (i, j, k) berikutnya (l , i+j-kmpNext[i], max{0, kmpNext[i]}).
   - kmpNext[i] > l maka (i, j, k) berikutnya (kmpNext[i], i+j-kmpNext[i], kmpNext[i] l).
3. i = m
   Jika k < l dan x[k] = y[j+k] maka (i, j, k) berikutnya adalah (i, j, k+1).
4. Sebaliknya salah satu dari k < l dan x[k] ≠ y[j+k], atau k = l (jika k = l kemunculan x diberitahukan) pada kedua kasus tersebut perhitungan (i, j, k) berikutnya sama seperti perhitungan pada kasus l < i < m[14].

2.6 Metode Eksponensial

Metode Perbandingan Eksponensial adalah salah satu metode untuk menentukan urutan prioritas alternatif keputusan dengan kriteria jamak. Dalam menghitung dan membandingkan proses pencarian dari kedua algoritma tersebut adalah sebagai berikut:

1. Menentukan alternatif
   Untuk menganalisa perbandingan kecepatan antara dua algorima berbeda dalam melakukan pencarian maka perlu dilakukan penentuan algortima yang mana yang akan digunakan sebagai algoritma pencarian.
2. Menentukan kriteria
   Untuk dapat mmebandingkan kedua alternatif tersebut, maka selanjutnya perlu dilakukan penentuan kriteria dalam menganlisa proses dan cara kerjanya.
3. Menentukan bobot kriteria

Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Penentuan bobot merupakan salah satu komponen yang sangat berpengaruh terhadap nilai analisa, untuk itu menetepkan bobot kriteria berdasarkan tingkatan pengaruh dalam menentukan kecepatan dalam melakukan pencarian.

4. Pemberian Nilai Pada Setiap Kriteria
   Pada kriteria yang telah dibentuk harus diberikan nilai. Nilai tersebut dapat dilihat pada contoh dibawah ini yang dimana nilainya diambil berdasarkan analisa algoritma sebelumnya
5. Menghitung Nilai
   Setelah melakukan pengisian nilai terhadap masing-masing kriteria, maka proses berikutnya adalah malakukan perhitungan dengan menggunakan rumus dari Metode Perbandingan Eksponensial (MPE)[15].

3. HASIL DAN PEMBAHASAN

Mengatasi permasalan di atas pada penelitian ini penulis melakukan perbandingan terhadap dua Algoritma *String Matching*, yaitu *Raita* dan *Apostolico Crochemore*.Algoritma *Raita* dan *Apostolico Crochemore* merupakan algoritma pecocokan *string* yang dapat digunakan untuk mempermudah proses pencarian suatu mesin pencarian. Kedua algoritma tersebut memiliki karakteristik pencarian yang berbeda untu menemukan hasil pencocokan *string*. Algoritma yang memiliki kecepatan lebih optimal dalam melakukan pencocokan *string* tentunya akan lebih untuk digunakan. Pada penelitian ini dilakukan perbandingan antara kedua algoritma tersebut menggunakan metode ekponensial agar mudah diketahui metode yang lebih optimal dalam melakukan pencarian diantara kedua algoritma tersebut. Implementasi kedua algoritma tersebut dilakukan pada sistem pencarian data perpustakaan yang akan penulis bangun menggunakan *Visual Basic Net* 2008.

3.1 Penerapan Algoritma Raita

Algoritma *Raita* melakukan pencocokan *string* dengan membandingkan karakter yang terakhir dari pola yang diawali dari karakter paling kanan dari "jendela". ika mereka cocok, kemudian karakter pertama dari pola teks paling kiri dari jendela juga dibandingkan. Jika mereka cocok, maka akan dibandingkan karakter tengah pola dengan karakter teks tengah jendela. Pada akhirnya, jika mereka benar-benar cocok, maka algoritma membandingkan karakter lain mulai dari pola karakter kedua ke karakter kedua terakhir, dan akan membandingkan dengan karakter tengah lagi. Sebagai contoh kasus pada penelitian ini penulis menerapkan algoritma *raita* untuk pencocokan *pattern* "DASAR" dengan text "DESAIN GRAFIS DASAR". Maka diketahui perhitungannya sebagai berikut :
Text = DESAIN GRAFIS DASAR
Pattern = DASAR
diketahui bahwa :
Text = Teks yang akan dicari
Pattern = Panjang pola
Maka :
Pattern = 5
Selanjutnya dilakukan pembuatan tabel *BmBc* untuk melakukan perhitungan dengan persamaan sebagai berikut:
Pattern – 2 .....(1)
5 – 2 = 3
Mencari nilai *BmBc* (a)
Pattern – 1 – i ..... (2)
Pencarian nilai *BmBc (a)* dengan menggunakan rumus persamaan berikut ini :

Tabel 1. Perhitungan tabel *BmBc*

I	0	1	2	3	4	*
A	D	A	S	A	R	
BmBc(a)	4	3	2	1	5	5

5 – 1 – 0 = 4 maka nilai diletakkan pada posisi indeks ke-0 dengan krakter D
5 – 1 – 1 = 3 maka nilai diletakkan pada posisi indeks ke-1 dengan karakter A
5 – 1 – 2 = 2 maka nilai diletakkan pada posisi indeks ke-2 dengan karakter S
5 – 1 – 3 = 1 maka nilai diletakkan pada posisi indeks ke-3 dengan karakter A

Nilai R adalah 5 sesuai dengan panjang pola, karena abjad yang tidak ada pada tabel maka diinisialisasikan dengan tanda (*) kemudian nilainya sesuai dengan panjang pola. Jadi, untuk perhitungan Algoritma raita sesuai tabel *BmBc* adalah sebagai berikut :

Tabel 2. Hasil *BmBc* (a)

A	D	A	S	A	R	*
BmBc	4	3	2	1	5	5

Langkah selanjut adalah melakukan pencarian menggunakan algoritma *Raita* dengan tahap-tahap berikut ini :

Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

1. Tahap pertama

Tabel 3. Proses Pencarian Pada Teks Ke-1

Teks	D	E	S	A	I	N		G	R	A	F	I	S		D	A	S	A	R
Pattern	D	A	S	A	R														

Pada proses diatas dinyatakan terjadi ketidakcocokan pada teks, I tidak terkandung di dalam *pattern*, maka dilakukan pergeseran sebanyak 5 langkah sesuai dengan nilai *BmBc**.

2. Tahap kedua

Tabel 4. Proses Pencarian Pada Teks ke-2

Teks	D	E	S	A	I	N		G	R	A	F	I	S		D	A	S	A	R
Pattern						D	A	S	A	R									

Pada proses diatas dinyatakan terjadi ketidakcocokan pada teks, A terkandung di dalam *pattern*, maka dilakukan pergeseran sebanyak 1 langkah sesuai dengan nilai *BmBc*A.

3. Tahap ketiga

Tabel 5. Proses Pencarian Pada Teks Ke-3

Teks	D	E	S	A	I	N		G	R	A	F	I	S		D	A	S	A	R
Pattern							D	A	S	A	R								

Pada proses diatas dinyatakan terjadi ketidakcocokan pada teks, F tidak terkandung di dalam *pattern*, maka dilakukan pergeseran sebanyak 5 langkah sesuai dengan nilai *BmBc**. Lakukan hingga semua pola memiliki kecocokan dengan teks. Maka pencarian berhenti pada tahap yang ke-6.

3.2 Penerapan Algoritma Apostolico Crochemore

Pada fase *PreprocessingApostolico Crochemore* dilakukan perhitungan jumlah langkah pergeseran *pattern* terbesar yang mungkin terjadi dengan menggunakan perbandingan yang dibentuk sebelum fase pencarian *string*. Fase ini menggunakan tabel pergeseran *kmpNext*. Untuk mendapatkan nilai pergeseran, *kmpNext[i]* untuk indeks 0 diberi nilai -1 sebagai inisialisasi. Selanjutnya, bandingkan karakter pada indeks 0 dengan karakter pada indeks 1. Jika karakternya sama, maka *kmpNext[1]* bernilai -1 Namun apabila karakternya berbeda pada saat melakukan proses pencocokan *string* maka *kmpNext[1]* bernilai 0. Sebagai contoh kasus pada penelitian ini penulis menerapkan algoritma *raita* untuk pencocokan *pattern* "DASAR" dengan text "DESAIN GRAFIS DASAR". Maka diketahui perhitungannya sebagai berikut :

Text = DESAIN GRAFIS DASAR
Pattern = DASAR

Tabel 6. *KmpNext*

I	0	1	2	3	4
x[i]	D	A	S	A	R
kmpNext[i]	-1	0	0	0	0

I = 1

Tahap pencocokan karakter menggunakan algoritma *Apostolic-Crochemore* dapat dilihat pada table berikut

Tabel 7. Tahap Ke-1

Teks	D	E	S	A	I	N		G	R	A	F	I	S		D	A	S	A	R
Pattern		1																	
	D	A	S	A	R														

Berdasarakan pencocokan karakter pada tahap 1 tidak ditemukan kecocokan karakter terhadap "A" dengan "E", maka *Pattern* digeser sebanyak : 1-*kmpNext[1]* = 1-0 = 1 langkah

Tabel 8. Tahap Ke-2

Teks	D	E	S	A	I	N		G	R	A	F	I	S		D	A	S	A	R
Pattern			1																
		D	A	S	A	R													

Berdasarakan pencocokan karakter pada tahap 2 tidak ditemukan kecocokan karakter terhadap "A" dengan "S", maka *Pattern* digeser sebanyak : 1-*kmpNext[1]* = 1-0 = 1 langkah

Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Tabel 9. Tahap Ke-3

Teks	D	E	S	A	I	N		G	R	A	F	I	S		D	A	S	A	R
Pattern				1	2														
			D	A	S	A	R												

Berdasarakan pencocokan karakter pada tahap 2 ditemukan kecocokan karakter terhadap "A" dengan "A", maka dilanjutkan pencocokan anatara karakter terhadap "S" dengan "I" namun tidak terjadi kecocokan, maka *Pattern* digeser sebanyak : 1-*kmpNext[1]* = 1-0 = 1 langkah

Tabel 10. Tahap Ke-4

Teks	D	E	S	A	I	N		G	R	A	F	I	S		D	A	S	A	R
Pattern					1														
				D	A	S	A	R											

Berdasarakan pencocokan karakter pada tahap 4 tidak ditemukan kecocokan karakter terhadap "A" dengan "I", maka *Pattern* digeser sebanyak : 1-*kmpNext[1]* = 1-0 = 1. Lakukan pencocokan karakter hingga ditemukan kecocokan karakter terhadap "A" dengan "A", S" dengan "S", "A" dengan "A", "R" dengan "R", dan "D" dengan "D". Maka dapat proses pencocokan karakter dengan *pattern* diberhentikan pada langkahke-14.

3.3 Penerapan Metode Eksponensial

Penerapan metode eksponensial adalah tahap yang dilakukan untuk membandingkan hasil kerja algortima *raita* dengan algoritma *Apostolico Crochemore* dalam melakukan pencocokan string untuk menyelesaika masalah pencarian data perpustakaan yang menjadi topik pada penelitian ini Adapun tahap membandingkan proses pencarian dari algoritma *Raita* dan *Apostolico Crochemore* menggunakan metode eksponensial adalah sebagai berikut:

1. Menentukan alternatif
   Untuk menganalisa perbandingan kecepatan antara algoritma *Raita* dan *Apostolico Crochemore* dalam melakukan pencarian data perpustakaan maka perlu dilakukan penentuan algortima yang mana yang akan digunakan sebagai algoritma pencarian.
2. Menentukan kriteria
   Untuk dapat membandingkan kedua alternatif tersebut, perlu ditentukan kriteria dalam analisis proses dan cara kerjanya. Kriteria tersebut dapat dilihat pada tabel di bawah ini:

Tabel 11. Kriteria

Kriteria	Keterangan
Ukuran Memory	Perhitungan penggunaan memori terjadi di ketika algoritma melakukan pencocokan string
Durasi Waktu	Perhitungan waktu diperoleh ketika algoritma melakukan pencocokan string dengan Pertandingan dari awal sampai akhir

3. Pembobotan Kriteria
   Pembobotan kriteria merupakan salah satu komponen yang memiliki pengaruh besar terhadap nilai suatu analisis karena menentukan bobot kriteria berdasarkan tingkat pengaruhnya dalam menentukan kecepatan pencarian. Bobot kriteria dapat dilihat pada tabel di bawah ini:

Tabel 12. Pembobotan Kriteria

Kriteria	Prioritas Kriteria	Bobot
Total penggunaan memori	50%	0,5
Waktu yang dibutuhkan	50%	0,5

4. Penentuan Nilai Pada Setiap Kriteria
   Kriteria yang akan ditetapkan perlu diberikan nilai. Nilai ini dapat dilihat pada contoh di bawah ini di mana nilai diambil berdasarkan analisis sebelumnya dari *Raita* dan *Apostolico Crochemore*.

Tabel 13. Penentuan Nilai Pada Setiap Kriteria

Alternatif	Pattern	Kriterian	
		Memory (Byte)	Waktu (ms)
Raita	Dasar	678	0,474105
Apostolico Crochemore	Dasar	456	0,374105

5. Menghitung Nilai

Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Setelah melengkapi nilai untuk masing-masing kriteria, proses selanjutnya adalah konversi menggunakan rumus dari Exponential Comparative Method (MPE). Adapun proses perhitungan untuk perbandingan *Raita* dan *Apostolico Crochemore* adalah sebagai berikut:

Nilai Raita = $(678)^{0.5}$+$(0.474105)^{0.5}$
= 26.72698596
Nilai AC = $(456)^{0.5}$+$(0.374105)^{0.5}$
= 21.96579774

6. Menghitung nilai prioritas keputusan
Total Nilai Raita = 26.72698596
Total Nilai AC = 21.96579774
Berdasarkan perhitungan nilai yang telah dilakukan di atas maka peroleh hasil perhitungan analisa perbandingan *Raita* dan *Apostolico Crochemore* menggunakan MPE pada tabel 4.28 berikut ini

Tabel 14. Perhitungan Analisa Perbandingan Menggunakan MPE

Kriteria						Total Nilai Raita	Total Nilai *Apostolico Crochemore* (AC)
Memory			Waktu				
B	Raita N	AC N	B	Raita N	AC N		
0.5	678	456	0.5	0.474105	0.374105	26.72698596	21.96579774

Keterangan :
B = Bobot
Raita = Algoritma *Raita*
AC = Algoritma *Apostolico Crochemore*
N = Nilai Keriteria
Total Nilai = $\sum(N)^B$

7. Menentukan Hasil Perbandingan
Setelah didapatkan nilai akhir atau total dari setiap alternatif, langkah selanjutnya adalah memprioritaskan keputusan berdasarkan nilai dari masing-
Masing alternatif. Hasil keputusan prioritas dapat dilihat pada table berikut ini:

Tabel 15. Hasil Perbandingan

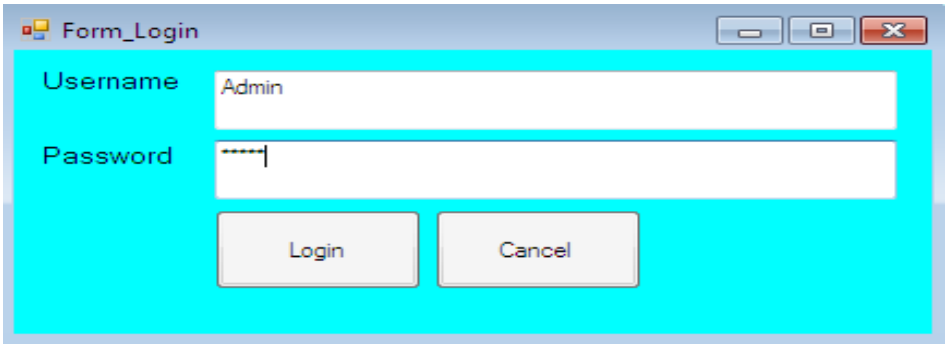
Alternatif	Total Nilai	Rangking
Raita	26.72698596	1
Apostolico Crochemore	21.96579774	2

Berdasarkan tabel 15, diketahui bahwa hasil dari analisis perbandingan algoritma *Raita* dan *Apostolico Crochemore* menunjukkan bahwa algoritma *Raita* lebih cepat dibandingkan *Apostolico Crochemore.*

3.4 Tampilan Sistem

Tampilan sistem merupakan *user interface* sistem yang menghubungkan *user* dengan sistem agar dapat melakukan proses perbandingan algoritma *Raita* dan *Apostolico Crochemore* dalam studi kasus pencarian data perpustakaan. Adapun tampilan dari sistem pencarian data perpustakaan yang dibangun dalam penelitian ini antara lain adalah sebagai berikut :

1. *Form Login*
Form Login merupakan tampilan sistem pencarian data perpustakaan yang berfungsiuntuk melakukan proses login agar dapat masuk ke halaman *form* menu utama. Adapun tampilan f*orm login* pada sistem pencarian data perpustakaan yang dibangun pada penelitian ini dapat dilihat pada gambar berikut:



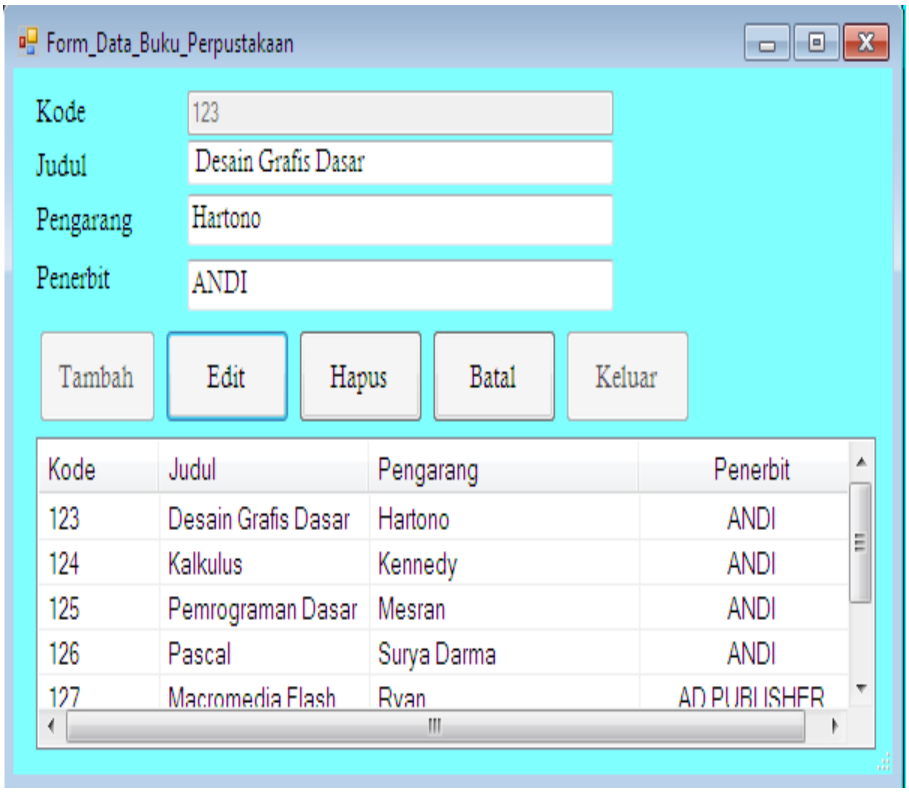
Gambar 2. *Form* Login

Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

2. *Form* Menu Utama
   Form menu utama merupakan tampilan sistem pencarian data perpustakaan yang berfungsi untuk menampilkan pilihan menu file (Data Perpustakaan), Perbandingan Algoritma, dan logout. Adapun *form* menu utama sistem pencarian data perpustakaan dapat dilihat pada gambar berikut:

Gambar 3. *Form* Menu Utama

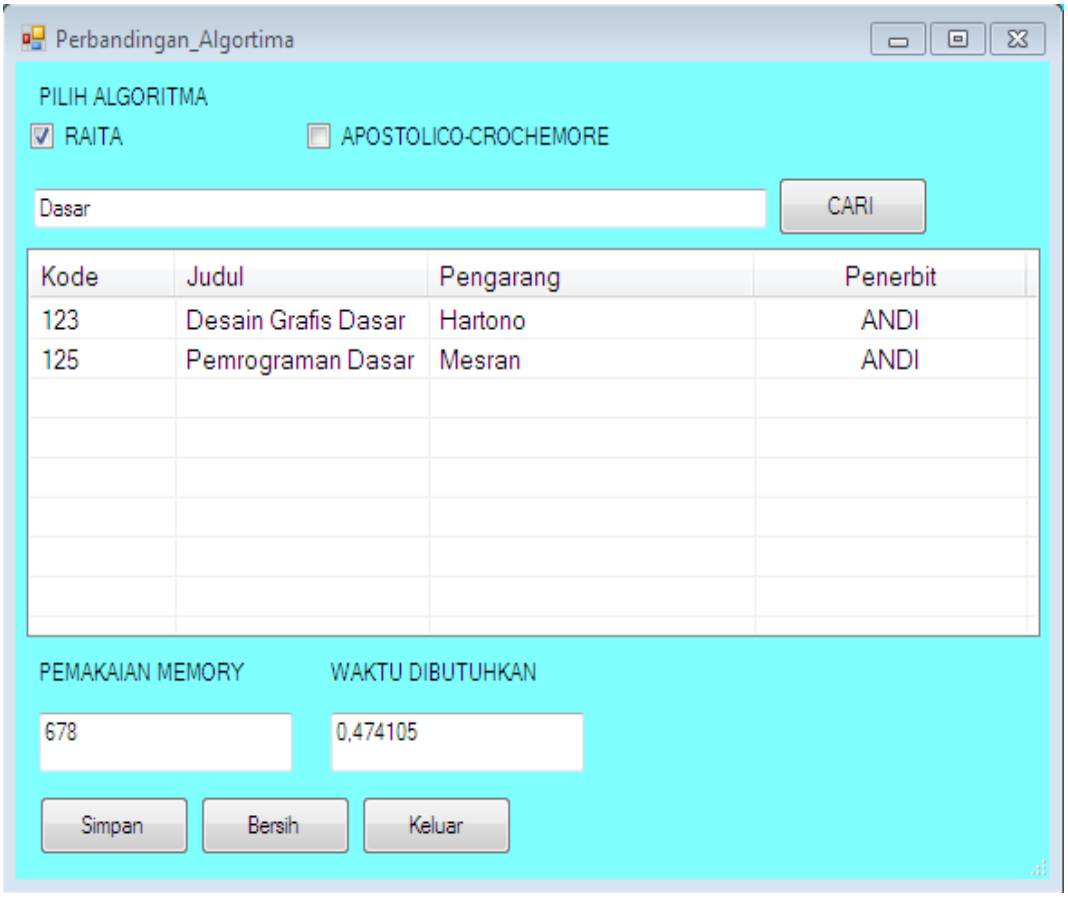
3. *Form* Data Perpustakaan
   Form gejala merupakan *form*sistem pencarian data perpustakaan yang berfungsi untuk melakukan pengolahan data perpustakaan. Adapun *form* perpustakaan pada sistem pencarian data perpustakaan dapat dilihat pada gambar berikut:



Gambar 4. *Form* Data Perpustakaan

4. *Form* Perbandingan Algoritma
   Form perbandingan algoritma merupakan *form*sistem pencarian data perpustakaan yang berfungsi melakukan pengolahan data perbandingan algoritma. Adapun *form* perbandingan algoritma pada sistem pencarian data perpustakaan dapat dilihat pada gambar berikut ini :
   a. Hasil Pencarian Algoritma *Raita*
      Adapun tampilan hasil sistem dengan implementasi pencarian bukut perpustakaan dengan menggunakan algoritma *Raita* adalah sebagai berikut:

Bulletin of Information System Research (BIOS)
Volume 1, No 2, April 2023 Page: 70-81
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios



Gambar 5. *Form* Hasil Pencarian Algoritma Raita

b. Hasil Pencarian Algoritma *Apostolico Crochemore*
Adapun tampilan hasil sistem dengan implementasi pencarian bukut perpustakaan dengan menggunakan algoritma *Apostolico Crochemore* adalah sebagai berikut:

Gambar 6. *Form* Hasil Pencarian Algoritma *Apostolico Crochemore*

3.5 Hasil Pengujian

Hasil pengujian dari penelitian ini adalah pemakaian memory dan waktu dibutuhkan dalam pencarian data buku perpustakaan pada implementasi Algoritma *Raita* dan Algoritma *Apostolico Crochemore*. Adapun hasil pengujian dari kedua Algoritma yang telah diterapkan adalah sebagai berikut.

1. Hasil Pengujian Algoritma *Raita*
   Adapun hasil pengujian algoritma Raita yang telah dilakukan menggunakan bahasa pemrograman visual basic net 2008 dapat dilihat pada Tabel 4.32 di bawah ini:

Tabel 16. Hasil Pengujian Algoritma *Raita*

No	Pattern	Pemakaian Memory (Byte)	Waktu Dibutuhkan (Detik)
1	Dasar	678	0,474105
2	D	682	0,677101
3	Da	684	0,794103
4	Das	687	0,814112
5	Dasr	0	0

2. Hasil Pengujian Algoritma *Apostolico Crochemore*
   Adapun hasil pengujian algoritma *Apostolico Crochemore* yang telah dilakukan menggunakan bahasa pemrograman visual basic net 2008 dapat dilihat pada Tabel 4.33 di bawah ini:

Tabel 17. Hasil Pengujian Algoritma *Apostolico Crochemore*

No	Pattern	Pemakaian Memory (Byte)	Waktu Dibutuhkan (Detik)
1	Dasar	456	0,374105
2	D	468	0,474101
3	Da	479	0,571100
4	Das	499	0,672112
5	Dasr	0	0

4. KESIMPULAN

Adapun kesimpulan yang penulis uraikan dari hasil penelitian ini adalah prosedur prosedur pencarian data perpustakaan dilakukan berdasarkan judul buku (*pattern)* sebagai kata kunci untuk mendapatkan hasil pencocokan *string*. Penerapan algoritma *Raita* dan *Apostolico Crochemore* dalam penelitian ini berhasil memberikan hasil pencocokan string untuk pencarian data perpustakaan dengan cepat dan tepat. Dari hasil analisa perbandingan Algoritma *Raita* dan *Apostolico Crochemore* untuk pencarian data perpustakaan diketahui bahwa Algoritma *Raita* lebih cepat dari *Apostolico Crochemore*.

REFERENCES

[1] S. Bill, P. Ginting, N. Marbun, M. Zarlis, and D. Hartama, "Penerapan Algoritma Horspool Perancangan Aplikasi Kamus Bahasa Bima – Indonesia Kamus adalah suatu sumber informasi atau referensi suatu Penerapan Algoritma Horspool Perancangan Aplikasi Kamus Bahasa Bima –," vol. 1, pp. 887–891, 2019.
[2] B. N. Sri Sugiarti, "Implementasi Algoritma Zhu Takaoka Pada Aplikasi Olshop Kamera Digital," *JURIKOM (Jurnal Ris. Komputer)*, 2020.
[3] D. R. Bawanto and N. Rosmawanti, "Perbandingan Algoritma Binary Search Dan Raita Dalam Pencarian Data," pp. 1311–1317.
[4] S. A. T, "perbandingan algoritma Raita dan algoritma Berry Ravindran dalam pencocokan string pada aplikasi kamus Indonesia-Korea," vol. 1, no. 2, pp. 6–38, 2019.
[5] A. M. S. Mamta Culkari Puding, Jumadil Nangi, "perbandingan algoritma Horspool dan algoritma Raita pada aplikasi istilah psikologi berbasis android," vol. 5, no. 13512025, pp. 1–13, 2019.
[6] H. K. Siregar, "Perbandingan Algoritma Knuth Morris Pratt dan Algoritma Apostolico Crochemore pada Aplikasi Kamus Bahasa Indonesia," 2017.
[7] S. Basuki, *Pengantar Ilmu Perpustakaan*. Jakarta: Universitas Terbuka, 2009.
[8] Kamara, *Visualisasi Beberapa Algoritma Pencocokan String Dengan Java*. Bandung: Institut Teknologi Bandung, 2008.
[9] A. Sonita and Khairunnisyh, "The 8 th University Research Colloquium 2018 Universitas Muhammadiyah Purwokerto IMPLEMENTASI ALGORITMA STING MATCHING PADA RANCANG BANGUN IMPLEMENTATION OF STRING MATCHING ALGORITHM ON THE DESIGN OF DRUG The 8 th University Research Colloquium 2018 Unive," no. 4, pp. 124–128, 2018, [Online]. Available: http://repository.urecol.org/index.php/proceeding/article/view/491.
[10] J. I. Sinaga, M. Mesran, and E. Buulo, "Aplikasi Mobile Pencarian Kata Pada Arti Ayat Al-Qur ' an Berbasis Android Menggunakan ...," *J. INFOTEK*, vol. II, no. Juni 2016, pp. 68–72, 2016.
[11] S. & Munir, *Pencocokan String Berdasarkan Kemiripan Ucapan (Phonetic String Matching) dalam Bahasa Inggris*. Yogyakarta: SNATI UII, 2005.
[12] N. Marbun, M. Zarlis, D. Hartama, and B. J. D. Sitompul, "Implementasi Algoritma Raita Pada Pencarian Katalog Alkes," pp. 520–523, 2019.
[13] C. and Lecroq, *Handbook of Exact String Matching Algorithms*, 238th ed. 2004.
[14] S. G. Ratri, "Penggunaan Algoritma Apostolico-Crochemore Pada Proses Pencarian String di Dalam Teks," 2007.
[15] A. Fau, Mesran, and G. L. Ginting, "Analisa Perbandingan Boyer Moore Dan Knuth Morris Pratt Dalam Pencarian Judul Buku Menerapkan Metode Perbandingan Eksponensial (Studi Kasus : Perpustakaan STMIK Budi Darma)," *J. Times (Technology Informatics Comput. Syst.*, vol. 6, no. 1, pp. 12–22, 2017.
[16] A. Nugroho, *Rekayasa Perangkat Lunak Berbasis Objek dengan Metode USDP*. Yogyakarta: Andi, 2010.
[17] R. A.S-M.Shalahuddin, *Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*. Bandung: Informatika, 2014.
[18] W. Komputer, *Membuat Aplikasi Client Server dengan Visual Basic 2008*. Yogyakarta: Andi, 2010.

[19] B. Nugroho, *Panduan proyek sistem penjualan retail mini market*. Yogyakarta: Andi, 2012.