

2017

Perbandingan Algoritma Horspool dan Algoritma Raita pada Aplikasi Kamus Bahasa Indonesia-Jerman Berbasis Web

Valba, Ryan Ridho

<http://repositori.usu.ac.id/handle/123456789/2227>

Downloaded from Repositori Institusi USU, Universitas Sumatera Utara

PERBANDINGAN ALGORITMA HORSPOOL DAN ALGORITMA RAITA

PADA APLIKASI KAMUS BAHASA INDONESIA – JERMAN

BERBASIS WEB

SKRIPSI

RYAN RIDHO VALBA

121401039



PROGRAM STUDI S1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2017

PERBANDINGAN ALGORITMA HORSPOOL DAN ALGORITMA RAITA PADA
APLIKASI KAMUS BAHASA INDONESIA-JERMAN
BERBASIS WEB

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer

RYAN RIDHO VALBA
121401039



PROGRAM STUDI S1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2017

PERSETUJUAN

Judul : PERBANDINGAN ALGORITMA HORSPOOL
DAN ALGORITMA RAITA PADA APLIKASI
KAMUS BAHASA INDONESIA-JERMAN
BERBASIS WEB

Kategori : SKRIPSI

Nama : RYAN RIDHO VALBA

Nomor Induk Mahasiswa : 121401039

Program Studi : SARJANA (S1) ILMU KOMPUTER

Departemen : ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI
INFORMASI UNIVERSITAS SUMATERA
UTARA

Diluluskan di
Medan, Oktober 2017

Komisi Pembimbing :

Pembimbing 2

Pembimbing 1

Herriyance, ST, M.Kom
NIP. 198010242010121002

Amalia, S.T., M.T. .
NIP. 97812212014042001

Diketahui/Disetujui oleh
Program Studi S1 Ilmu Komputer
Ketua,

Dr. Poltak Sihombing, M.Kom
NIP. 196203171991031001

PERNYATAAN

PERBANDINGAN ALGORITMA HORSPOOL DAN ALGORITMA RAITA PADA APLIKASI KAMUS BAHASA INDONESIA-JERMAN BERBASIS WEB

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, Oktober 2017

Ryan Ridho Valba

121401039

PENGHARGAAN

Alhamdulillahirrabbi'l'amin. Puji dan syukur penulis ucapkan atas kehadiran Allah SWT yang telah memberikan limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini sebagai syarat untuk memperoleh gelar Sarjana Komputer, Program Studi Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara. Shalawat beriring salam penulis juga persembahkan kepada Nabi Besar Muhammad SAW.

Dengan segala kerendahan hati, pada kesempatan ini penulis menyampaikan terima kasih kepada semua pihak yang telah membantu penyelesaian skripsi dengan judul Perbandingan Algoritma *Horspool* dan Algoritma *Raita* Pada Aplikasi Kamus Bahasa Indonesia – Jerman Berbasis Web. Penulis mengucapkan terima kasih kepada:

1. Bapak Prof. Dr. Runtung Sitepu, SH., M.Hum sebagai Rektor Universitas Sumatera Utara.
2. Bapak Prof. Dr. Opim Salim Sitompul, M.Si sebagai Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Bapak Dr. Poltak Sihombing, M.Kom sebagai Ketua Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Amalia, S.T., M.T. sebagai Dosen Pembimbing I yang telah memberikan arahan dan motivasi kepada penulis dalam pengerjaan skripsi ini.
5. Bapak Herriyance, ST, M.Kom sebagai Dosen Pembimbing II yang telah memberikan arahan dan motivasi kepada penulis dalam pengerjaan skripsi ini.
6. Ibu Dr. Elviawaty Muisa Zamzami, ST, MT, MM selaku Dosen Penguji I yang telah memberikan saran dan kritik kepada penulis dalam penyempurnaan skripsi ini.
7. Ibu Dr. Maya Silvi Lydia, Msc selaku Dosen Penguji II yang telah memberikan saran dan kritik kepada penulis dalam penyempurnaan skripsi ini.
8. Seluruh Dosen serta staf Pegawai di Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
9. Kedua Orangtua penulis tercinta Ibunda Surmayani Mahari Putri dan Ayahanda Reyes Syarifuddin Sihombing, serta Adik Zuhri Ahmad, Pangeran Watanida dan Rafli Defransyah yang telah memberikan do'a, dukungan,

perhatian, kesabaran serta kasih sayang yang tulus serta pengorbanan yang tidak ternilai harganya.

10. Sahabat penulis Sari Ulfah yang telah memberikan do'a, dukungan, perhatian, kesabaran serta kasih sayang yang tulus serta pengorbanan yang tidak ternilai harganya.
11. Para sahabat yang selama ini telah menjadi keluarga kedua penulis selama mengikuti perkuliahan, tempat berbagi suka dan duka dalam pengerjaan skripsi ini, kepada Miftahul Huda, Ade Mutiara Kartika Nasution S.kom, Nadhira Dwi Sabrina S.kom , Ardi Betesda Clintia Ginting
12. Teman-teman kuliah, khususnya Kom A 2012 serta Stambuk 2012 yang tidak dapat disebut satu-persatu, yang telah banyak membantu dalam pengerjaan skripsi ini.
13. Semua pihak yang terlibat langsung ataupun tidak langsung yang tidak dapat penulis ucapkan satu-persatu yang telah membantu penyelesaian skripsi ini.

Sekali lagi penulis mengucapkan terima kasih kepada semua pihak yang membantu dalam penyelesaian skripsi ini yang tidak dapat disebutkan satu persatu, terima kasih atas ide, saran dan motivasi yang diberikan. Semoga Allah SWT memberikan limpahan karunia kepada semua pihak yang telah memberikan bantuan, perhatian, kasih sayang serta dukungan kepada penulis dalam menyelesaikan skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan karena kesempurnaan hanyalah milik Allah SWT semata. Oleh karena itu penulis menerima kritik dan saran dari semua pihak yang bersifat membangun dan menyempurnakan skripsi ini. Penulis berharap semoga skripsi ini bermanfaat bagi penulis sendiri pada khususnya dan pembaca pada umumnya.

Medan, Oktober 2017
Penulis,

Ryan Ridho Valba

ABSTRAK

Kamus adalah buku yang memuat kata-kata beserta keterangan arti. Kamus merupakan alat bantu yang ditujukan untuk membantu kita dalam memahami makna sebuah kata. Kamus Indonesia - Jerman adalah sebuah kamus istilah yang berisi istilah kata Indonesia – Jerman. Kamus identik dengan buku yang tebal yang lama dalam pencarian dan kurang efisien dalam penggunaannya. Untuk itu dibuatlah sebuah aplikasi kamus Indonesia – Jerman yang menerapkan Algoritma *Horspool - Raita* sebagai masalah pencariannya dan database *SQLite* sebagai penyimpanan data. Algoritma pencarian diperlukan untuk mempersingkat waktu dalam pencarian itu sendiri. Pencocokan *string* pada Algoritma *Horspool* dilakukan melalui pergeseran dari kanan kemudian ke kiri karakter. Sedangkan pencocokan *string* pada Algoritma *Raita* yang dilakukan melalui pergeseran dari kanan karakter kemudian ke kiri karakter dan ke tengah karakter. *Field* yang digunakan dalam pencarian ini adalah istilah bahasa Indonesia - Jerman secara umum, dengan memasukkan kata Indonesia – Jerman sebagai inputannya dalam pencarian, kemudian akan didapatkan hasil pencarian berupa arti dari istilah tersebut. Hasil dari aplikasi ini menampilkan keseluruhan pattern yang cocok dengan teks.

Kata Kunci: Kamus Indonesia - Jerman, Algoritma *Horspool*, Algoritma *Raita*, *String*

**COMPARISON OF HORSPOOL ALGORITHM AND RAITA ALGORITHM
ON APPLICATION OF INDONESIAN – JERMAN DICTIONARY WEB
BASED**

ABSTRACT

Dictionary is a book that contains words along with description of meanings. Dictionary is a auxiliary tools for helping to understand the meaning of the words. Dictionary Indonesian-Jerman is a Dictionary that contains Indonesia- Jerman terms. Dictionnary identical with the thick book that slow in its search and not efficient in its us. To overcome this, is made a Indonesian-Jerman Dictionary that applying *Horspool- Raita* as its search and SQLite database as a data storage. Searching algorithm is needed to shorten the time for searching. String matching on Horspool Algorithm moves from right to left of character. While String matching on Algorithm Raita moves from right to left and middle of character. The field that is used in this searching is a terms of Indonesian- Jerman in general, by inserting it as an input in search, then would got the result in the form of the meaning from the terms. The result of this application shows the whole of patterns that are match with text.

.

Keywords: Dictionary Indonesia - Jerman, *Horspool* algorithm, *Raita* algorithm string.

DAFTAR ISI

	Halaman
Persetujuan	ii
Pernyataan	iii
Penghargaan	iv
Abstrak	vi
Abstract	vii
Daftar Isi	viii
Daftar Tabel	x
Daftar Gambar	xi
Daftar Lampiran	xii
Bab 1 Pendahuluan	
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	3
1.3 Batasan Penelitian	3
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	3
1.6 Metodologi Penelitian	4
1.7 Sistematika Penulisan	4
Bab 2 Landasan Teori	
2.1 Kamus	6
2.2 Aplikasi Berbasis Web	6
2.3 Pengertian Pencocokan <i>String (String Matching)</i>	6
2.3.1 <i>Cara Kerja String Matching</i>	7
2.4 Algoritma	8
2.4.1 <i>Kompleksitas Algoritma</i>	8
2.4.2 <i>Algoritma Horspool</i>	8
2.4.3 <i>Algoritma Raita</i>	13
2.5 Penelitian Relevan	16
Bab 3 Analisis dan Perancangan Sistem	
3.1 Analisis Sistem	18
3.1.1 <i>Analisis Masalah</i>	18
3.1.2 <i>Analisis Persyaratan</i>	19
3.1.2.1 <i>Analisis persyaratan Fungsional</i>	19
3.1.2.2 <i>Analisis persyaratan Non-Fungsional</i>	20
3.1.3 Pemodelan Sistem	20
3.1.3.1 <i>Use Case Diagram</i>	21
3.1.3.2 <i>Activity Diagram</i>	22
3.1.3.3 <i>Sequence Diagram</i>	23

3.1.3.4 <i>Arsitektur Umum</i>	24
3.1.3.5 <i>Flowchart</i>	25
3.2 Perancangan <i>Interface</i>	30
3.2.1 <i>Form Beranda</i>	30
3.2.2 <i>Form Input Data</i>	31
3.2.3 <i>Form Tentang</i>	33
 Bab 4 Implementasi dan Pengujian	
4.1 Implementasi	34
4.1.1 <i>Halaman Home</i>	34
4.1.2 <i>Halaman Admin</i>	35
4.1.3 <i>Halaman tentang</i>	36
4.2 Implementasi Sistem	36
4.3 Hasil Pengujian Sistem	44
4.3.1 <i>Kompleksitas Algoritma Big θ</i>	44
4.3.2 <i>Running Time</i>	48
 Bab 5 Kesimpulan dan Saran	
5.1 Kesimpulan	53
5.2 Saran	54
 Daftar Pusaka	55
Listing Program	A-1
Curriculum Vitae	B-1

DAFTAR TABEL

Nomor Tabel	Nama Tabel	Halaman
2.1	<i>BmBc</i> pada praproses	10
2.2	Inisialisasi awal <i>BmBc</i>	11
2.3	Pembuatan <i>BmBc</i>	11
2.4	Iterasi algoritma <i>Horspool</i> pertama	12
2.5	Iterasi algoritma <i>Horspool</i> kedua	12
2.6	Iterasi algoritma <i>Horspool</i> ketiga	13
2.7	Iterasi algoritma <i>Horspool</i> keempat	14
2.8	Pencarian pada teks proses pertama	15
2.9	Pencarian pada teks proses kedua	15
2.10	Pencarian pada teks proses ketiga	16
4.1	Tabel <i>Bmbc</i>	38
4.2	Pencarian kata pada teks pertama	38
4.3	Pencarian kata pada teks kedua	39
4.4	Pencarian kata pada teks pertama	39
4.5	Pencarian kata pada teks kedua	40
4.6	Tabel <i>BmBc</i>	41
4.7	Pencarian kata pada teks pertama	41
4.8	Pencarian kata pada teks kedua	42
4.9	Pencarian kata pada teks ketiga	42
4.10	Pencarian kata pada teks pertama	43
4.11	Pencarian kata pada teks kedua	43
4.12	Pencarian kata pada teks ketiga	43
4.13	Tabel kompleksitas <i>Horspool</i>	44
4.14	Tabel kompleksitas <i>Raita</i>	46
4.15	Hasil pengujian <i>Horspool</i>	49
4.16	Hasil pengujian <i>Raita</i>	50

DAFTAR GAMBAR

Nomor Gambar	Nama Gambar	Halaman
3.1	Diagram <i>Ishikawa</i>	19
3.2	<i>Use Case Diagram</i> Sistem	21
3.3	<i>Activity Diagram</i> Proses Deskripsi	22
3.4	<i>Sequence Diagram</i> Sistem	23
3.5	Arsitektur Umum Sistem Kamus Indonesia-Jerman	24
3.6	<i>Flowchart</i> Tampilan Aplikasi Kamus	25
3.7	<i>Flowchart</i> Proses <i>BmBc</i>	26
3.8	<i>Flowchart</i> Algoritma <i>Horspool</i>	27
3.9	<i>Flowchart</i> Tabel <i>BmBc</i>	28
3.10	<i>Flowchart</i> Algoritma <i>Raita</i>	29
3.11	<i>Form</i> Beranda Pada Sistem	30
3.12	<i>Form</i> Admin	32
3.13	<i>Form</i> Tentang	33
4.1	Implementasi Halaman <i>Home</i>	35
4.2	Implementasi Halaman Admin	36
4.3	Implementasi Halaman Tentang	36
4.4	Pengujian Pencarian Kata “Aku” Indonesia-Jerman	37
4.5	Pengujian Pencarian Kata “Ich” Jerman- Indonesia	40
4.6	Perbandingan Hasil <i>Running Time</i> Algoritma <i>Horspool</i> dan Algoritma <i>Raita</i>	52

DAFTAR LAMPIRAN

	Hal
1. Listing Program	A1
2. Daftar Riwayat Hidup	B1

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Bahasa Jerman merupakan salah satu bahasa yang banyak dipelajari di dunia. Saat ini bahasa Jerman menjadi salah satu bahasa penting dalam komunikasi internasional. Bahasa ini memiliki peranan penting di bidang ekonomi global maupun ilmu pengetahuan. Di Indonesia bahasa Jerman menjadi salah satu mata pelajaran pilihan dimulai setingkat SMA. Mata pelajaran bahasa Jerman pada dasarnya memiliki peran yang cukup penting bagi perkembangan anak didik di Indonesia sejalan dengan pesatnya perkembangan jaman pada era teknologi informasi saat ini (Santoso, *et al* 2011).

Dengan kemajuan teknologi internet yang semakin maju manusia dapat mempelajari dan menemukan istilah kosakata bahasa Jerman dengan mudah dalam bentuk digital dan bisa diakses bebas tanpa batas. Pesatnya teknologi internet yang berkembang membuat aplikasi-aplikasi berbasis web bermunculan. Aplikasi berbasis web seperti kamus *online* yang bisa digunakan tanpa harus melakukan penginstalan seperti aplikasi kamus berbasis *android* dan *desktop*. Dengan berkembangnya aplikasi berbasis web, maka penulis tertarik membuat kamus online bahasa Jerman dengan aplikasi berbasis web untuk mempermudah manusia dalam mempelajari bahasa Jerman dengan mudah dan praktis. Pembuatan kamus bahasa dalam bentuk teknologi digital dapat diimplementasikan dengan menggunakan metode pencocokan string. Pencocokan string banyak digunakan dalam aplikasi pengolahan teks untuk pencarian kata dalam berkas teks. Pencarian string di dalam teks disebut juga string matching

Algoritma *Horspool* merupakan turunan dari algoritma *Boyer-Moore* dan mudah dalam implementasinya. Ketika panjang dari *pattern* kecil, sangat tidak efisien untuk menggunakan algoritma *Boyer-Moore*. Algoritma *Horspool* hanya

menggunakan perpindahan *bad-character* yang terjadi pada *Boyer-Moore*. Untuk melakukan dan menghitung nilai pergeseran *bad-character* adalah dengan melihat karakter paling kanan pada *window*. Nilai pergeseran ini dihitung pada tahap praproses untuk semua karakter pada set alfabet sebelumnya. Algoritma ini lebih efisien digunakan ketika ditemukan panjang *pattern* yang kecil (Sheik, *et al.* 2004). Adapun kelebihan Algoritma *Horspool* telah digunakan dalam beberapa penelitian. Penelitian oleh Tambun, E.D. (2010) yang menbandingkan penggunaan Algoritma *Boyer-Moore* dan Algoritma *Horspool* pada pencarian *string* dalam bahasa medis telah dihasilkan dari pengujian dan analisa program yang dibuat bahwa pencarian *string* dalam bahasa medis dengan menggunakan Algoritma *Boyer-Moore* lebih baik dibandingkan dengan Algoritma *Horspool* dikarenakan bahasa medis banyak menggunakan kosakata yang panjang dengan karakter yang beragam. Namun Algoritma *Boyer-Moore* kurang cocok digunakan untuk teks dengan varian karakter yang sedikit dikarenakan lompatan pengecekan yang dilakukan oleh *Boyer-Moore* menjadi cenderung lebih pendek.

Algoritma *Raita* adalah algoritma yang merancang perbandingan karakter Pada tahap awal di mulai dengan membandingkan karakter paling kanan pada pola di *window* sampai kecocokan terjadi. Selanjutnya membandingkan karakter paling kiri pola di *window* kemudian karakter yang tersisa dibandingkan dari kanan hingga ke kiri sampai pencocokkan selesai atau ketidakcocokan terjadi (Alkhamaiseh & Alshagarin 2014). Adapun kelebihan Algoritma *Raita* telah digunakan dalam beberapa penelitian oleh S Zariani mutia (2017) yang membandingkan Algoritma *Smith* dan Algoritma *Raita* pada fase *preprocessing* dan fase pencarian dari kedua algoritma menghasilkan bahwa Algoritma *Raita* lebih efektif dan memiliki *running time* yang lebih cepat untuk setiap *pattern* dari pada Algoritma *Smith*.

Alasan penulis membandingkan kedua algoritma ini untuk mengetahui algoritma string matching yang lebih baik antara algoritma *Horspool* dan algoritma *Raita*. Parameter yang di uji adalah running time dan kompleksitas algoritma (Big O) dalam pencarian *string* di dalam kamus bahasa Indonesia – Jerman.

1.2 Rumusan Masalah

Berdasarkan uraian dari latar belakang di atas, rumusan masalah yang akan dibahas dalam penelitian ini adalah bagaimana melakukan pencarian kata dengan algoritma *Horspool* dan algoritma *Raita* serta membandingkan hasilnya untuk pencarian *string* pada teks.

1.3 Batasan Penelitian

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Aplikasi menyimpan 1000 kata bahasa Indonesia dan 1000 kata bahasa Jerman
2. Bahasa Pemrograman yang digunakan adalah Hypertext Preprocessor (PHP).
3. Aplikasi ini menggunakan DBMS MySQL untuk penyimpanan kosakata bahasa Indonesia dan bahasa Jerman
4. Aplikasi ini dirancang untuk menerjemahkan kata bahasa Indonesia ke bahasa Jerman dan berlaku sebaliknya
5. Parameter pembandingnya adalah waktu (ms) dan kompleksitas algoritma ($\text{Big } \theta$).
6. Pengujian ini dilakukan dengan perangkat keras menggunakan *Processor Intel(R)Core(TM) i3-2330M CPU @ 2.2GHz. RAM 2GB. Sistem operasi Windows 7 Ultimate Type 64-bit*

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mengetahui algoritma mana yang lebih baik diantara Algoritma *Horspool* dan Algoritma *Raita* untuk proses pencarian kata dengan menggunakan parameter pembanding, yaitu *running time* dan kompleksitas algoritma ($\text{Big } \theta$).

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut :

1. Mengetahui kecepatan waktu pencarian kata di dalam kamus bahasa Jerman - Indonesia ini dengan menggunakan algoritma *Hoorspool* dan algoritma *Raita* dan membandingkan hasilnya.
2. Mengetahui algoritma mana yang lebih baik diantara algoritma *Horspool* dan algoritma *Raita* pada aplikasi kamus bahasa Jerman – Indonesia
3. Membantu masyarakat yang ingin mempelajari bahasa Jerman dengan menggunakan kamus bahasa Jerman berbasis web ini.

1.6 Metodologi Penelitian

Penelitian ini menerapkan beberapa metode penelitian sebagai berikut:

1. Studi Literatur

Pada tahap ini dilakukan pengumpulan referensi yang diperlukan dalam penelitian. Hal ini dilakukan untuk memperoleh informasi dan data yang diperlukan untuk penulisan skripsi ini. Referensi yang digunakan dapat berupa buku, jurnal, artikel, situs internet yang berkaitan dengan penelitian ini.

2. Pengumpulan dan Analisis Data

Pada tahap ini dilakukan pengumpulan dan analisa data yang berhubungan dengan penelitian ini seperti fungsi algoritma *Horspool* dan *raita*.

3. Perancangan Sistem

Pada tahap ini dilakukan perancangan *user interface*, *Unified Modeling Language (UML)*, dan struktur program sistem pengamanan teks.

4. Implementasi Sistem

Sistem diimplementasikan dengan menggunakan Algoritma *Horspool* dan *raita*.

5. Pengujian Sistem

Pada tahap ini akan dilakukan pengujian terhadap sistem yang telah dikembangkan.

6. Dokumentasi Sistem

Melakukan pembuatan dokumentasi sistem mulai dari tahap awal hingga pengujian sistem, untuk selanjutnya dibuat dalam bentuk laporan penelitian (skripsi).

1.7 Sistematika Penulisan

Setelah uraian di bab satu ini, penyajian selanjutnya disampaikan dengan sistematika berikut :

BAB 1: PENDAHULUAN

Bab ini berisi mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan skripsi.

BAB 2: LANDASAN TEORI

Bab ini berisi mengenai teori-teori yang berkaitan *string matching*, dan penerapan menggunakan algoritma *Horspool* dan *raita*.

BAB 3: ANALISIS DAN PERANCANGAN SISTEM

Bab ini terdiri dari tahap analisis sitem dan desain perancangan dari program yang akan dibuat, dalam hal ini termasuk juga algoritma dan program tersebut.

BAB 4: IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi ulasan dan pengujian terhadap program yang telah diimplementasikan dengan menggunakan bahasa PHP dan Mysql.

BAB 5: KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil penelitian yang telah selesai dilakukan, juga saran atau rekomendasi guna penelitian lebih lanjut.

BAB 2

TINJAUAN PUSTAKA

2.1. Kamus

Menurut Kamus Besar Bahasa Indonesia, pengertian kamus adalah buku acuan yang memuat kata dan ungkapan yang biasanya disusun menurut abjad berikut keterangan maknanya, pemakaiannya dan terjemahannya.

2.2. Aplikasi Berbasis Web

Aplikasi adalah sebuah sistem komputer yang tidak mempunyai wujud secara fisik di dunia nyata, tetapi aplikasi disimpan di dalam sistem komputer dalam bentuk data digital. Aplikasi bisa menjalankan setiap perintah yang dimasukkan oleh si pengguna komputer, hasil dari proses aplikasi tersebut bisa berbentuk data digital ataupun data fisik.

Aplikasi berbasis web adalah aplikasi yang dijalankan melalui browser. Aplikasi seperti ini pertama kali dibangun hanya dengan menggunakan bahasa yang disebut dengan HTML (HyperText Markup Language) dan protokol yang digunakan dinamakan HTTP (HyperText Transfer Protokol).

2.3. Pengertian Pencocokan *String* (*String Matching*)

String matching adalah pencarian sebuah *pattern* pada sebuah teks (Cormen, *et al.* 1994). *String matching* digunakan untuk menemukan suatu string yang disebut dengan *pattern* dalam *string* yang disebut dengan teks (Charras & Lecroq 1997). Pencocokan *string* merupakan bagian penting dari sebuah dokumen. Hasil dari pencarian sebuah *string* tergantung dari teknik atau cara pencocokan *string* yang digunakan (Syahroni &

Munir 2005) Prinsip kerja algoritma *string matching* (Effendi, *et al.* 2013) adalah sebagai berikut:

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 pergeseran ke kanan pada *window*. Prosedur ini dilakukan berulang-ulang sampai *window* berada pada akhir teks. Mekanisme ini disebut mekanisme *sliding window*.

Algoritma string matching mempunyai tiga komponen utama (Effendi, *et al.* 2013), yaitu:

1. *Pattern*, yaitu deretan karakter yang akan dicocokkan dengan teks, dinyatakan dengan $x[0 \dots m - 1]$, panjang *pattern* dinyatakan dengan m .
2. Teks, yaitu tempat pencocokan *pattern* dilakukan. Dinyatakan dengan $y[0 \dots n - 1]$, panjang teks dinyatakan dengan n .
3. Alfabet, berisi semua simbol yang digunakan oleh bahasa pada teks dan *pattern*, dinyatakan dengan Σ dengan ukuran dinyatakan ASIZE.

2.3.1. Cara kerja String Matching

Cara yang jelas untuk mencari *pattern* yang cocok dengan teks adalah dengan mencoba mencari di setiap posisi awal dari teks dan mengabaikan pencarian secepat mungkin jika karakter yang salah ditemukan (Knuth, *et al.* 1977). Proses pertama adalah menyelaraskan bagian paling kiri dari *pattern* dengan teks. Kemudian dibandingkan karakter yang sesuai dari teks dan *pattern*. Setelah seluruhnya cocok maupun tidak cocok dari *pattern*, *window* digeser ke kanan sampai posisi $(n - m + 1)$ pada teks. Menurut Singh & Verma (2011), efisiensi dari algoritma terletak pada dua tahap:

1. Tahap praproses, tahap ini mengumpulkan informasi penuh tentang *pattern* dan menggunakan informasi ini pada tahap pencarian.
2. Tahap pencarian, *pattern* dibandingkan dengan *window* dari kanan ke kiri atau kiri ke kanan sampai kecocokan atau ketidakcocokan terjadi.

2.4. Algoritma

Algoritma merupakan suatu prosedur untuk menyelesaikan suatu masalah yang tersusun secara logis dan sistematis serta akan memperoleh data masukan menjadi keluaran yang diinginkan berupa informasi (Rahayuningsih 2016).

2.4.1. Kompleksitas Algoritma

Algoritma yang efisien adalah algoritma yang meminimalkan kebutuhan waktu dan ruang. Digunakan untuk menjelaskan model pengukuran waktu dan ruang ini adalah kompleksitas algoritma. Namun, kebutuhan waktu dan ruang dari suatu algoritma bergantung pada jumlah data yang diproseskan dan algoritma yang digunakan. Kompleksitas Waktu, $T(n)$, adalah jumlah operasi yang dilakukan untuk melaksanakan algoritma sebagai fungsi dari ukuran masukan n . Maka, dalam mengukur kompleksitas waktu dihitunglah banyaknya operasi yang dilakukan oleh algoritma. Kompleksitas waktu untuk algoritma-algoritma yang dibahas akan dinyatakan dengan notasi O besar (Big-O notation). Definisi dari notasi O besar adalah, jika sebuah algoritma mempunyai waktu asimptotik $O(f(n))$, maka jika n dibuat semakin besar, waktu yang dibutuhkan tidak akan pernah melebihi suatu konstanta C dikali dengan $f(n)$. Jadi $f(n)$ adalah batas atas (upper bound) dari $T(n)$ untuk n yang besar (Rahayuningsih 2016).

2.4.2. Algoritma Horspool

Algoritma Horspool adalah penyederhanaan dari algoritma Boyer-Moore yang dibuat oleh R. Nigel Horspool. Menurut Horspool, R.N.(1980), algoritma Horspool bekerja dengan metode yang hampir sama dengan algoritma Boyer-Moore namun tidak melakukan lompatan berdasarkan karakter pada pattern yang ditemukan tidak cocok pada teks. R. Nigel Horspool melakukan penelitian tentang algoritma Boyer-Moore. Dari hasil penelitiannya beliau mengemukakan gagasan tambahan untuk algoritma Boyer-Moore. Algoritma ini dikenal dengan algoritma Horspool atau algoritma Boyer-Moore-Horspool (Tambun 2010).

Algoritma Horspool mempunyai nilai pergeseran karakter yang paling kanan dari *window*. Pada tahap observasi awal (*preprocessing*), nilai *shift* akan dihitung untuk semua karakter. Pada tahap ini, dibandingkan pattern dari kanan ke kiri hingga kecocokan atau ketidakcocokan pattern terjadi. Karakter yang paling kanan pada *window* digunakan sebagai indeks dalam melakukan nilai *shift*. Dalam kasus ketidakcocokan (karakter tidak terdapat pada pattern) terjadi, *window* digeser oleh panjang dari sebuah *pattern*. Jika tidak, *window* digeser menurut karakter yang paling kanan pada *pattern* (Baeza-Yates & Regnier 1992). Terdapat dua tahap pada pencocokan *string* menggunakan algoritma *Horspool* (Singh & Verma 2011), yaitu :

1. Tahap praproses

Pada tahap ini, dilakukan observasi *pattern* terhadap teks untuk membangun sebuah tabel *bad-match* yang berisi nilai *shift* ketika ketidakcocokan antara *pattern* dan teks terjadi. Secara sistematis, langkah-langkah yang dilakukan algoritma *Horspool* pada tahap praproses adalah :

- a. Algoritma *Horspool* melakukan pencocokan karakter ter-kanan *pattern*.
- b. Setiap karakter pada *pattern* ditambah ke dalam tabel *BmBc* dan dihitung nilai *shift*-nya.
- c. Karakter yang berada pada ujung *pattern* tidak dihitung dan tidak dijadikan karakter ter-kanan dari karakter yang sama dengannya.
- d. Apabila terdapat dua karakter yang sama dan salah satunya bukan karakter ter-kanan, maka karakter dengan indeks terbesar yang dihitung nilai *shift*-nya.
- e. Algoritma *Horspool* menyimpan panjang dari *pattern* sebagai panjang nilai *shift* secara *default* apabila karakter pada teks tidak ditemukan dalam *pattern*.
- f. Nilai (BM) *shift* yang akan digunakan dapat dicari dengan perhitungan panjang, dari *pattern* dikurang indeks terakhir karakter dikurang 1, untuk masing-masing karakter (BC) , $BM = m - i - 1$.

Sebagai contoh, dapat dilihat pada tabel 2.1 berikut:

Pattern : VALBA

$$\begin{array}{ccccccccc} & V & A & L & B & A & & & \\ \underbrace{} & \underbrace{} & \underbrace{} & \underbrace{} & \underbrace{} & & & & \\ 0 & 1 & 2 & 3 & 4 & & & & \end{array}$$

Tabel 2.1 BmBc pada praproses

BC	BM
V	4
A	3
L	2
B	1
*	5

Untuk mencari nilai geser pada tabel BmBc, digunakan rumus $BM[i] = m - i - 1$.

$$BM[0] = 5 - 0 - 1 = 4$$

$$BM[1] = 5 - 1 - 1 = 3$$

$$BM[2] = 5 - 2 - 1 = 2$$

$$BM[3] = 5 - 3 - 1 = 1$$

(*) : karakter yang tidak dikenali

2. Tahap Pencarian

Secara sistematis, langkah-langkah yang dilakukan algoritma *Horspool* pada tahap pencarian adalah:

- Dilakukan perbandingan karakter paling kanan pattern terhadap *window*.
- Tabel BmBc digunakan untuk melewati karakter ketika ketidakcocokan terjadi.
- Ketika ada ketidakcocokan, maka karakter paling kanan pada *window* berfungsi sebagai landasan untuk menentukan jarak *shift* yang akan dilakukan.
- Setelah melakukan pencocokan (baik hasilnya cocok atau tidak cocok) dilakukan pergeseran ke kanan pada *window*.
- Prosedur ini dilakukan berulang-ulang sampai *window* berada pada akhir teks atau ketika pattern cocok dengan teks.

Sebagai contoh dalam mendeskripsikan algoritma *Hoorspol* diberikan teks dan *pattern* sebagai berikut

Teks = RYAN RIDHO VALBA

Pattern = VALBA

Inisialisasi awal dan pembuatan BmBc terlihat pada Tabel 2.2 dan 2.3 berikut.

Tabel 2.2 Inisialisasi awal *BmBc*

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	R	Y	A	N	R	I	D	H	O		V	A	L	B	A
P	V	A	L	B	A										
I	0	1	2	3	4										

Tabel 2.3 Pembuatan *BmBc*

P	V	A	L	B	*
I	0	1	2	3	-
V	4	3	2	1	5

Dilihat dari Tabel 2.2 di atas, inisialisasi awal *BmBc* dilakukan. Setiap teks dan *pattern* masing masing diberi nilai m dan i , dimana m adalah panjang *pattern* dan i adalah indeks. Tabel 2.3 menunjukkan nilai pergeseran *BmBc* dengan menghitung nilai BM seperti yang telah dilakukan pada Tabel 2.1. Pada tahap awal pencarian, dilakukan perbandingan karakter paling kanan *pattern* terhadap *window*. Apabila terjadi ketidakcocokan maka akan dilakukan pergeseran ke kanan untuk melewati karakter yang tidak cocok di mana nilai pergeserannya terdapat pada tabel *BmBc*. Karakter paling kanan teks pada *window* berfungsi sebagai landasan untuk menentukan jarak geser yang akan dilakukan. Hal ini terlihat pada Tabel 2.4 berikut.

Tabel 2.4 Iterasi algoritma *Horspool* pertama

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	R	Y	A	N	R	I	D	H	O		V	A	L	B	A
P	V	A	L	B	A										

I	0	1	2	3	4										
---	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--

Pada Tabel 2.4 terdapat ketidakcocokan antara karakter “R” dan “A”. Karakter “R” tidak terdapat pada tabel *BmBc* sehingga digantikan oleh tanda (*). Tanda (*) bernilai sebesar 5 sehingga dilakukan pergeseran sebanyak 5 kali. Hal ini terlihat pada Tabel 2.5.

Tabel 2.5 Iterasi algoritma *Horspool* kedua

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	R	Y	A	N	R	I	D	H	O		V	A	L	B	A
P						V	A	L	B	A					
I						0	1	2	3	4					

Pada Tabel 2.5 terdapat ketidakcocokan kembali antara karakter “(spasi)” dan “A”. Karakter “(spasi)” tidak terdapat pada tabel *BmBc* sehingga digantikan oleh tanda (*). Tanda (*) bernilai sebesar 5 sehingga dilakukan pergeseran sebanyak 5 kali. Hal ini terlihat pada Tabel 2.6.

Tabel 2.6 Iterasi algoritma *Horspool* ketiga

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	R	Y	A	N	R	I	D	H	O		V	A	L	B	A
P											V	A	L	B	A
I											0	1	2	3	4

Pada Tabel 2.6 *window* telah berada pada akhir teks dan semua *pattern* cocok dengan teks. Seluruh pencocokan karakter menggunakan algoritma *Horspool* telah selesai dan berhenti pada iterasi ketiga.

2.4.3. Algoritma Raita

Algoritma *Raita* merupakan bagian dari algoritma *exact string matching* yaitu pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama. *Raita* merancang sebuah algoritma dengan membandingkan karakter yang terakhir dari pola dari karakter paling kanan dari *window*. Jika mereka cocok, kemudian karakter pertama dari pola teks paling kiri dari *window* juga dibandingkan. Jika mereka cocok, 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 karakter kedua ke karakter kedua terakhir, dan mungkin membandingkan dengan karakter tengah lagi (Charras & Lecroq 1997).

Tahap pencarian algoritma *Raita* :

- a. Buat tabel pergeseran pola yang dicari sebagai kata yang akan dicari pada teks.
- b. Jika dalam proses pembandingan terjadi ketidakcocokan antara pasangan karakter pada akhir pola dengan karakter teks, pergeseran dilakukan sesuai nilai karakter pada tabel *BmBc*.
- c. Jika dalam proses pembandingan akhir pola terjadi ketidakcocokan lagi maka karakter akan digeser lagi sesuai tabel *BmBc*.
- d. Jika karakter akhir pola dengan karakter pada teks yang sedang dibandingkan cocok, maka posisi karakter pada pola dan teks akan memiliki nilai (0), dan dilanjutkan pencocokan pada karakter awal pola. Jika cocok maka dilanjutkan pencocokan dengan karakter tengah pola.
- e. 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.

Sebagai contoh perhitungan algoritma *Raita* akan diberikan teks dan *pattern* berikut

Teks = RYAN RIDHO VALBA

Pattern = VALBA

Untuk melakukan perhitungan maka dibuat tabel *BmBc* dengan persamaan sebagai berikut :

$$m-2 \dots\dots\dots(1)$$

Berfungsi sebagai batas pencarian karakter pada pola.

$m-i-1$ (2)

Berfungsi sebagai pencari nilai karakter pada tabel BmBc

VALBA
0 1 2 3 4

Tabel 2.7 Tabel *BmBc*

	BC	BM
0	V	4
1	A	3
2	L	2
3	B	1
4	*	5

Berdasarkan Tabel 2.7 dapat diketahui perhitungan tabel BmBC dengan persamaan :

$$\begin{aligned} & m - 2 \\ &= 5 - 2 \\ &= 3 \end{aligned}$$

Maka $i = 0 - 3$

Selanjutnya untuk mencari nilai geser pada tabel *BmBc* digunakan rumus $m - i - 1$.

$$BM[BC[V]] = 5 - 0 - 1 = 4$$

$$BM[BC[A]] = 5 - 1 - 1 = 3$$

$$BM[BC[L]] = 5 - 2 - 1 = 2$$

$$BM[BC[B]] = 5 - 3 - 1 = 1$$

(*) karakter yang tidak dikenali.

Panjang pola pada contoh di atas adalah sebesar 5. Maka untuk karakter yang tidak ada pada tabel diinisialisasikan dengan tanda (*) yang nilainya sesuai dengan panjang pola (m).

Pencarian algoritma *Raita* tahap pertama yaitu mencocokkan akhir pola dengan teks, jika terjadi ketidakcocokan maka pola akan bergeser ke kanan sebanyak nilai teks yang ada di tabel 2.7.

Tabel 2.8 Pencarian pada teks proses pertama

T	R	Y	A	N	R	I	D	H	O		V	A	L	B	A
P	V	A	L	B	A										

Dilihat pada Tabel 2.8 terjadi ketidakcocokan antara karakter “R” dan “A”. Karakter “R” tidak terdapat dalam tabel *BmBc* sehingga digantikan dengan tanda (*). Tanda (*) memiliki nilai sebesar 5 sehingga dilakukan pergeseran sebanyak 5 kali.

Pencarian algoritma *Raita* tahap kedua yaitu melakukan pergeseran sebanyak 5 kali (sesuai dengan karakter “R”).

Tabel 2.9 Pencarian pada teks proses kedua

T	R	Y	A	N	R	I	D	H	O		V	A	L	B	A
P						V	A	L	B	A					

Dilihat pada Tabel 2.9 terjadi ketidakcocokan antara karakter “(spasi)” dan “A”. Karakter “(spasi)” tidak terdapat dalam tabel *BmBc* sehingga digantikan dengan tanda (*). Tanda (*) memiliki nilai sebesar 5 sehingga dilakukan pergeseran sebanyak 5 kali.

Pencarian algoritma *Raita* tahap ketiga yaitu melakukan pergeseran sebanyak 5 kali (sesuai dengan karakter “(spasi)”).

Tabel 2.10 Pencarian pada teks proses ketiga

T	R	Y	A	N	R	I	D	H	O		V	A	L	B	A
P											V	A	L	B	A
											2	4	3	5	1

Dilihat pada Tabel 2.10 terjadi kecocokan antara teks dan pola. Seluruh pencocokan karakter menggunakan algoritma *Raita* telah selesai dan berhenti pada pencarian teks proses ketiga.

2.5 Penelitian yang Relevan

Berikut ini beberapa penelitian yang terkait dengan algoritma *Horspool* dan algoritma *Raita* :

1. Nasution Ade Mutiara Kartika (2016) dalam skripsi yang berjudul *Implementasi Algoritma Horspool Dalam Pembuatan Kamus Istilah Isikologi pada Platform Android*. Didalam skripsi tersebut dijelaskan bahwa algoritma *Horspool* terdiri tahap praproses dan tahap pencarian. Pada tahap praproses dilakukan observasi *pattern* terhadap teks untuk membangun sebuah tabel *bad-match* yang berisi nilai *shift* ketika ketidakcocokan antara *pattern* dan teks terjadi. Pada tahap pencarian dilakukan perbandingan karakter paling kanan *pattern* terhadap *window* ketika ada ketidakcocokan maka karakter paling kanan pada *window* berfungsi sebagai landasan untuk menentukan jarak *shift* yang akan dilakukan. Prosedur ini dilakukan berulang-ulang sampai *window* berada pada akhir teks atau *pattern* cocok dengan teks.

2. Nasution Nurhasbiah (2016) dalam skripsi yang berjudul *Implementasi Algoritma Raita Dalam kamus Bahasa Indonesia – Mandailing Berbasis Android*. Didalam skripsi tersebut dijelaskan algoritma *Raita* tidak seperti pencarian string lainnya seperti brute force, knuth-morris-pratt yang mempunyai cara kerja membandingkan satu persatu karakter dari kiri ke kanan. *Raita* membandingkan karakter dari kanan ke kiri kemudian tengah dan memiliki loncatan karakter yang besar sesuai nilai karakter pada tabel *BmBc* sehingga mempercepat pencarian *string* karena dengan hanya memeriksa akhir karakter pola dengan karakter teks, dapat langsung diketahui bahwa *string* yang dicari tidak ditemukan sehingga dapat digeser ke posisi berikutnya.
3. Tambun Evelyn Dwi (2010) dalam skripsi yang berjudul *Perbandingan Penggunaan Algoritma BM dan Algoritma Horspool Pada Pencarian String Dalam bahasa Medis*. Didalam skripsi tersebut dijelaskan algoritma *horspool* adalah penyerderhanaan algoritma boyer-moore. Algoritma ini dibuat karena kurang cocok di implementasikan untuk pencarian *string* dengan varian karakter yang sedikit seperti *binary string*. Algoritma *Horspool* menggunakan karakter paling kanan pada *window* untuk menentukan jarak shift yang akan dilakukan
4. Ibrahim Amin Mubark Alamin Ibrahim & Elgili Mustafa (2015) dalam jurnal yang berjudul *Comparison Criteria Between Matching Algorithms Teks Application On Horspool and Brute Force Algorithms*. Penelitian tersebut dijelaskan perbedaaan pada algoritma *Horspool* dan *Brute Force* diuji dengan jumlah perbandingan untuk satu karakter teks dan eksekusi waktu. Pada studi tersebut hasilnya adalah algoritma *Horspool* tidak lebih baik dari algoritma *Brute Force*
5. Nebel Markus E (2006) dalam jurnal yang berjudul *Fast String Matching By Using Probabilities; On Optimal Mismatch Variant Of Horspool's Algorithm*. Penelitian tersebut dijelaskan ide dari memodifikasi algoritma ini untuk mengubah urutan simbol dari pola dibandingkan dengan simbol-simbol teks sehingga probabilitas untuk ketidakcocokan dimaksimalkan.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

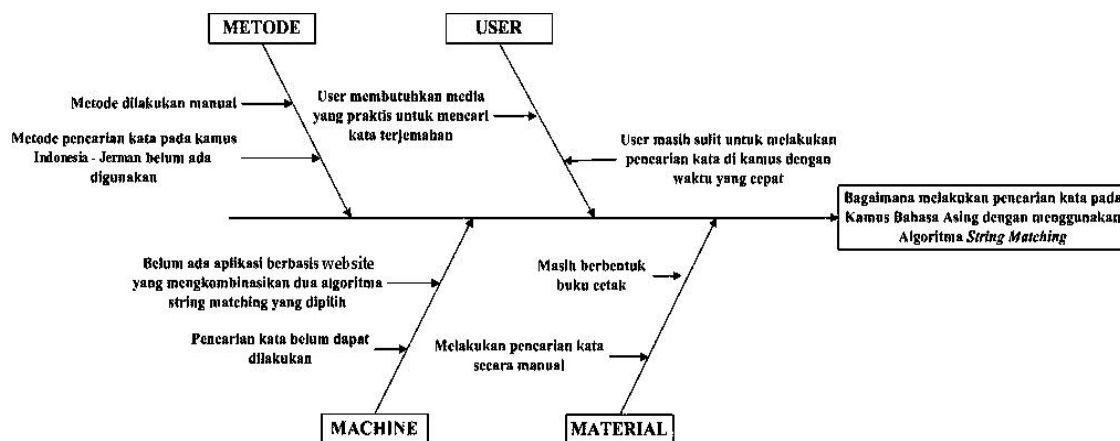
Analisis sistem adalah sebuah teknik pemecahan masalah dimana sistem diuraikan menjadi komponen-komponen dengan tujuan untuk mempelajari kinerja masing-masing komponen tersebut dalam mencapai tujuan sistem. Tahapan ini dilakukan agar pada saat proses perancangan aplikasi tidak terjadi kesalahan dan sistem dapat berjalan sesuai dengan tujuan utama. Ada dua tahapan analisis dalam tugas akhir ini yaitu analisis masalah dan analisis persyaratan. Analisis masalah untuk memahami kelayakan masalah dan analisis persyaratan untuk menjelaskan fungsifungsi yang ditawarkan dan mampu dikerjakan oleh sistem

3.1.1. Analisis masalah

Analisis masalah merupakan proses mengidentifikasi sebab dan akibat dibangunnya sebuah sistem agar sistem yang akan dibangun tersebut dapat berjalan sebagaimana mestinya sesuai dengan tujuan dari sistem itu. Selama ini jika seseorang ingin mempelajari bahasa asing, media yang digunakan untuk memperlancar penguasaan kosakatanya adalah melalui kamus. Aplikasi kamus didalam Berbasis *Website* tidak mencari kata secara manual. Tetapi dengan adanya fasilitas pencarian pada aplikasi kamus tersebut mempermudah *user* mendapatkan kata yang ingin dicarinya. Dengan memanfaatkan salah satu dari Algoritma String Matching seperti Algoritma *Horspool* dan Algoritma *Raita*, maka mempermudah dan mempercepat pencarian kata dalam kamus.

Pencarian kata adalah masalah yang akan diselesaikan dengan menggunakan sistem ini. Untuk mengidentifikasi masalah tersebut digunakan diagram Ishikawa (fishbone diagram). Diagram Ishikawa berbentuk seperti ikan yang strukturnya terdiri dari kepala ikan (fish's head) dan tulang-tulang ikan (fish's bones). Nama atau judul dari

masalah yang diidentifikasi terletak pada bagian kepala ikan. Sedangkan tulang-tulang ikan menggambarkan penyebab-penyebab masalah tersebut. Diagram Ishikawa pada sistem ini dapat dilihat pada Gambar 3.1



Gambar 3.1 Diagram Ishikawa

Pada **Gambar 3.1** dapat dilihat bahwa terdapat empat kategori penyebab masalah pada penelitian pencarian kata pada kamus Bahasa Asing dengan menggunakan Algoritma *String Matching* yang digambarkan dengan tanda panah yang mengarah ke tulang utama, yaitu berkaitan dengan pengguna (*user*), bahan (*material*), metode (*method*) dan media/alat yang terlibat (*machine*). Setiap detail penyebab masalah tersebut digambarkan dengan tanda panah yang mengarah ke masing-masing kategori.

3.1.2. Analisis persyaratan

Untuk membangun sebuah sistem, perlu dilakukan sebuah tahap analisis persyaratan. Terdapat dua bagian pada analisis persyaratan, yaitu persyaratan fungsional dan persyaratan non-fungsional. Persyaratan fungsional mendeskripsikan aktivitas yang disediakan suatu sistem. Sedangkan Persyaratan nonfungsional mendeskripsikan fitur, karakteristik dan batasan lainnya

3.1.2.1 Analisis Persyaratan Fungsional

Persyaratan fungsional disini mendeskripsikan tentang sistem yang disediakan.

Terdapat beberapa hal yang menjadi persyaratan fungsional pada Aplikasi Kamus Bahasa Indonesia-Jerman, antara lain :

1. Sistem melakukan pencocokan *string* melalui kata yang diinput oleh pengguna.
2. Sistem dapat menghasilkan kata terjemahan dari *inputan* yang dicari dengan menggunakan Algoritma *Horspool* dan Algoritma *Raita*
3. Sistem melakukan *inputan* dan hasil pencarian hanya berupa kata dan tidak berupa kalimat

3.1.2.2 Analisis Persyaratan Non-Fungsional

Persyaratan non fungsional sistem merupakan karakteristik atau batasan yang menentukan kepuasan pada sebuah sistem seperti kinerja, kemudahan penggunaan, biaya, dan kemampuan sistem bekerja tanpa mengganggu fungsionalitas sistem lainnya, Terdapat beberapa persyaratan non-fungsional yang harus dipenuhi diantaranya :

1. Performa

Sistem yang akan dibangun harus dapat menampilkan hasil pencarian yang sesuai dengan apa yang dicari.

2. Mudah digunakan

Sistem yang akan dibangun harus mudah digunakan (*user friendly*), artinya sistem ini akan mudah digunakan oleh *user* dengan tampilan yang sederhana dan dapat dimengerti.

3. Hemat biaya

Sistem yang dibangun tidak memerlukan perangkat tambahan ataupun perangkat pendukung lainnya yang dapat mengeluarkan biaya.

4. Kontrol

Sistem yang dibangun mampu menampilkan pesan ketika tidak ada *inputan* yang akan dicari.

5. Manajemen Kualitas

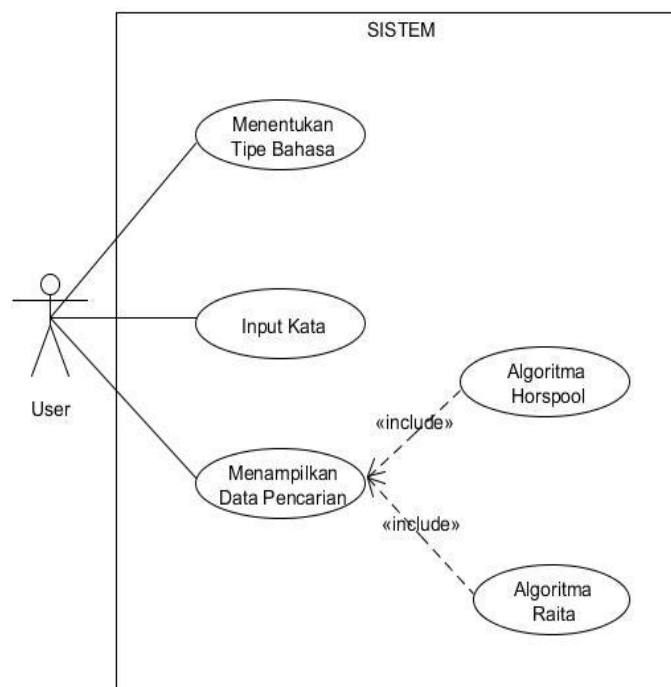
Sistem yang akan dibangun harus memiliki kualitas yang baik yaitu tidak mempersulit *user* untuk melakukan pencarian kata.

3.1.3. Pemodelan Sistem

Pada bagian pemodelan sistem, akan dirancang bagaimana nantinya sistem diharapkan bekerja. Pemodelan ini menggunakan UML (*Unified Modelling Language*).

3.1.3.1 Use case diagram

Use Case Diagram adalah sebuah diagram yang dapat merepresentasikan interaksi yang terjadi antara user dengan sistem. Diagram *use case* ini mendeskripsikan siapa saja yang menggunakan sistem dan bagaimana cara mereka berinteraksi dengan sistem. *Use Case Diagram* dari sistem yang akan dibangun dapat ditunjukkan pada **Gambar 3.2**.



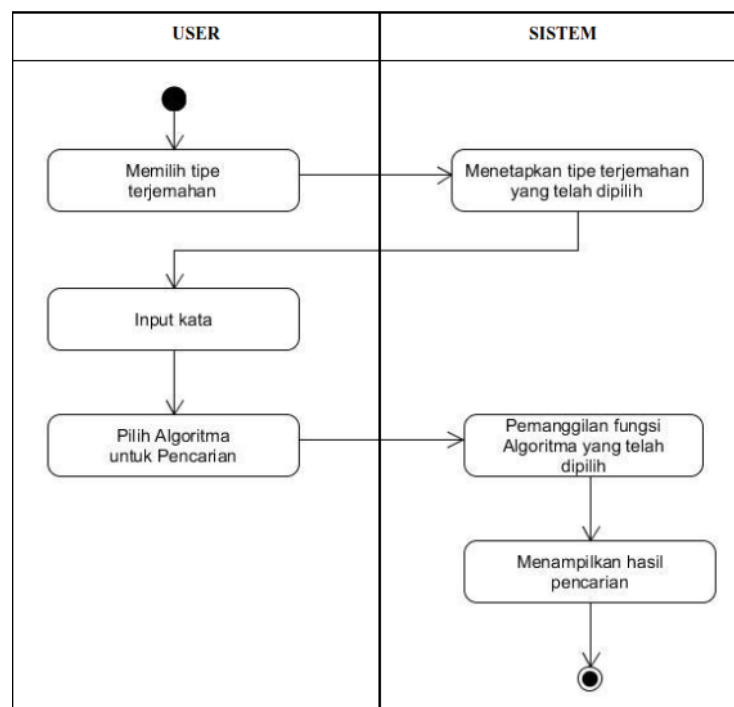
Gambar 3.2 Use Case Diagram Sistem

Didalam *use case* diagram dapat digambarkan bahwa terdapat 1(satu) orang aktor yang akan berperan yaitu *user*. Untuk memperoleh kata terjemahan maka *user* harus menentukan tipe terjemahan terlebih dahulu, seperti bahasa Indonesia-Jerman atau bahasa Jerman-Indonesia. Selanjutnya *user* harus memasukkan *input* berupa kata yang ingin diterjemahkan ke dalam *search box*. Selanjutnya *user* memilih Algoritma yang

ingin digunakan antara Algoritma *Horspool* dan Algoritma *Raita*. Setelah Algoritma dipilih lalu sistem akan melakukan pencarian dan menampilkan hasil terjemahan.

3.1.3.2 Activity Diagram

Activity diagram adalah diagram aktivitas yang mendeskripsikan proses kerja dalam sebuah sistem yang sedang berjalan. *Activity diagram* bertujuan untuk membantu memahami keseluruhan proses dan menggambarkan interaksi antara beberapa *Use Case Diagram*. *Activity diagram* dari sistem yang akan dibangun dapat ditunjukkan pada **Gambar 3.3**.

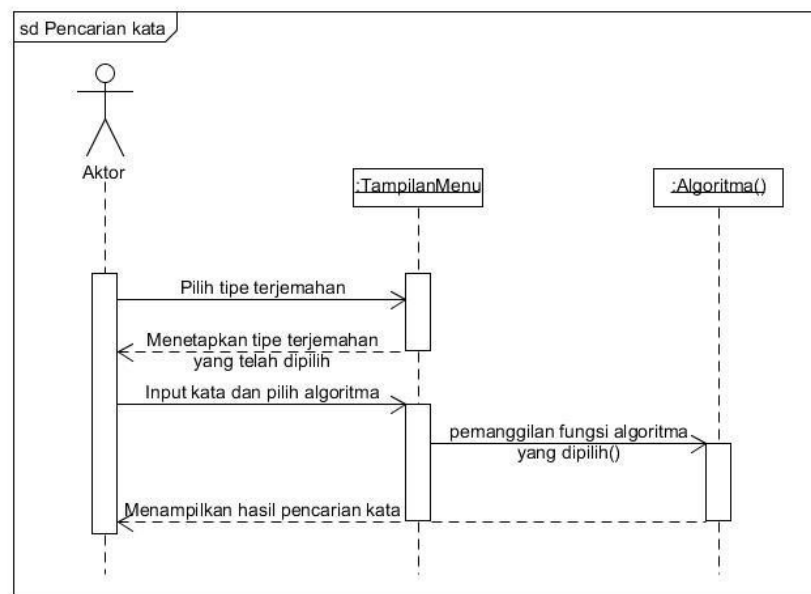


Gambar 3.3 Activity Diagram proses dekripsi

Didalam *Activity* diagram dapat dijelaskan bahwa *user* harus menentukan tipe terjemahan terlebih dahulu, seperti bahasa Indonesia – Jerman atau bahasa Jerman - Indonesia. Lalu sistem menetapkan tipe terjemahan yang telah dipilih. Selanjutnya *user* menginput kata yang ingin dicari. Lalu *user* memilih Algoritma untuk melakukan pencarian kata. Maka sistem melakukan pemanggilan fungsi pada Algoritma yang telah dipilih. Lalu sistem akan menampilkan hasil pencarian sesuai dengan Algoritma yang telah dipilih.

3.1.3.3 Sequence Diagram

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar objek pada sistem dalam sebuah urutan waktu atau rangkaian waktu. *Sequence Diagram* untuk sistem dapat dilihat pada Gambar 3.4.

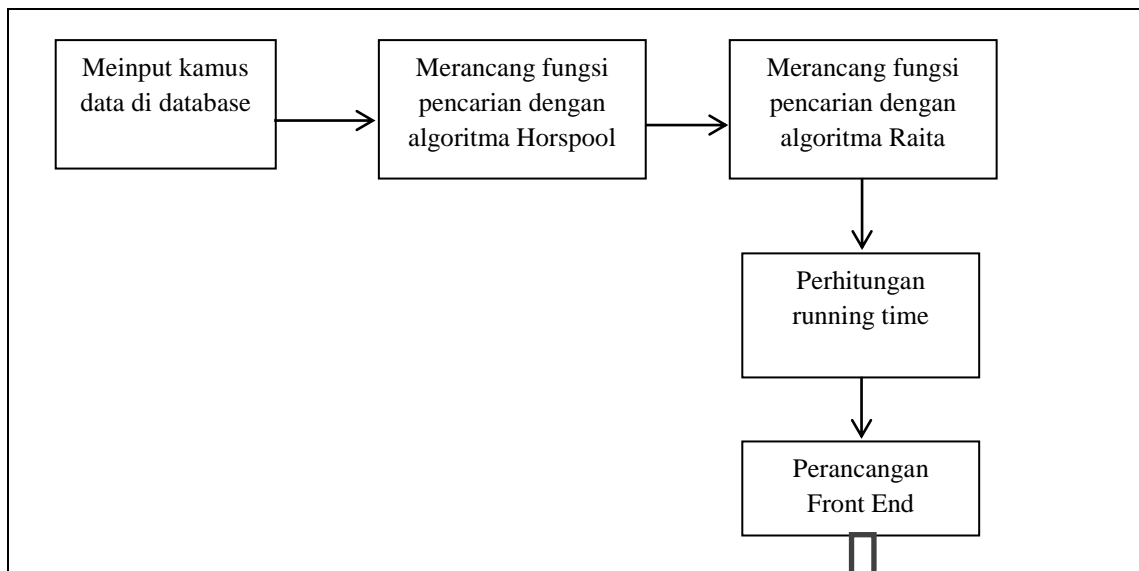


Gambar 3.4 Sequence Diagram Sistem

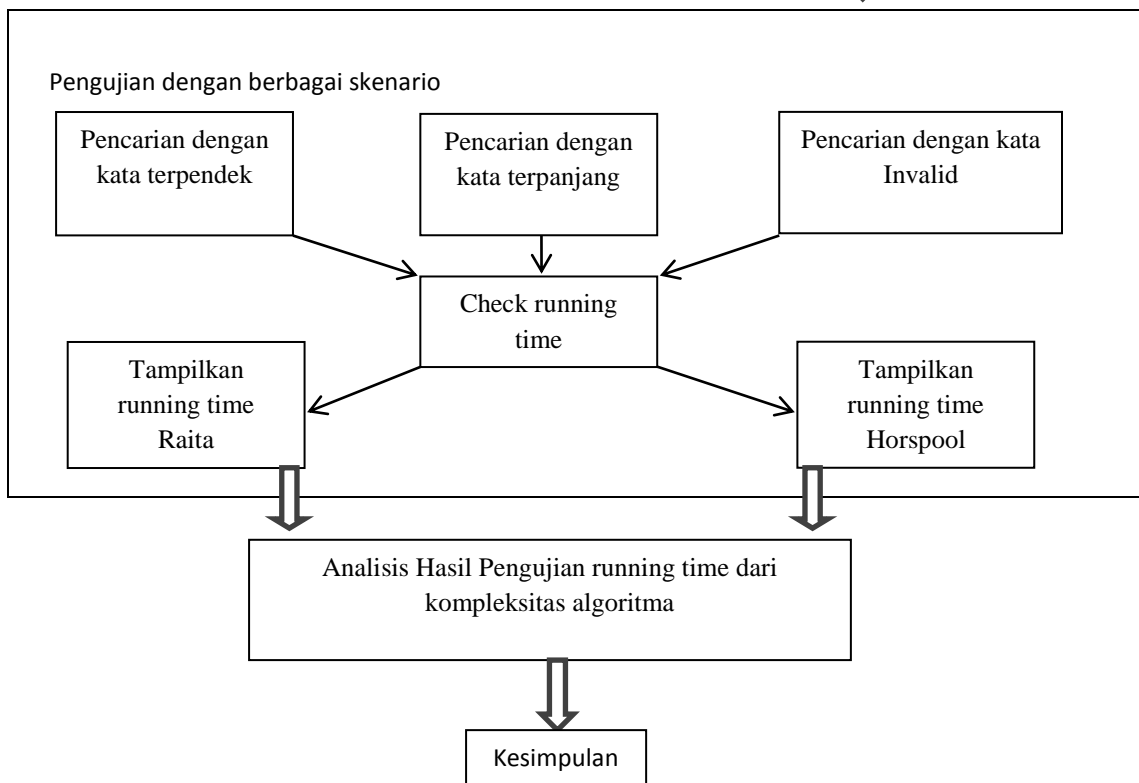
Pada tahap ini, yang dilakukan *user* adalah menentukan tipe terjemahan terlebih dahulu, seperti bahasa Indonesia-Jerman atau bahasa Jerman-Indonesia di tampilan menu. Lalu tampilan menu menetapkan tipe terjemahan yang telah dipilih. Selanjutnya yang dilakukan *user* adalah menginput kata yang ingin dicari dan memilih algoritma untuk melakukan pencarian, lalu akan memanggil fungsi Algoritma yang telah dipilih. Kemudian menampilkan hasil pencarian kata kepada *user*.

3.1.3.4 Arsitektur Umum

Tahap Perancangan

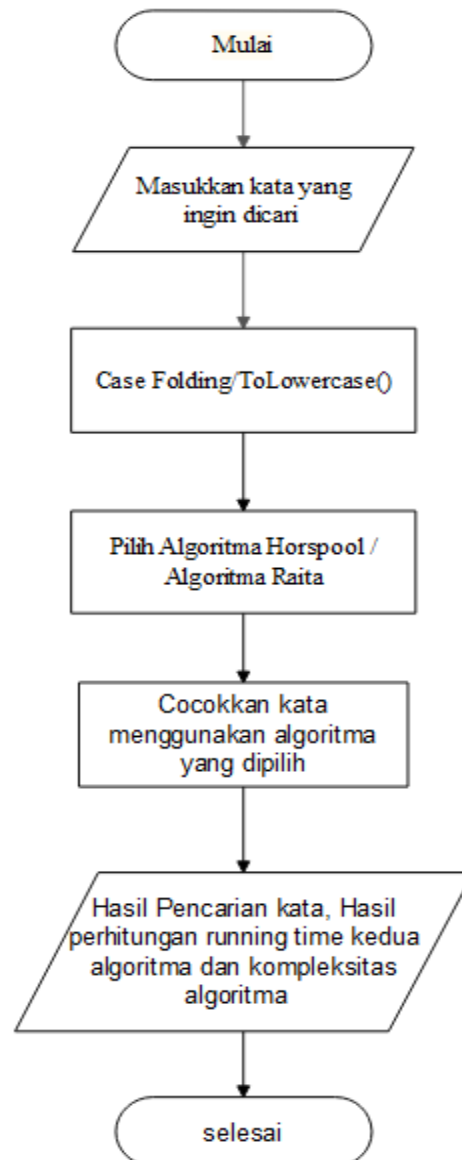


Tahap Pengujian



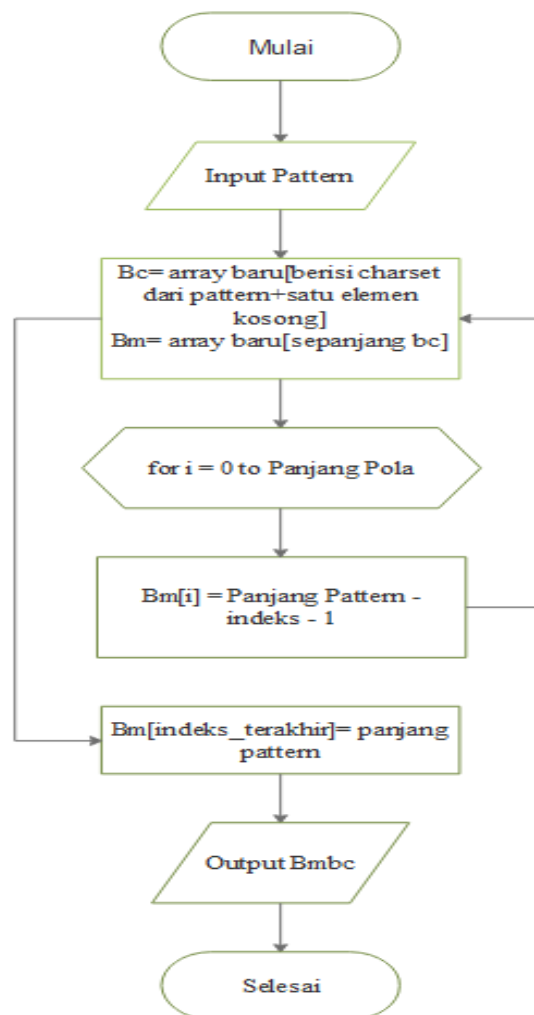
Gambar 3.5 Arsitektur Umum Sistem Kamus Indonesia - Jerman

3.1.3.5 Flowchart



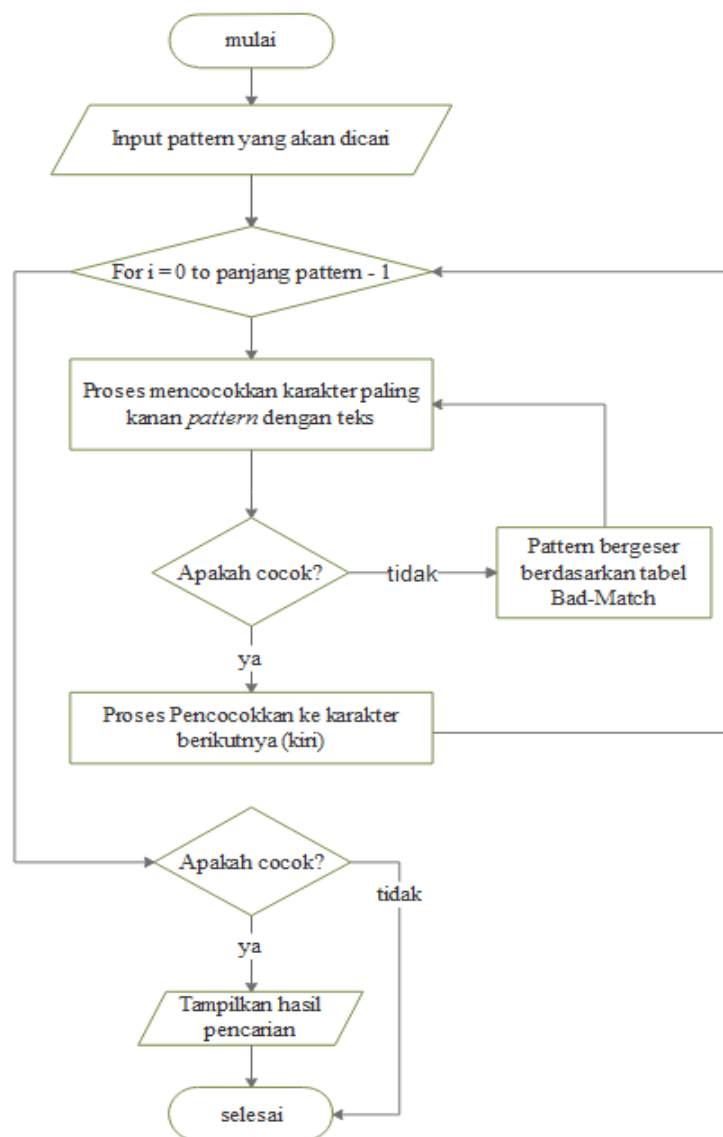
Gambar 3.6 Flowchart tampilan aplikasi kamus

Gambar 3.6 menggambarkan proses aplikasi dimulai dengan user menginput data yang ingin dicari. Sistem memproses kata menggunakan algoritma yang dipilih sebelumnya oleh user. Output yang keluar, berupa hasil pencarian kata, hasil perhitungan running time kedua algoritma dan kompleksitas algoritma



Gambar 3.7 Flowchart Proses *BmBc*

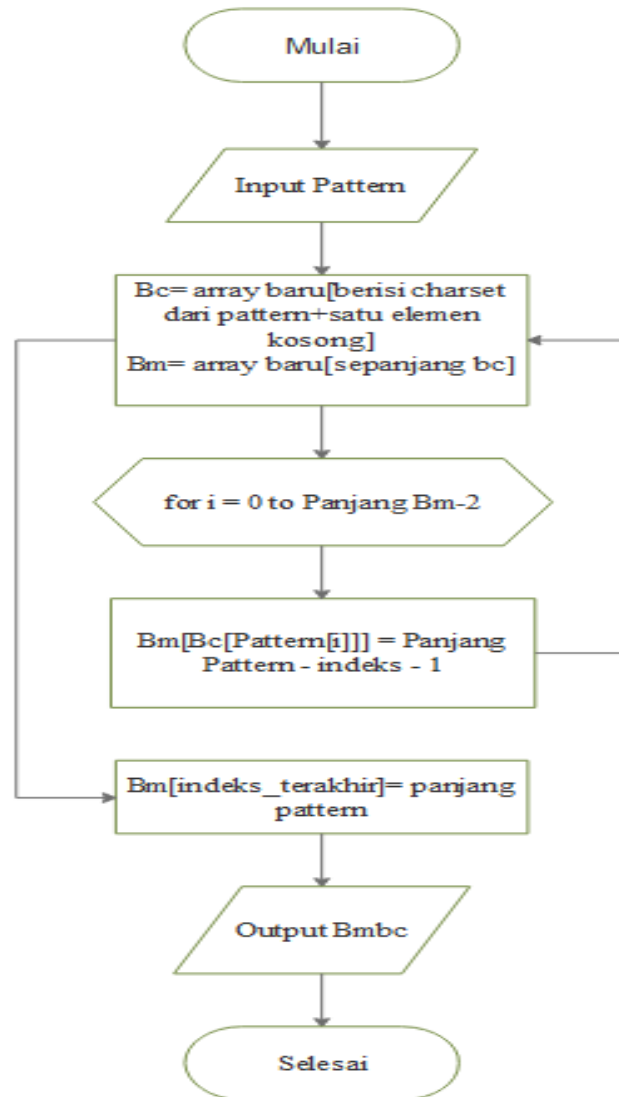
Gambar 3.8 menggambarkan proses perhitungan tabel BmBc. Proses ini dimulai dari memasukkan pola yang akan dicari kemudian membuat kolom BC yang berisikan charset dari *pattern* ditambah satu elemen kosong dan kolom BM sepanjang kolom BC. Dalam proses perhitungan tabel BmBc memiliki syarat yaitu for $i = 0$ to panjang - 1 dan untuk mencari nilai BM digunakan rumus $Bm[i] = \text{panjang } pattern - \text{indeks} - 1$. Apabila nilai indeks sudah tidak memenuhi persyaratan maka nilai BM paling akhir sama dengan panjang *pattern*. Selanjutnya akan ditampilkan hasil BmBc dan proses perhitungan BmBc selesai.



Gambar 3.9 *Flowchart Algoritma Horspool*

Gambar 3.9 menggambarkan proses pencarian algoritma *horspool*. Pertama *user* memulai dan memasukkan *pattern* yang akan dicari. Sistem melakukan proses pencarian dengan syarat $\text{for } i = 0 \text{ to panjang } \textit{pattern} - 1$. Selanjutnya sistem melakukan proses pencocokkan karakter paling kanan *pattern* dengan teks. Apabila terjadi ketidakcocokkan maka *pattern* bergeser sebesar nilai yang ada di tabel BmBc dan melakukan proses pencocokkan kembali. Apabila terjadi kecocokkan antara *pattern* dan teks maka akan dicocokkan dengan karakter berikutnya (kiri). Kemudian sistem mencocokkan kembali *pattern* dan teks apabila tidak cocok maka hasil

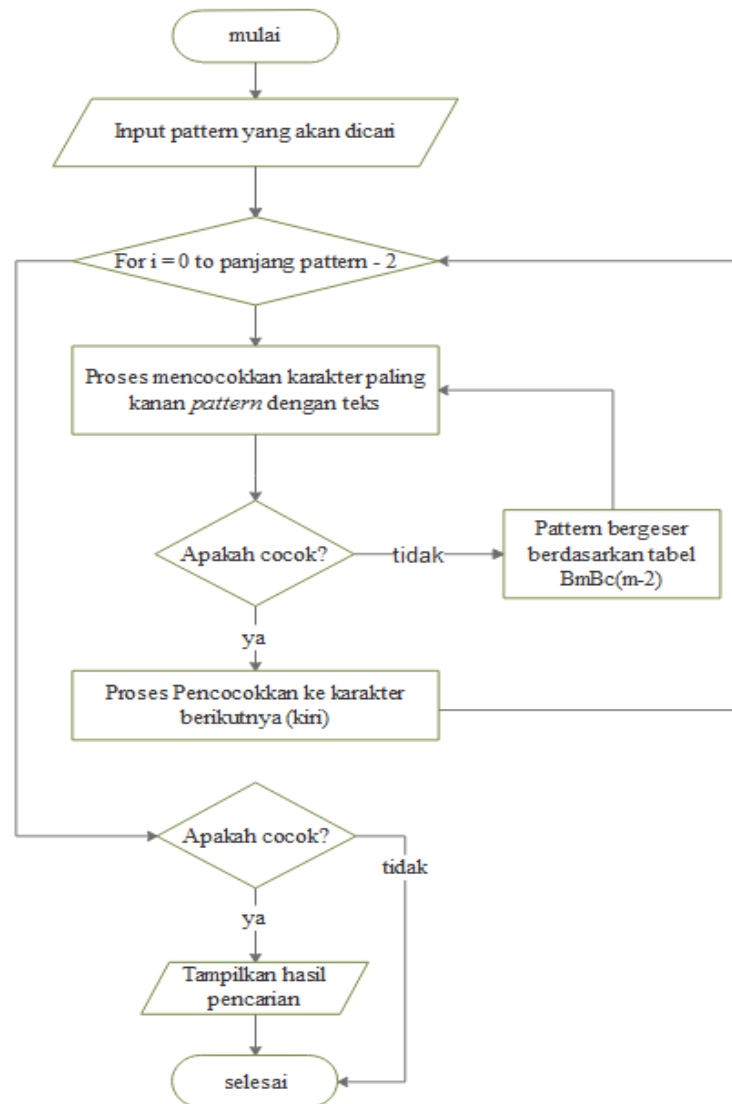
pencarian selesai dan apabila terjadi kecocokkan maka sistem akan menampilkan hasil pencarian dan selesai.



Gambar 3.10 Flowchart tabel *BmBc*

Gambar 3.10 menggambarkan proses perhitungan tabel *BmBc*. Proses ini dimulai dari memasukkan pola yang akan dicari kemudian membuat kolom BC yang berisikan charset dari *pattern* ditambah satu elemen kosong dan kolom BM sepanjang kolom BC. Dalam proses perhitungan tabel *BmBc* memiliki syarat yaitu $\text{for } i = 0 \text{ to panjang } Bm - 2$ dan untuk mencari nilai BM digunakan rumus $Bm[Bc[Pattern[i]]] = \text{panjang } pattern - \text{indeks} - 1$. Apabila nilai indeks sudah tidak memenuhi persyaratan maka

nilai BM paling akhir sama dengan panjang *pattern*. Selanjutnya akan ditampilkan hasil BmBc dan proses perhitungan BmBc selesai.



Gambar 3.11 Flowchart Algoritma Raita

Gambar 3.11 menggambarkan proses pencarian algoritma Raita. Pertama *user* memulai dan memasukkan *pattern* yang akan dicari. Sistem melakukan proses pencarian dengan syarat $\text{for } i = 0 \text{ to } \text{panjang } \text{pattern} - 2$. Selanjutnya sistem melakukan proses pencocokkan karakter paling kanan *pattern* dengan teks. Apabila terjadi ketidakcocokkan maka *pattern* bergeser sebesar nilai yang ada di tabel BmBc dan melakukan proses pencocokkan kembali. Apabila terjadi kecocokkan antara *pattern* dan teks maka akan dicocokkan dengan karakter berikutnya (kiri). Kemudian

sistem mencocokkan kembali *pattern* dan teks apabila tidak cocok maka hasil pencarian selesai dan apabila terjadi kecocokkan maka sistem akan menampilkan hasil pencarian dan selesai.

3.2 Perancangan *Interface*

Pada perancangan sistem terdapat pembuatan *Graphic User Interface* (GUI) yang akan mempermudah user dalam menggunakan sistem yang ada. Pada sistem, terdapat 5 buah *Form* yang akan ditampilkan antara lain *Form Beranda*, *Form Input Data*, dan *Form Tentang*.

3.2.1 *Form Beranda*

Pada halaman Menu terdapat sistem pengujian, dimana sistem ini menampilkan hasil dari pencarian algoritma *Horspool* dan *Raita* serta pemilihan bahasa antara Indonesia – Jerman atau Jerman – Indonesia, seperti terlihat pada Gambar 3.12

The diagram shows a web interface layout for a dictionary system. It includes a header with a logo (1), navigation buttons for 'Home' (2) and 'Tentang' (3), a title 'Kamus Indonesia - Jerman Dan Jerman Indonesia', and three input fields for 'Pilihan Bahasa' (4), 'Horspool' (5), and 'Raita' (6). Below these is a 'Search' button (7) and a 'Button8' (8). At the bottom are two large output areas labeled 'Tabel Horspool' (9) and 'Tabel Raita' (10).

Gambar 3.12 *Form Beranda* pada Sistem

Keterangan :

1. *Picture*

Menampilkan gambar dari LOGO Universitas.

2. *Button Home*

Untuk menampilkan halaman utama

3. *Button Tentang*

Berguna untuk menampilkan halaman tentang.

4. *Radio Button*

Berguna untuk menampilkan pilihan Indonesia – Jerman atau Jerman – Indonesia.

5. *Text*

Berguna untuk menampilkan hasil *running time* dari *Horspool*

6. *Text*

Berguna untuk menampilkan hasil *running time* dari *Raita*

7. *Text Input*

Untuk menginputkan kata yang ingin dicari oleh *user*.

8. *Button*

Memproses pencarian dari inputan teks oleh *user*.

9. Tabel

Menampilkan hasil dari data kamus sesuai dengan algoritma *horspool* data yang terdapat dalam *database*.

10. Tabel

Menampilkan hasil dari data kamus sesuai dengan algoritma *Raita* data yang terdapat dalam *database*.

3.2.2 Form Input Data

Pada halaman *input data* terdapat tabel yang berisi dengan data yang terdapat dalam database kamus serta dapat menambahkan kata untuk kamus serta menghapus atau mengedit data seperti dapat dilihat pada Gambar 3.13.

Gambar 3.13 Form Admin

Keterangan:

1. *Picture*

Menampilkan gambar dari LOGO Universitas.

2. *Button Home*

Untuk menampilkan halaman utama

3. *Button Tentang*

Berguna untuk menampilkan halaman tentang.

4. *Textfield*

Untuk diisi kata bahasa indonesia

5. *Textfield*

Untuk diisi kata bahasa jerman

6. *Button*

Untuk memasukan data bahasa indonesia dan bahasa jerman ke dalam database

7. *Tabel*

Untuk menampilkan seluruh data yang terdapat dalam database

3.2.3 Form Tentang

Pada *Form* tentang berisi informasi tentang penulis yang membuat aplikasi tersebut, pada aplikasi dapat dilihat pada Gambar 3.14.

The image shows a web application interface for an 'About' form. At the top left is a logo placeholder labeled '1'. To the right are two buttons: 'Home' labeled '2' and 'Tentang' labeled '3'. Below these is a title 'Kamus Indonesia - Jerman Dan Jerman Indonesia'. Under the title are three text input fields: the first labeled '4' for the author's name, the second labeled '5' for the author's NIM, and the third labeled '6' for the research title.

Gambar 3.14 Form Tentang

Keterangan :

1. *Picture*

Menampilkan gambar dari LOGO Universitas.

2. *Button Home*

Untuk menampilkan halaman utama

3. *Button Tentang*

Berguna untuk menampilkan halaman tentang.

4. *Textfield*

Untuk menampilkan nama penulis

5. *Texfield*

Berisi tampilan NIM penulis

6. *Textfield*

Menampilkan judul dari penelitian

BAB 4

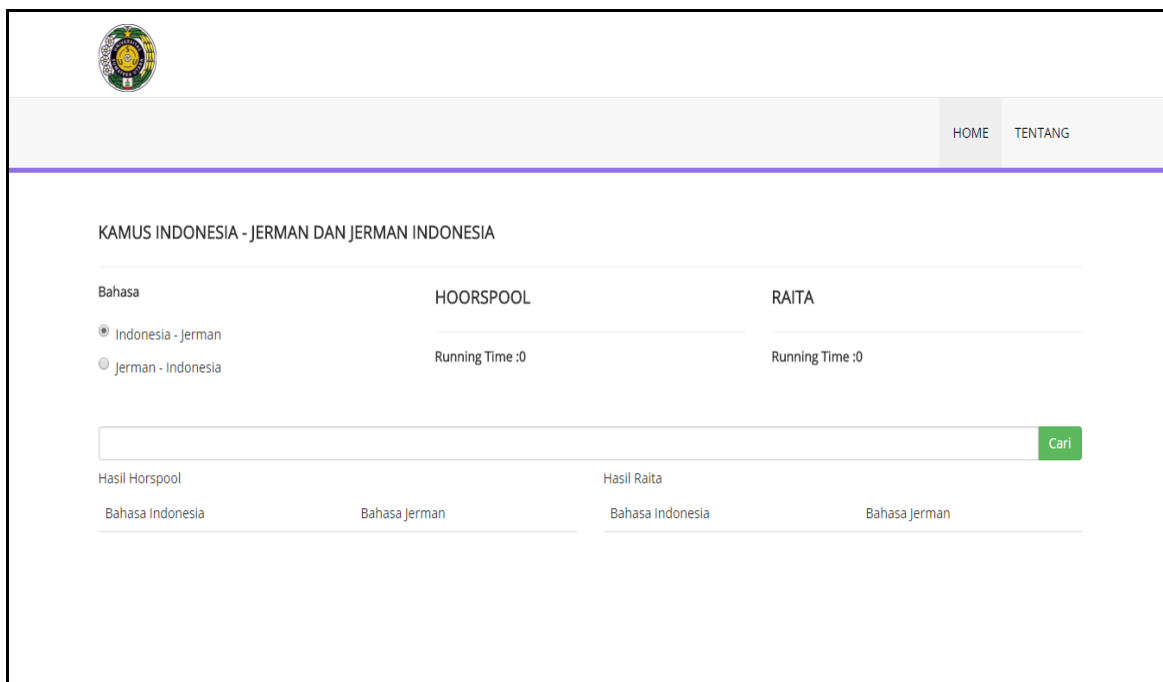
IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi

Implementasi sistem merupakan lanjutan dari tahap analisis dan perancangan sistem. Sistem ini dibangun dengan menggunakan bahasa pemrograman *php*, *html*, *javascript*, *css* dan *framework bootstrap* menggunakan *Software Adobe dreamweaver*. Pada sistem ini terdapat 3 (tiga) tampilan halaman, yaitu Halaman *Home*, *Adminsitrator*, dan Halaman Tentang.

4.1.1. Halaman Home

Halaman *Home* merupakan halaman yang berisi tautan ke halaman yang lain dan merupakan halaman pengujian yang ditampilkan sistem. Halaman *Home* dapat dilihat pada Gambar 4.1.



Gambar 4.1 Implementasi Halaman *Home*

4.1.2. Halaman *Admin*

Halaman *admin* adalah halaman untuk mengelola data kata yang terdapat didalam database seperti terlihat pada Gambar 4.2.

KAMUS INDONESIA - JERMAN DAN JERMAN INDONESIA

Indonesia
Indonesia

Jerman
Jerman

Input Data Kamus

DATA KAMUS

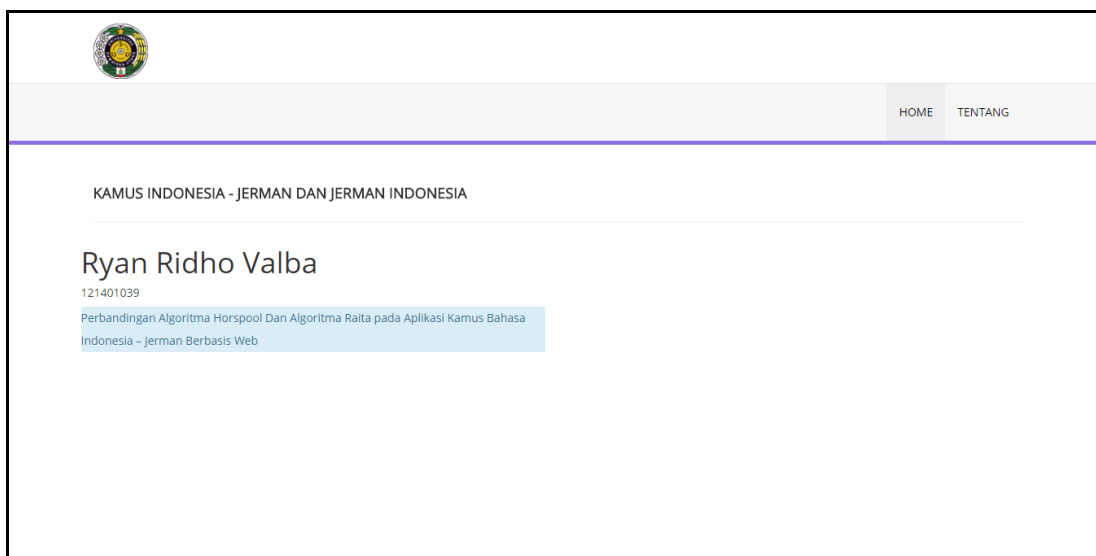
Show 10 entries Search:

NO	Indonesia	Jerman
1	Abad	Das Jahrhundert
2	Abadi	Ewig
3	Abjad	Das Alphabet
4	Aborsi	Die Abtreibung
5	Absolut	Absolut
6	Abu-Abu	Grau

Gambar 4.2. Implementasi Halaman *Admin*

4.1.3. Halaman *Tentang*

Halaman *tentang* berisi tentang informasi penulis dan pembuat sistem sebagaimana yang terlihat pada Gambar 4.3.



Gambar 4.3. Implementasi Tentang

4.2. Implementasi Sistem

Pada penelitian ini sistem yang akan dibuat merupakan perbandingan algoritma *Horspool* dan *Raita*. Metode algoritma *Horspool* yaitu melakukan nilai pergeseran karakter yang paling kanan dari *window*. Selanjutnya nilai *shift* dihitung untuk semua karakter. pada tahap selanjutnya, dibandingkan *pattern* dari kanan ke kiri hingga kecocokan atau ketidakcocokan *pattern* terjadi. Karakter yang paling kanan pada *window* digunakan sebagai indeks dalam melakukan nilai *shift*. Dalam kasus ketidakcocokan (karakter tidak terdapat pada *pattern*) terjadi, *window* digeser oleh panjang dari sebuah *pattern*. Jika tidak, *window* digeser menurut karakter yang paling kanan pada *pattern*.

Pada metode algoritma *Raita* yaitu membandingkan karakter yang terakhir dari pola dari karakter paling kanan dari *window*. pada tahap selanjutnya, dibandingkan *pattern* dari kanan ke kiri hingga kecocokan atau ketidakcocokan *pattern* terjadi. Jika mereka cocok, kemudian karakter pertama dari pola teks paling kiri dari *window* juga dibandingkan. Jika mereka cocok, maka akan dibandingkan karakter tengah pola dengan karakter teks tengah *window*.

Cara kerja sistem yaitu *user* akan meng-*input string*, lalu *user* akan memilih pencarian Indonesia – Jerman atau Jerman – Indonesia mana yang ingin dicari

oleh *user*. Lalu sistem akan melakukan pencarian *string* dengan algoritma *Horspool* terlebih dahulu dan menyimpan *runningtime* nya lalu pencarian *string* dengan menggunakan algoritma *Raita* dan menyimpan *running time* nya. Sistem akan menampilkan hasil pencarian *string*. Seperti yang dapat dilihat pada gambar 4.4 dan gambar 4.5.

Contoh *string* : “aku”

KAMUS INDONESIA - JERMAN DAN JERMAN INDONESIA

Bahasa

☒ Indonesia - Jerman

☐ Jerman - Indonesia

HOORSPPOOL

Running Time :0.33285999298096

Total :3

RAITA

Running Time :0.33034896850586

Total :3

Cari

Hasil Horspool

Bahasa Indonesia	Bahasa Jerman
Aku	Ich
Fakultas	Die Fakultæt
Takut	Angst haben

Hasil Raita

Bahasa Indonesia	Bahasa Jerman
Takut	Angst haben
Fakultas	Die Fakultæt
Aku	Ich

Gambar 4.4 Pengujian Pencarian kata “aku” Indonesia – Jerman

Contoh pencarian Indonesia – Jerman, contoh *string* : “Aku”

Halaman ini adalah halaman pencarian kata dengan *user* meng-*input* contoh *string* “aku” dengan menggunakan 2 algoritma *string matching* yaitu algoritma *Horspool* dan *Raita* setelah itu *user* memilih Indonesia – Jerman pada bahasa yang dicari. Lalu di klik *button* cari kemudian di mulai pencarian kata jika ditemukan. maka di tampilkan kata yang cocok di kedua tabel, *running time* dan jumlah data pada kedua pencarian. Sebelum melakukan proses pencocokan, algoritma *Horspool* dan *Raita* memiliki tahap *preprocessing* untuk menentukan nilai pergeseran. Tahapan untuk membuat tabel nilai pergeseran BmBc pada fase *preprocessing* algoritma *Horspool* dan *Raita* sudah dipaparkan sebelumnya pada bab kedua.

Tabel nilai pergeseran BmBc untuk *pattern* AKU dapat dilihat pada tabel 4.1 berikut ini.

Tabel 4.1 Tabel *BmBc*

	BC	BM
0	A	2
1	K	1
2	*	3

Proses pencarian kata pada algoritma *Horspool*.

Pada proses pencarian diberikan teks dari database dengan *pattern* berikut

Teks = fakultas

Pattern = aku

a. algoritma *Horspool* melakukan perbandingan antara teks dengan pola yang mana pencocokan dilakukan pada karakter terkanan *pattern*. Tabel *BmBc* digunakan untuk melewati karakter ketika ketidakcocokan terjadi. Hal ini dapat dilihat pada tabel 4.2.

Tabel 4.2 Pencarian kata pada teks pertama

M	1	2	3	4	5	6	7	8
T	F	A	K	U	L	T	A	S
P	A	K	U					

Pada tabel 4.2 terdapat ketidakcocokan Karakter “K” dengan “U”. Karakter “K” terdapat pada tabel BmBc sehingga dilakukan pergeseran sebanyak 1 kali.

b. Setelah melakukan pergeseran sebesar karakter yang ada pada tabel *Bmbc* maka akan dilakukan pencocokan antara teks dan *pattern*, apabila terjadi kecocokan antara karakter dalam teks dengan karakter pada *pattern* maka akan dilakukan pencocokan karakter berikutnya dari kiri. Hal ini dapat dilihat pada tabel 4.3.

Tabel 4.3 Pencarian kata pada teks kedua

M	1	2	3	4	5	6	7	8
T	F	A	K	U	L	T	A	S
P		A	K	U				

Pada tabel 4.3 terdapat kecocokan antara *pattern* dan teks. Seluruh pencocokan karakter menggunakan algoritma *Horspool* telah selesai dan berhenti pada pencarian teks proses kedua.

Dari 1 langkah menghasilkan 1 kali pergeseran ke kanan, yaitu :

Langkah ke 1 : menghasilkan nilai pergeseran sejauh 1 ($BmBc[1]$).

Proses kata pada algoritma *Raita*.

Pada proses pencarian diberikan teks dari database dengan *pattern* berikut

Teks = fakultas

Pattern = aku

a. algoritma *Raita* melakukan perbandingan antara teks dengan pola yang mana pencocokan dilakukan pada karakter terkanan *pattern*. Tabel *BmBc* digunakan untuk melewati karakter ketika ketidakcocokan terjadi. Hal ini dapat dilihat pada tabel 4.4.

Tabel 4.4 Pencarian kata pada teks pertama

M	1	2	3	4	5	6	7	8
T	F	A	K	U	L	T	A	S
P	A	K	U					

Pada tabel 4.4 terdapat ketidakcocokan Karakter “K” dengan “U”. Karakter “K” terdapat pada tabel *BmBc* sehingga dilakukan pergeseran sebanyak 1 kali.

b. Setelah melakukan pergeseran sebesar karakter yang ada pada tabel *Bmbc* maka akan dilakukan pencocokan antara teks dan *pattern*, apabila terjadi kecocokan antara karakter dalam teks dengan karakter pada *pattern* maka akan dilakukan pencocokan karakter dari kanan, kiri awal, dan tengah karakter. Hal ini dapat dilihat pada tabel 4.5.

Tabel 4.5 Pencarian kata pada teks kedua

M	1	2	3	4	5	6	7	8
T	F	A	K	U	L	T	A	S
P		A	K	U				

Pada tabel 4.5 terdapat kecocokan antara *pattern* dan teks. Seluruh pencocokan karakter menggunakan algoritma *Raita* telah selesai dan berhenti pada pencarian teks proses kedua.

Dari 1 langkah menghasilkan 1 kali pergeseran ke kanan, yaitu :

Langkah ke 1 : menghasilkan nilai pergeseran sejauh 1 ($BmBc[1]$).

KAMUS INDONESIA - JERMAN DAN JERMAN INDONESIA

Bahasa: ☒ Indonesia - Jerman ☐ Jerman - Indonesia

HOORSPOOL: Running Time :0.34493494033813, Total :39

RAITA: Running Time :0.35892200469971, Total :39

Hasil Horspool:

Bahasa Jerman	Bahasa Indonesia
Sich Lohnen	Ada Gunanya
Der Speichel	Air Liur
Ich	Aku
Natuerlich	Alami

Hasil Raita:

Bahasa Jerman	Bahasa Indonesia
Sich Lohnen	Ada Gunanya
Der Speichel	Air Liur
Ich	Aku
Natuerlich	Alami

Gambar 4.5 Pengujian Pencarian Kata “ich” Jerman – Indonesia

Contoh pencarian Jerman – Indonesia, contoh *string* : “Ich”

Halaman ini adalah halaman pencarian kata dengan *user* meng-*input* contoh *string* “ich” dengan menggunakan 2 algoritma *string matching* yaitu algoritma *Horspool* dan *Raita* setelah itu *user* memilih Jerman – Indonesia pada bahasa yang dicari. Lalu di klik *button* cari kemudian di mulai pencarian kata jika ditemukan. maka di tampilkan kata yang cocok di kedua tabel, *running time* dan jumlah data pada kedua pencarian. Tahapan untuk membuat tabel nilai pergeseran $BmBc$ pada fase *preprocessing*

algoritma *Horspool* dan *Raita* sudah dipaparkan sebelumnya pada bab kedua. Tabel nilai pergeseran BmBc untuk *pattern* ICH dapat dilihat pada tabel 4.6 berikut ini.

Tabel 4.6 Tabel *BmBc*

	BC	BM
0	I	2
1	C	1
2	*	3

Proses pencarian kata pada algoritma *Horspool*.

Pada proses pencarian diberikan teks dari database dengan *pattern* berikut

Teks = natuerlich

Pattern = ich

a. Setelah melakukan pergeseran sebesar karakter yang ada pada tabel *Bmbc* maka akan dilakukan pencocokan antara teks dan *pattern*, apabila terjadi ketidakcocokan antara karakter dalam teks dengan karakter pada *pattern* maka akan dilakukan pergeseran sebesar nilai karakter pada tabel BmBc. Hal ini dapat dilihat pada tabel 4.7.

Tabel 4.7 Pencarian kata pada teks pertama

M	1	2	3	4	5	6	7	8	9
T	N	A	T	U	R	L	I	C	H
P	I	C	H						

Pada tabel 4.7 terdapat ketidakcocokan karakter “T” dengan “H”. Karakter “T” tidak terdapat pada tabel BmBc sehingga dilakukan pergeseran sebanyak 3 kali.

b. Setelah melakukan pergeseran sebesar karakter yang ada pada tabel *Bmbc* maka akan dilakukan pencocokan antara teks dan *pattern*, apabila terjadi ketidakcocokan antara karakter dalam teks dengan karakter pada *pattern* maka akan dilakukan kembali pergeseran sesuai dengan nilai karakter pada tabel BmBc. Hal ini dapat dilihat pada tabel 4.8.

Tabel 4.8 Pencarian kata pada teks kedua

M	1	2	3	4	5	6	7	8	9
T	N	A	T	U	R	L	I	C	H
P				I	C	H			

Pada tabel 4.8 terdapat ketidakcocokan karakter “L” dengan “H”. Karakter “L” tidak terdapat pada tabel BmBc sehingga dilakukan pergeseran sebanyak 3 kali.

Tabel 4.9 Pencarian kata pada teks ketiga

M	1	2	3	4	5	6	7	8	9
T	N	A	T	U	R	L	I	C	H
P							I	C	H

Pada tabel 4.9 terdapat kecocokan antara *pattern* dan teks. Seluruh pencocokan karakter menggunakan algoritma *Horspool* telah selesai dan berhenti pada pencarian teks proses ketiga.

Dari 3 langkah menghasilkan 3 kali pergeseran ke kanan, yaitu :

Langkah ke 1 : menghasilkan nilai pergeseran sejauh 3 (BmBc[3]).

Langkah ke 2 : menghasilkan nilai pergeseran sejauh 3 (BmBc[3]).

Langkah ke 3 : menghasilkan nilai pergeseran sejauh 3 (BmBc[3]).

Proses pencarian kata pada algoritma *Raita*.

Pada proses pencarian diberikan teks dari database dengan *pattern* berikut

Teks = natuerlich

Pattern = ich

a. algoritma *Raita* melakukan perbandingan antara teks dengan pola yang mana pencocokan dilakukan pada karakter terkanan *pattern*. Tabel *BmBc* digunakan untuk melewati karakter ketika ketidakcocokan terjadi. Hal ini dapat dilihat pada tabel 4.10.

Tabel 4.10 Pencarian kata pada teks pertama

M	1	2	3	4	5	6	7	8	9
T	N	A	T	U	R	L	I	C	H
P	I	C	H						

Pada tabel 4.10 terdapat ketidakcocokan karakter “T” dengan “H”. Karakter “T” tidak terdapat pada tabel BmBc sehingga dilakukan pergeseran sebanyak 3 kali. Hal ini terlihat pada tabel 4.11.

b. Setelah melakukan pergeseran sebesar karakter yang ada pada tabel *BmBc* maka akan dilakukan pencocokan antara teks dan *pattern*, apabila terjadi ketidakcocokan antara karakter dalam teks dengan karakter pada *pattern* maka akan dilakukan kembali pergeseran sesuai dengan nilai karakter pada tabel BmBc. Hal ini dapat dilihat pada tabel 4.11.

Tabel 4.11 Pencarian kata pada teks kedua

M	1	2	3	4	5	6	7	8	9
T	N	A	T	U	R	L	I	C	H
P				I	C	H			

Pada tabel 4.11 terdapat ketidakcocokan karakter “L” dengan “H”. Karakter “L” tidak terdapat pada tabel BmBc sehingga dilakukan pergeseran sebanyak 3 kali. Hal ini terlihat pada tabel 4.12

Tabel 4.12 Pencarian kata pada teks ketiga

M	1	2	3	4	5	6	7	8	9
T	N	A	T	U	R	L	I	C	H
P							I	C	H

Pada tabel 4.12 terdapat kecocokan antara *pattern* dan teks. Seluruh pencocokan karakter menggunakan algoritma *Raita* telah selesai dan berhenti pada pencarian teks proses ketiga.

Dari 3 langkah menghasilkan 3 kali pergeseran ke kanan, yaitu :

Langkah ke 1 : menghasilkan nilai pergeseran sejauh 3 (BmBc[3]).

Langkah ke 2 : menghasilkan nilai pergeseran sejauh 3 (BmBc[3]).

Langkah ke 3 : menghasilkan nilai pergeseran sejauh 3 (BmBc[3]).

4.3. Hasil Pengujian

Hasil pengujian pada dari penelitian ini adalah kompleksitas dari kedua algoritma (*Horspool* dan *Raita*) dan perbandingan *running-time*.

4.3.1. Kompleksitas Algoritma Big θ

Tabel 4.13 Kompleksitas Algoritma Horspool

Code			
<code>class horspool{</code>	C	#	#.C
<code>private \$n, \$y, \$m, \$x,\$BmBc,\$matches=false;</code>	C1	1	C1
<code>public function search(\$yy, \$xx) {</code>			
<code>\$this->setText(\$yy);</code>	C2	1	C2
<code>\$this->setPattern(\$xx);</code>	C2	1	C2
<code>\$this->horspoolSearch();</code>	C2	1	C2
<code>}</code>			
<code>private function setText(\$yy){</code>			
<code>\$this->n=strlen(\$yy);</code>	C3	1	C3
<code>\$this->y=str_split(\$yy);</code>	C3	1	C3
<code>}</code>			
<code>private function setPattern(\$xx) {</code>			

<code>\$this->m=strlen(\$xx);</code>	C3	1	C3
<code>\$this->x=str_split(\$xx);</code>	C3	1	C3
<code>\$this->horspoolPreprocess();</code>	C2	1	C2
<code>}</code>			
<code>private function horspoolPreprocess(){</code>			
<code>for (\$i=0; \$i<256; \$i++)</code>	C4	256	256C4
<code> \$this->BmBc[\$i] = \$this->m;</code>	C3	256	256C3
<code>for (\$i=0; \$i<\$this->m - 1; \$i++)</code>	C4	M	C4m
<code> \$this->BmBc[ord(\$this->x[\$i])] = \$this->m-1-\$i;</code>	C3	M	C3m
<code>}</code>			
<code>private function horspoolsearch() {</code>			
<code>\$i = \$this->m-1;</code>	C3	1	C3
<code>\$k = 0;</code>	C3	1	C3
<code>while (\$i <= \$this->n-1) {</code>	C5	N	C5n
<code> while ((\$k < \$this->m) && ((\$this->y[\$i-\$k] == \$this->x[\$this->m-1-\$k])))</code>	C5	Mn	C5mn
<code> \$k++;</code>	C3	Mn	C3mn
<code> if (\$k==\$this->m) {</code>	C6	N	C6n
<code> \$this->matches = true;</code>	C3	N	C3n
<code> break; }</code>			
<code> else {</code>			
<code> \$i = \$i+\$this->BmBc[ord(\$this->y[\$i])];</code>	C3	N	C3n
<code> }</code>			
<code>}</code>			
<code>}</code>			
<code>public function report(){</code>			

<code>return \$this->matches;</code>	C2	1	C2
<code>}</code>			
<code>}</code>			

$$\begin{aligned}
 T(n) &= C1 + 5C2 + 262C3 + 256C4 + C3m + C4m + 2C3n + C5n + C6n + C5mn \\
 &\quad + C3mn \\
 &= (C1 + 5C2 + 262C3 + 256C4) + (C3 + C4)m + (2C3 + C5 + C6)n + \\
 &\quad (C3 + C5)mn \\
 &= 0 + m + n + mn \\
 &= \Theta(mn)
 \end{aligned}$$

Tabel 4.13 adalah tabel kompleksitas algoritma *Horspool*, dimana proses pencarian kompleksitasnya menggunakan bahasa php, C sebagai konstanta, # sebagai ukuran masukan, dan C. # (C kali #) adalah untuk mencari *Theoretical Running Time* (T(n)) atau kompleksitas waktu, sehingga dapat dijumlahkan hasil dari perkalian C kali #, maka diperoleh hasil Θmn .

Tabel 4.14 Kompleksitas Raita

Code			
<code>class raita{</code>	C	#	#.C
<code>private \$n, \$y, \$m, \$x,\$BmBc,\$matches=false;</code>	C1	1	C1
<code>private \$firstCh, \$secondCh, \$middleCh, \$lastCh;</code>	C1	1	C1
<code>public function search(\$yy, \$xx) {</code>			
<code>\$this->setText(\$yy);</code>	C2	1	C2
<code>\$this->setPattern(\$xx);</code>	C2	1	C2
<code>\$this->raitasearch();</code>	C2	1	C2

}			
private function setText(\$yy){			
\$this->n=strlen(\$yy);	C3	1	C3
\$this->y=str_split(\$yy);	C3	1	C3
}			
private function setPattern(\$xx){			
\$this->m=strlen(\$xx);	C3	1	C3
\$this->x=str_split(\$xx);	C3	1	C3
\$this->raitaPreprocess();	C2	1	C2
\$this->firstCh=\$this->x[0];	C3	1	C3
\$this->middleCh=\$this->x[\$this->m/2];	C3	1	C3
\$this->lastCh=\$this->x[\$this->m-1];	C3	1	C3
}			
private function raitaPreprocess(){			
for (\$i=0; \$i<256; \$i++)	C4	256	256C4
\$this->BmBc[\$i] = \$this->m;	C3	256	256C3
for (\$i=0; \$i<\$this->m - 1; \$i++)	C4	M	C4m
\$this->BmBc[ord(\$this->x[\$i])] = \$this->m-1-\$i;	C3	M	C3m
}			
private function raitasearch() {			
\$k = 0;	C3	1	C3
while (\$k <= \$this->n-\$this->m) {	C5	M	C5m
\$c = \$this->y[\$k+\$this->m-1];	C3	M	C3m
if (\$this->lastCh == \$c && \$this-> middleCh == \$this->y[\$k+\$this->m/2] && \$this->firstCh == \$this->y[\$k]) {	C6	M	C6m
\$this->matches = true;	C3	M	C3m

<code>break;</code>			
<code>} else {</code>			
<code>\$k += \$this->BmBc[ord(\$c)];</code>	C3	M	C3m
<code>}</code>			
<code>}</code>			
<code>}</code>			
<code>public function report() {</code>			
<code>return \$this->matches;</code>	C2	1	C2
<code>}</code>			
<code>}</code>			

$$\begin{aligned}
T(n) &= 2C1 + 5C2 + 264C3 + 256C4 + 4C3m + C4m + C5m + C6m \\
&= (2C1 + 5C2 + 264C3 + 256C4) + (4C3 + C4 + C5 + C6)m^1 \\
&= 0 + m^1 \\
&= \Theta(m)
\end{aligned}$$

Tabel 4.14 adalah tabel kompleksitas algoritma *Raita*, dimana proses pencarian kompleksitasnya menggunakan bahasa php, C sebagai konstanta, # sebagai ukuran masukan, dan C. # (C kali #) adalah untuk mencari *Theoretical Running Time* (T(n)) atau kompleksitas waktu, sehingga dapat dijumlahkan hasil dari perkalian C kali #, maka diperoleh hasil $\Theta(m)$ untuk seluruh proses algoritmanya.

Berdasarkan hasil kompleksitas serta keterangan dari **Tabel 4.13** dan **Tabel 4.14** didapat bahwa keseluruhan kompleksitas algoritma *Horspool* adalah $\Theta(mn)$ sedangkan hasil dari keseluruhan kompleksitas algoritma *Raita* adalah $\Theta(m)$. Dari hasil tersebut didapat bahwa kompleksitas algoritma *Raita* lebih baik jika dibandingkan dengan kompleksitas algoritma *Horspool*.

4.3.2. Running Time

Hasil pengujian dari penelitian ini adalah *running time* dari pencarian kata dan jumlah kata yang ditemukan pada algoritma *Horspool* dan algoritma

Raita yang dilakukan terhadap *string* yang berbeda. Untuk pengujian *string* Bahasa Indonesia-Jerman dijelaskan pada No. 1 sampai 5. Sedangkan pengujian *string* Bahasa Jerman-Indonesia dijelaskan pada No. 6 sampai 10. Adapun hasil pengujian dari kedua algoritma yang digunakan akan dijelaskan pada **Tabel 4.15** dan **Tabel 4.16**.

Tabel 4.15 Hasil Pengujian Horspool

No	Pola Indonesia – Jerman	<i>Running Time</i>	Total kata	Keterangan
1	Aku	0,205	4	<i>Match</i>
2	Makan	0,260	1	<i>Match</i>
3	Datang	0,212	1	<i>Match</i>
4	Sekolah	0,220	1	<i>Match</i>
5	Perpustakaan	0,240	1	<i>Match</i>
No	Pola Jerman – Indonesia	<i>Running time</i>	Total kata	Keterangan
6	Gut	0,210	5	<i>Match</i>
7	Kalt	0,212	2	<i>Match</i>
8	Schoen	0,225	1	<i>Match</i>
9	Die Auskunft	0,267	1	<i>Match</i>
10	Das Einzelwesen	0,330	1	<i>Match</i>
	Total	2,381		
	Rata – Rata	0,2381		

Berdasarkan **Tabel 4.15** hasil pengujian *Horspool* pada pola Indonesia Jerman :

1. “Aku” memiliki *running time* sebesar 0,205 dengan total kata sebesar 4 dari hasil yang diuji adalah cocok.
2. “Makan” memiliki *running time* sebesar 0,260 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
3. “Datang” memiliki *running time* sebesar 0,212 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.

4. “Sekolah” memiliki *running time* sebesar 0,220 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
5. “Perpustakaan” memiliki *running time* sebesar 0,240 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.

Berdasarkan **Tabel 4.15** hasil pengujian *Horspool* pada pola Jerman – Indonesia :

1. “Gut” memiliki *running time* sebesar 0,210 dengan total kata sebesar 5 dari hasil yang diuji adalah cocok.
2. “Kalt” memiliki *running time* sebesar 0,212 dengan total kata sebesar 2 dari hasil yang diuji adalah cocok.
3. “Schoen” memiliki *running time* sebesar 0,225 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
4. “Die Auskunft” memiliki *running time* sebesar 0,267 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
5. “Das Einzelwesen” memiliki *running time* sebesar 0,330 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.

Tabel 4.16 Hasil Pengujian Raita

No	Pola Indonesia – Jerman	<i>Running Time</i>	Total kata	Keterangan
1	Aku	0,212	4	<i>Match</i>
2	Makan	0,270	1	<i>Match</i>
3	Datang	0,205	1	<i>Match</i>
4	Sekolah	0,215	1	<i>Match</i>
5	Perpustakaan	0,235	1	<i>Match</i>
No	Pola Jerman – Indonesia	<i>Running time</i>	Total kata	Keterangan
6	Gut	0,215	5	<i>Match</i>
7	Kalt	0,242	2	<i>Match</i>
8	Schoen	0,217	1	<i>Match</i>
9	Die Auskunft	0,248	1	<i>Match</i>
10	Das Einzelwesen	0,277	1	<i>Match</i>

	Total	2,336		
	Rata – Rata	0,2336		

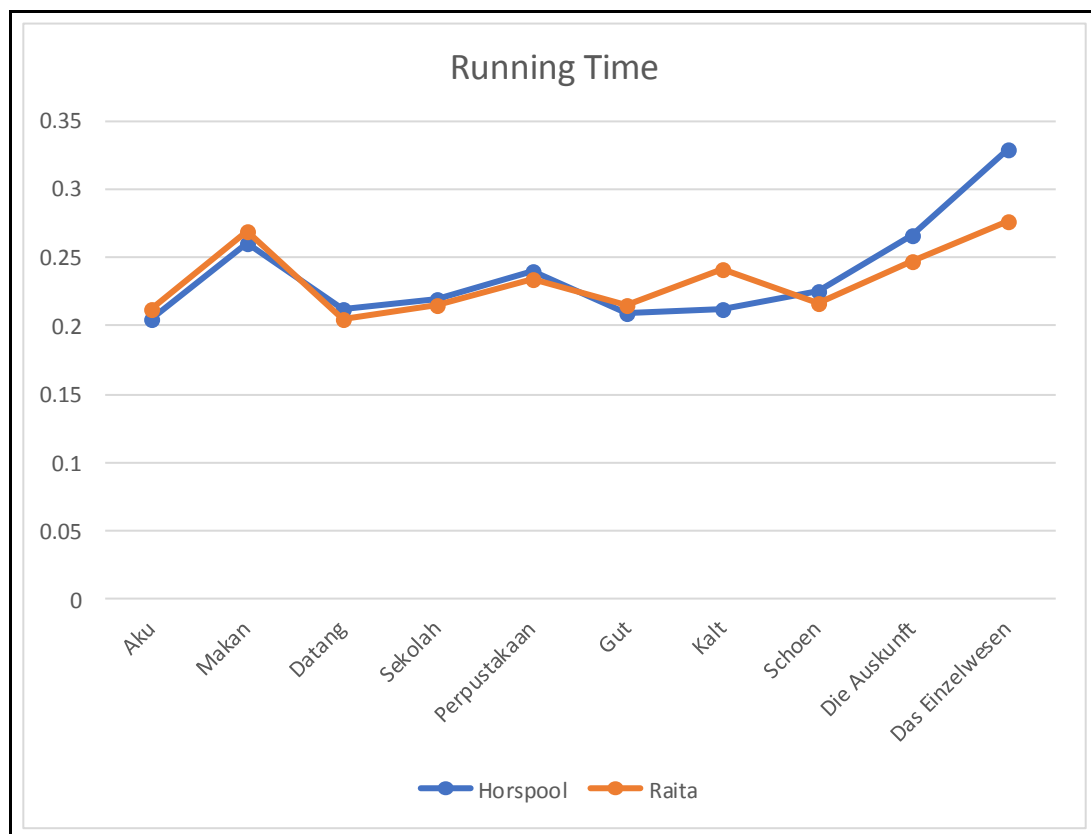
Berdasarkan **Tabel 4.16** hasil pengujian *Raita* pada pola Indonesia - Jerman :

1. “Aku” memiliki *Running time* sebesar 0,212 dengan total kata sebesar 4 dari hasil yang diuji adalah cocok.
2. “Makan” memiliki *Running time* sebesar 0,270 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
3. “Datang” memiliki *Running time* sebesar 0,205 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
4. “Sekolah” memiliki *Running time* sebesar 0,215 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
5. “Perpustakaan” memiliki *Running time* sebesar 0,235 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.

Berdasarkan **Tabel 4.16** hasil pengujian *Raita* pada pola Jerman - Indonesia:

1. “Gut” memiliki *Running time* sebesar 0,215 dengan total kata sebesar 5 dari hasil yang diuji adalah cocok.
2. “Kalt” memiliki *Running time* sebesar 0,242 dengan total kata sebesar 2 dari hasil yang diuji adalah cocok.
3. “Schoen” memiliki *Running time* sebesar 0,217 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
4. “Die Auskunft” memiliki *Running time* sebesar 0,248 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.
5. “Das Einzelwesen” memiliki *Running time* sebesar 0,277 dengan total kata sebesar 1 dari hasil yang diuji adalah cocok.

Setelah mendapatkan Hasil Pengujian dari **Tabel 4.15** dan **Tabel 4.16** maka dibuat grafik perbandingan hasil pengujian dari kedua Algoritma tersebut. Grafik dapat dilihat pada **Gambar 4.6**.



Gambar 4.6 Perbandingan Hasil *Running Time* Algoritma *Horspool* dan Algoritma *Raita*

Dari grafik diatas dapat dijelaskan bahwa algoritma *Horspool* mendapatkan Hasil *Running Time* yang relatif tinggi dibandingkan dengan Algoritma *Raita*. Artinya bahwa algoritma *Horspool* lebih lama untuk pencocokan kata dibandingkan dengan Algoritma *Raita* namun, perbedaan ini tidak terlalu signifikan dikarenakan hanya berbeda sangat sedikit.

BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil studi literatur, analisis, perancangan, implementasi dan pengujian sistem ini, Maka kesimpulan yang didapat adalah sebagai berikut:

1. Algoritma *Horspool* melakukan pergeseran karakter sebanyak satu karakter paling kanan apabila cocok melakukan pergeseran karakter berikutnya ke kiri. Sedangkan Algoritma *Raita* melakukan pergeseran karakter sebanyak satu karakter paling kanan apabila cocok melakukan pergeseran ke awal karakter dan tengah karakter. Hal ini menunjukkan bahwa kedua Algoritma tersebut memiliki persamaan dalam pencarian karakter pada pencocokan karakter paling kanan dari teks. Rata – rata hasil *running time* Algoritma *Horspool* untuk pencarian kata adalah 0,2381 ms. Pada Algoritma *Raita* rata – rata hasil *running time* 0,2336 ms.
2. Hasil Kompleksitas algoritma *Horspool* adalah $T(n) = \Theta(mn)$. Sedangkan Hasil kompleksitas algoritma *Raita* adalah $T(n) = \Theta(m)$. Maka pencarian dengan menggunakan algoritma *Raita* lebih cepat dibandingkan dengan algoritma *Horspool*. Walaupun terkadang kedua algoritma ini tidak terlihat jauh berbeda saat diimplementasikan.
3. Pada *running time* dan kompleksitas dari kedua algoritma menghasilkan bahwa algoritma *Raita* lebih efektif dari algoritma *Horspool* pada pencarian kata.

5.2. Saran

Saran yang dapat diberikan pada penulis untuk pengembangan dan perbaikan sistem lebih lanjut adalah:

1. Untuk penelitian selanjutnya diharapkan aplikasi ini menyediakan menu pilihan algoritma berbagai pencocokan *string* seperti *Quick Search*, *Zhu-Takaoka*, *Two Way*.
2. Sistem ini sebaiknya ditambahkan *Sound* untuk pelafalan kata bahasa Jerman agar pengguna mampu mengucapkan kata yang baik dan benar.

DAFTAR PUSTAKA

- Al-khamaiseh, K. & Alshagarin, S. 2014. A survey of string matching algorithms. *Journal Of Engineering Research And Applications* Vol. 4: 144-156
- Baeza-Yates, R.A. & Regnier, M. 1992. Average running time of the boyer-moore-horspool algorithm. *Journal Theoretical Computer Science* Vol. 92, No. 1: 19-31.
- Charras, C. & Lecroq, T. 1997. *Handbook of Exact String Matching Algorithms*. Oxford University Press: United Kingdom.
- Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C. 1994. *Introduction to Algorithms*. McGraw-Hill Book Company: North America.
- Effendi, D., Hartono, T. & Kurnaedi, A. 2013. Penerapan string matching menggunakan algoritma Boyer-Moore pada translator bahasa Pascal ke C. *Majalah Ilmiah Unikom* Vol. 11, No. 2: 262-271.
- Horspool, R.N. 1980. Practical fast searching in strings. *Journal Software Practice and Experience* Vol. 10: 501-506.
- Ibrahim A.M.A. & Mustafa M.E. 2015. Comparison criteria between matching algorithms teks application on (horspool's and brute force algorithms). *Journal of advance Computer Science & Technology*, 4(1): 175-179.
- Knuth D.E., Morris (Jr) J.H. & Pratt V.R. 1977. Fast Pattern Matching In Strings. *SIAM Journal on Computing* Vol. 6, No.1: 323-350.
- Nasution, A.M.K.D. Implementasi Algoritma *Horspool* Dalam Pembuatan Kamus Istilah Psikologi Pada *Platform* Android. Skripsi. Universitas Sumatera Utara.
- Nasution, N. Implementasi Algoritma *Raita* Dalam Kamus Bahasa Indonesia-Mandailing Berbasis Android. Skripsi. Universitas Sumatera Utara.
- Nebel M.E. 2006. Fast String Matching By Using Probabilities: On An Optimal Mismatch Variant Of Horspool's Algorithm. *Theoretical Computer Science*, 359 : 329-343
- Rayahuningsih, P.A. 2016. Analisis Perbandingan Kompleksitas Algoritma Pengurutan Nilai (*Sorting*). *Jurnal Evolusi* Vol. 4, No 2 : 2338-8161.
- Santoso, I., Marzuki A., Haryati I. 2011. Pemanfaatan Teks Sastra Dalam Pembelajaran Bahasa Jerman Di Sekolah Menengah Atas. *Jurnal Allemania* Vol. 1, No 1 : 2088-7582.
- Sheik, S.S., Aggarwal, S.K., Poddar, A., Balakhrisnan, N. & Sekar, K. 2004. A fast pattern matching algorithm. *Journal of Chemical Information and Computer Science* Vol. 44, No. 4: 1251-1256.
- Singh, R. & Verma, H.N. 2011. A fast string matching algorithm. *International Journal of Computer Technology and Applications* Vol. 2, No. 6: 1877-1883.
- Syara, Z.M. Implementasi Dan Perbandingan Algoritma Smith Dan Algoritma Raita Pada Pencarian Kata. Skripsi. Universitas Sumatera Utara.

- Syaroni, M. & Munir, R. 2005. Pencocokan *String* Berdasarkan Kemiripan Ucapan (*Phonetic String Matching*) dalam Bahasa Inggris. (Online)
informatika.stei.itb.ac.id/~rinaldi.munir/Penelitian/Phonetic_string_Matching.pdf (20 desember 2016).
- Tambun, E.D. 2010. Perbandingan penggunaan algoritma boyer-moore dan algoritma horspool pada pencarian string dalam bahasa medis. Skripsi. Institut Teknologi Bandung.

LISTING PROGRAM

1. Indeks.php

```

<?php include 'head.php'; ?>
<?php include'horspool.php' ?>
<?php include'raita.php'?>
<?php
error_reporting(0);
//deklarasi untuk penyimpanan semua data
$dataH = array("");
$dataR = array("");
$judul = array("Bahasa Indonesia","Bahasa Jerman");
$dictionary = array("");
//deklarasi untuk running time
$Htime = 0;
$Rtime      = 0;
//deklarasi untuk jumlah data
$countH=0;
$countR=0;
if(isset($_POST['cari'])){
    if($_POST['indojerman'] == "indojerman"){

        $sql = "select * from indojerman order by indo
asc";

        $sql = mysqli_query($conn, $sql);
        $i = 0;

        $start = microtime(true);

        while ($row = mysqli_fetch_array($sql)){
            $dictionary[$row[1]] = $row[2];

            $horspool = new horspool;
            $horspool->search(strtolower($row[1]),strtolower($_POST
['stringcari']));
            if($horspool->report() == 1){
                $dataH[$i] = $row[1];
                $i++;
            }
        }
        $countH = count($dataH);
        $Htime = microtime(true)-$start;

        //pencarian untuk raita
        $sql = "select * from indojerman order by jerman
asc";

        $sql = mysqli_query($conn, $sql);
        $i=0;
        //memulai hitungan waktu untuk raita
        $start = microtime(true);
        while ($row = mysqli_fetch_array($sql)){
            $dictionary[$row[1]] = $row[2];
            $raita = new raita;

```

```

        $raita-
>search(strtolower($row[1]),strtolower($_POST['stringcari']));
        if($raita->report() == 1){
            $dataR[$i] = $row[1];
            $i++;
        }
    }

$countR = count($dataR);
//menyimpan hitungan waktu untuk raita
$Rtime = microtime(true)-$start;

} else if($_POST['indojerman'] == "jermanindo"){
$judul[0] = "Bahasa Jerman";
$judul[1] = "Bahasa Indonesia";
//pencarian untuk horspool
$sql = "select * from indojerman";
$sql = mysqli_query($conn, $sql);
$i = 0;
//memulai hitungan waktu untuk horspool
$start = microtime(true);
while ($row = mysqli_fetch_array($sql)){
    $dictionary[$row[2]] = $row[1];
    //untuk memulai pencarian horspool
    $horspool = new horspool;
    $horspool-
    >search(strtolower($row[2]),strtolower($_POST['stringcari']));
    if($horspool->report() == 1){
        $dataH[$i] = $row[2];
        $i++;
    }
}
$countH = count($dataH);
//menyimpan hitungan waktu untuk horspool
$Htime = microtime(true)-$start;

//pencarian untuk raita
$sql = "select * from indojerman";
$sql = mysqli_query($conn, $sql);
$i=0;
//memulai hitungan waktu untuk raita
$start = microtime(true);
while ($row = mysqli_fetch_array($sql)){
    $dictionary[$row[2]] = $row[1];
    //untuk memulai pencarian raita
    $raita = new raita;
    $raita-
>search(strtolower($row[2]),strtolower($_POST['stringcari']));
    if($raita->report() == 1){
        $dataR[$i] = $row[2];
        $i++;
    }
}
$countR = count($dataR);
//menyimpan hitungan waktu untuk raita
$Rtime = microtime(true)-$start;

```



```

    }
}

```

2. Horspool.php

```

<?php
class horspool{
    private $n, $y, $m, $x,$BmBc,$matches=false; //

    public function search($yy, $xx) {
        $this->matches = false;
        $this->setText($yy);
        $this->setPattern($xx);
        $this->horspoolSearch();
    }

    //Set text yang ingin di uji

    private function setText($yy)
    {
        //mengambil panjang text uji
        $this->n=strlen($yy);
        //mengambil setiap karakter text uji
        $this->y=str_split($yy);
    }
    private function setPattern($xx)
    {
        //mengambil panjang text input user
        $this->m=strlen($xx);
        //mengambil setiap karakter text input user
        $this->x=str_split($xx);
        //memanggil fungsi preprocess
        $this->horspoolPreprocess();
    }

    private function horspoolPreprocess(){
        //memasukan panjang dari text input user dari 0 sampai
        //256 ke dalam array
        //sebagai bmbc
        for ($i=0; $i<256; $i++){
            $this->BmBc[$i] = $this->m;
        }
        //untuk mengisi panjang text yang diinput user dikurangi
1      for ($i=0; $i<$this->m - 1; $i++){
            //fungsi ord mengambil nilai tabel ascii, jadi
            //angka m-1-i disisipkan kedalam 0-256
            //sesuai dengan tabel ascii
            $this->BmBc[ord($this->x[$i])] = $this->m-1-$i;
        }
    }

    private function horspoolsearch() {
        $i = $this->m-1; // mengambil nilai i
        $k = 0; // kasih nilai
        while ($i <= $this->n-1) // melakukan perulangan ketika
        kata yg dicari lebih pendek dari database maka dilakukan
        proses pencarian
    }
}

```

```

        {
            while (($k < $this->m) && (($this->y[$i-$k] == $this->x[$this->m-1-$k])))//jika huruf pada indeks pattern sama
            dengan indeks huruf pada text(proses pencocokan)
            {
                $k++;
            }

            if ($k==$this->m) //apabila jumlah k sama dengan jumlah
            pattern
            {
                $this->matches = true;
                break;
            }
            else { // jika ada kata yg tidak sama dibandingkan
                $k=0;
                //pergeseran horspool
                $i = $i+$this->BmBc[ord($this->y[$i])];
            }
        }
    }

    public function report(){
        return $this->matches;
    }
}
?>

```

3. raita.php

```

<?php
class raita{
    private $n, $y, $yy, $m, $x, $xx,$BmBc,$matches=false;
    private $firstCh, $secondCh, $middleCh, $lastCh;

    public function search($yy, $xx) {
        $this->matches = false;
        $this->setText($yy);
        $this->setPattern($xx);
        $this->raitasearch();
    }
    private function setText($yy)
    {
        //mengambil panjang text uji
        $this->n=strlen($yy);
        //mengambil setiap karakter text uji
        $this->y=str_split($yy);
    }
    private function setPattern($xx)
    {
        //mengambil panjang text input user
        $this->m=strlen($xx);
        //mengambil setiap karakter text input user
        $this->x=str_split($xx);
        //memanggil fungsi preprocess
        $this->raitapreprocess();
    }
}

```

```

        $this->firstCh=$this->x[0];
        //$this->secondCh=$this->x+1;
        $this->middleCh=$this->x[$this->m/2];
        $this->lastCh=$this->x[$this->m-1];
    }

    private function raitaPreprocess(){
        //memasukan panjang dari text input user dari 0 sampai
        256 ke dalam array
        //sebagai bmbc
        for ($i=0; $i<256; $i++){
            $this->BmBc[$i] = $this->m;
        }
        //fungsi ord mengambil nilai tabel ascii, jadi angka m-1-
        i disisipkan kedalam 0-256
        //sesuai dengan tabel ascii
        for ($i=0; $i<$this->m - 1; $i++){
            $this->BmBc[ord($this->x[$i])] = $this->m-1-$i;
        }
    }

    private function raitasearch() {
        $k = 0;
        while ($k <= $this->n-$this->m)
        {
            $c = $this->y[$k+$this->m-1];
            if ($this->lastCh == $c && $this->firstCh == $this->
                >y[$k] && $this->middleCh == $this->y[$k+$this->m/2] &&
                strcmp(substr($this->yy,$k,$this->m),$this->xx) == 0)
            $this->matches = true;
            break;
        } else {
            $k += $this->BmBc[ord($c)];
        }
    }
}

public function report(){
    return $this->matches;
}
}
?>

```

Curriculum Vitae

Data Pribadi

Nama : Ryan Ridho Valba
Tempat, Tanggal lahir : Tarutung, 17 Maret
1995
Agama : Islam
Alamat rumah : Jl. Sei Wampu Baru No.4C
Nomor telepon : 082272302366
Email : ryanridhovalba@gmail.com



Riwayat Pendidikan

2012 – 2017 : S-1 Ilmu Komputer Universitas Sumatera
Utara, Medan
2009 – 2012 : SMA Negeri 4 Medan
2006 – 2009 : SMP Shafiyatul Amaliyah
2000 – 2006 : SD Negeri 060884

Pengalaman

Periode 30 Juli s/d 16 Agustus 2015 : Praktik Kerja Lapangan (PKL) di Dinas
Kependudukan dan Catatan Sipil

Keahlian

- bahasa pemrograman : php
- DBMS : MySQL