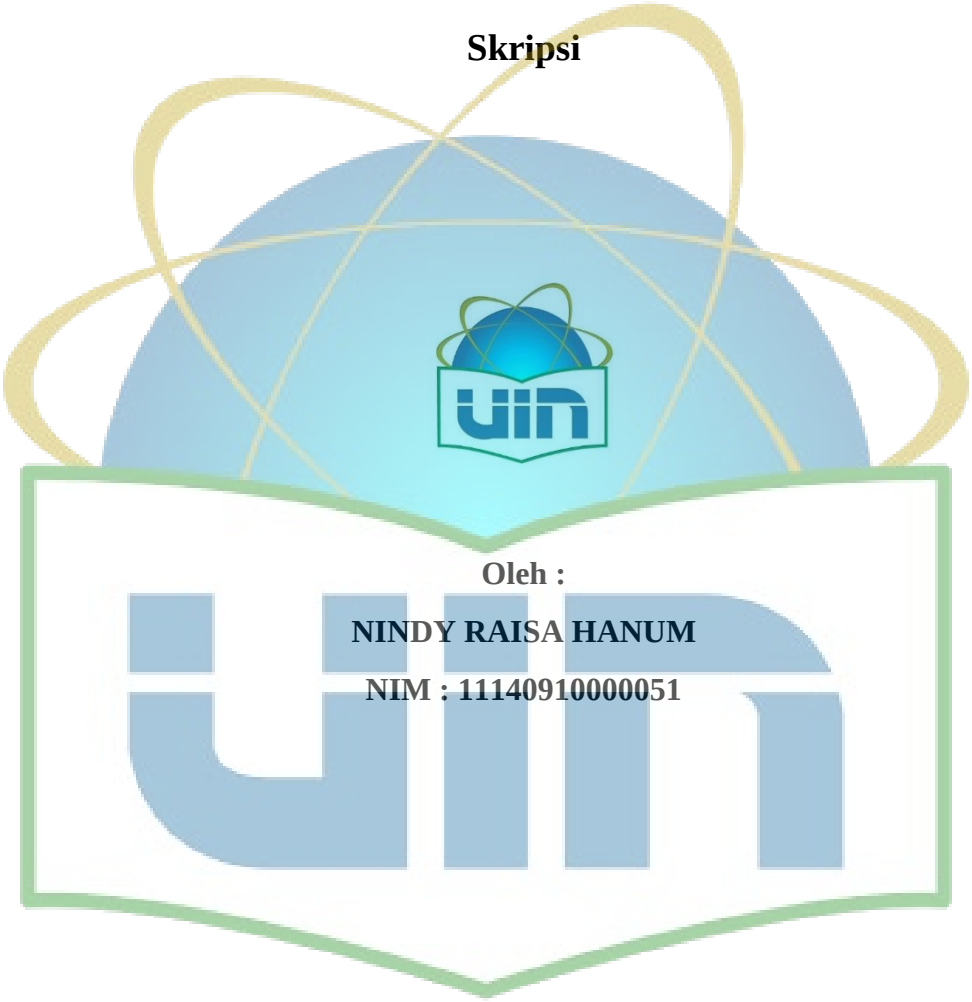
ANALISIS PERBANDINGAN KINERJA ALGORITMA BOYER MOORE, HORSPOOL, DAN ZHU TAKAOKA PADA REPOSITORI HADITS BUKHORI TERJEMAHAN BAHASA INDONESIA

Skripsi



Oleh :

NINDY RAISA HANUM

NIM : 11140910000051

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH

JAKARTA

2018 M / 1440 H

ANALISIS PERBANDINGAN KINERJA ALGORITMA BOYER MOORE, HORSPOOL, DAN ZHU TAKAOKA PADA REPOSITORI HADITS BUKHORI TERJEMAHAN BAHASA INDONESIA

Skripsi

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh

Gelar Sarjana Komputer (S.Kom)

Oleh :

NINDY RAISA HANUM

NIM : 11140910000051

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH

JAKARTA

2018 M / 1440 H

LEMBAR PERSETUJUAN
ANALISIS PERBANDINGAN KINERJA
ALGORITMA BOYER MOORE, HORSPOOL, DAN ZHU TAKAOKA
PADA REPOSITORI HADITS BUKHORI
TERJEMAHAN BAHASA INDONESIA

Skripsi

Sebagai Salah Satu Syarat untuk

Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:

NINDY RAISA HANUM

11140910000051

Menyetujui,

Pembimbing I	Pembimbing II
Dr. Imam Marzuki Shofi, MT	Siti Ummi Masruroh, M.Sc
NIP. 19720205 200801 1 010	NIP. 19820823 201101 2 013

Mengetahui,

Ketua Program Studi Teknik Informatika

Arini, MT.

NIP. 19760131 200901 2 001

LEMBAR PENGESAHAN

Skripsi Berjudul Analisis Perbandingan Kinerja Algoritma Boyer Moore, Horspool, dan Zhu Takaoka pada Repositori Hadits Bukhori Terjemahan Bahasa Indonesia telah diujikan dan dinyatakan lulus dalam sidang munaqasah Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta pada Oktober 2018. Skripsi ini telah diterima sebagai salah satu syarat memperoleh gelar Sarjana Komputer (S.Kom) pada Program Studi Teknik Informatika.

Jakarta, Oktober 2018

Tim Penguji,

Penguji I	Penguji II
Nashrul Hakiem, Ph.D	Victor Amrizal, M.Kom
NIP. 19710608 200501 1 005	NIP. 19740624 200710 1 001

Tim Pembimbing,

Pembimbing I	Pembimbing II
Dr. Imam Marzuki Shofi, MT	Siti Ummi Masruroh, M.Sc
NIP. 19720205 200801 1 010	NIP. 19820823 201101 2 013

Mengetahui,

Dekan Fakultas Sains dan Teknologi	Ketua Program Studi Teknik Informatika
Dr. Agus Salim, M.Si	Arini, MT
NIP. 19720816 199903 1 003	NIP. 19760131 200901 2 001

PERNYATAAN ORISINALITAS

Dengan ini saya menyatakan bahwa:

1. Skripsi ini merupakan hasil karya asli saya yang diajukan untuk memenuhi salah satu persyaratan memperoleh gelar strata 1 di UIN Syarif Hidayatullah Jakarta.
2. Semua sumber yang tercantum dalam penulisan ini telah saya cantumkan sesuai dengan ketentuan yang berlaku di UIN Syarif Hidayatullah Jakarta.
3. Apabila di kemudian hari terbukti karya ini bukan hasil karya asli saya, maka saya bersedia menerima sanksi yang telah ditetapkan di UIN Syarif Hidayatullah Jakarta.

Ciputat, Oktober 2018

Nindy Raisa Hanum

PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI

Sebagai civitas akademik UIN Syarif Hidayatullah Jakarta, saya yang bertanda tangan di bawah ini:

Nama	: Nindy Raisa Hanum
NIM	: 11140910000051
Program Studi	: Teknik Informatika
Fakultas	: Sains dan Teknologi
Jenis Karya	: Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Islam Negeri Syarif Hidayatullah Jakarta Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul:

Analisis Perbandingan Kinerja
Algoritma Boyer Moore, Horspool, dan Zhu Takaoka
pada Repositori Hadits Bukhori Terjemahan Bahasa Indonesia

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Islam Negeri Syarif Hidayatullah Jakarta berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan pemilik Hak Cipta. Demikian pernyataan ini saya buat dengan sebenarnya.

Ciputat, Oktober 2018

Nindy Raisa Hanum

Penulis : Nindy Raisa Hanum
Program Studi : Teknik Informatika
Judul : Analisis Perbandingan Kinerja Algoritma Boyer Moore, Horspool, dan Zhu Takaoka pada Repositori Hadits Bukhori Terjemahan Bahasa Indonesia

ABSTRAK

Seiring dengan perkembangan zaman, kemajuan teknologi semakin meningkat, terutama teknologi informasi. Hal ini mendorong munculnya inovasi baru dalam penyajian informasi berupa digital. *String matching* menjadi kebutuhan dalam penyajian informasi, khususnya dalam pencarian teks. Oleh karena itu, dibutuhkan Algoritma *string matching* yang digunakan untuk mencari satu atau lebih *string* dalam kumpulan *string* (teks). Penelitian ini dilakukan untuk membandingkan kinerja beberapa Algoritma *String Matching*, seperti Algoritma Boyer Moore, Horspool, dan Zhu Takaoka dalam proses pencarian *string*. Untuk menilai kinerja algoritma, penelitian ini menggunakan parameter *runtime* dan *memory consumption*. Kedua parameter tersebut harus memiliki nilai sekecil mungkin untuk mendapatkan nilai kinerja terbaik. Parameter *input* dalam penelitian ini adalah beberapa karakter dan sejumlah kata yang dibagi dalam beberapa skenario dan *output* dari penelitian ini adalah *runtime*, *memory consumption*, dan *accuracy*. Kedua parameter tersebut dikalkulasikan dengan menggunakan Metode Perbandingan Eksponensial (MDE) untuk mengetahui hasil terbaik. Hasil akhir dari penelitian ini menunjukkan bahwa Algoritma Boyer Moore di posisi pertama, kemudian Zhu Takaoka, dan Algoritma Horspool diposisi terakhir.

Kata kunci : Analisis Algoritma, Algoritma Boyer Moore, Algoritma Horspool, Algoritma Zhu Takaoka, *Runtime*, *Memory Consumption*, Metode Simulasi

Jumlah Pustaka : 12 Buku + 17 Jurnal + 1 *Website*
Jumlah Halaman : VI Bab + xv Halaman + 96 Halaman

Author : Nindy Raisa Hanum
Study Program : Informatics
Title : Performance Analysis of Boyer Moore, Horspool, and Zhu Takaoka Algorithm on Indonesian Translation Repository of Bukhori Hadith

ABSTRACT

Increasing of information technology, encourages of new innovations in the presentation of digital information. String matching is a necessity in presenting information, especially for searching of the text. Therefore, a string matching algorithm is needed which is used to search for one or more strings in a set of strings (text). This research was conducted to compare the performance of three string matching algorithms, they are Boyer Moore, Horspool, and Zhu Takaoka algorithms in the string search process which implemented in the Indonesian Translation of Bukhori Hadith. To assess the performance of the algorithm, this research using runtime and memory consumption as parameters. The input parameters in this research are several characters and words divided into several scenarios and output from this research are runtime, memory consumption and accuracy. Both parameters are calculated using the Exponential Comparison Method (MDE) to find out the best results. The final result of this research shows that the Boyer Moore algorithm shows the best value, followed by the Zhu Takaoka Algorithm and then Horspool Algorithm.

Keyword : Performance Analysis, Boyer Moore Algorithm Horspool Algorithm, Zhu Takaoka Algorithm, Runtime Memory Consumption, Simulation Method

Number of Reference : 12 Books + 17 Journals + 1 Website
Number of Page : VI Chapters + xv Pages + 96 Pages

KATA PENGANTAR

Alhamdulillah segala puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan Rahmat dan Hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini. Tak lupa shalawat serta salam kepada Nabi Muhammad SAW, beserta keluarga, para sahabat, dan para pengikutnya dari awal hingga akhir zaman.

Skripsi berjudul "Analisis Perbandingan Kinerja Algoritma Boyer Moore, Horspool, dan Zhu Takaoka pada Repositori Hadits Bukhori Terjemahan Bahasa Indonesia" disusun untuk memenuhi persyaratan guna mendapatkan gelar Sarjana Komputer (S.Kom) pada Program Studi Teknik Informatika di Universitas Islam Negeri Syarif Hidayatullah Jakarta.

Selama proses penyusunan skripsi ini, penulis mendapat banyak bimbingan, bantuan, masukan, dukungan, serta motivasi yang sangat bermanfaat dari berbagai pihak. Oleh karena itu melalui kata pengantar ini penulis ingin menyampaikan terima kasih banyak kepada:

1. Bapak Dr. Agus Salim, M.Si selaku Dekan Fakultas Sains dan Teknologi.
2. Ibu Arini, MT selaku Ketua Program studi Teknik Informatika dan Bapak Feri Fahrianto, M.Sc selaku Sekretaris Program Studi Teknik Informatika.
3. Bapak Dr. Imam Marzuki Shofi, MT selaku Dosen Pembimbing I dan Ibu Siti Ummi Masruroh, M.Sc selaku Dosen Pembimbing II yang telah memberikan banyak perhatian, mendukung dan meluangkan waktu untuk membimbing, memotivasi, memberikan arahan serta saran yang sangat berguna bagi penulis.
4. Seluruh Dosen dan Staf Karyawan Fakultas Sains dan Teknologi, khususnya Program Studi Teknik Informatika yang telah memberikan ilmu, dukungan dan bantuan selama masa perkuliahan.

5. Kedua orang tua penulis, Ayah Mawardi dan Ibu Intiham yang tidak pernah berhenti mendoakan, mendukung, memberikan kasih sayang dan memotivasi penulis untuk menjadi orang yang sukses dan bermanfaat sehingga penulis dapat menyelesaikan skripsi ini.
6. Adik penulis Indri Raisa Hanum yang selalu menghibur dan memberikan semangat selama masa pengerjaan skripsi ini.
7. Sahabat-sahabat seperjuangan, Vanya Kalriska, Chusnul Yunita, Yunita Riska, Annisa Rezka Alhamra, yang selalu mendukung dan mendorong penulis untuk menjadi lebih baik. Terima kasih telah menemani penulis selama empat tahun masa perkuliahan.
8. Sahabat Kosan (Bu Muslim) Adiba Zahrotul Wilda, Afifah, Asyifa Darti, dan Fani Hayatunnisa yang selalu membantu, menemani, dan memberikan semangat selama masa pengerjaan skripsi ini.
9. Teman-teman DPH HIMTI 2018, Hafsah Mawarni, Tasya Nabilah Putri, Yulianti, Alfat Nursyahban, Amin Rois, Erry Aditya Saputra, Sigit Widodo. Terima kasih atas dukungan, do'a, dan semangatnya.
10. Seluruh teman-teman Teknik Informatika angkatan 2014, khususnya kelas TI-C tercinta. Terima kasih atas kebersamaan, kenangan, ilmu dan pengalaman selama masa kuliah.
11. Seluruh pihak yang tidak dapat disebutkan satu persatu baik secara langsung maupun tidak langsung telah membantu penulis menyelesaikan skripsi ini.

Penulis berharap semoga skripsi ini dapat bermanfaat bagi para pembaca. Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan jauh dari kata sempurna, oleh karena itu penulis menerima kritik maupun saran yang membangun untuk pengembangan penelitian yang lebih baik.

Ciputat, Oktober 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Setiap agama memiliki pedoman masing-masing dalam memberikan petunjuk kepada penganutnya. Sebagaimana hal nya dengan agama Islam, Al-Qur'an adalah petunjuk untuk menjalankan kehidupan di dunia maupun di akhirat. Selain Al-Qur'an, Islam juga memiliki Hadits sebagai sumber hukum yang memiliki kedudukan kedua dalam sumber agama Islam setelah Al-Qur'an. Oleh sebab itu, Hadits wajib diimani oleh umat Islam, sebab hadits merupakan perkataan dan perbuatan Nabi Muhammad yang dapat dijadikan sebagai landasan. Sesuai dengan firman Allah SWT yang berarti *"... dan apa-apa yang diberikan Rasul kepadamu maka terimalah ia. Dan apa-apa yang dilarangnya, maka tinggalkanlah*." (Q.S Al-Asyr : 7). Dalam ayat lain Allah SWT berfirman "*Barang siapa mentaati Rosul (Muhammad), maka ia telah mentaati Allah SWT, dan barang siapa yang berpaling darinya, maka (ketahuilah) Kami tidak mengutus (Muhammad) untuk menjadi pemelihara mereka*." (Q.S An-Nisa : 80).

Seiring dengan perkembangan zaman, kemajuan teknologi semakin meningkat, terutama teknologi informasi, sehingga mendorong munculnya inovasi baru dalam penyajian informasi berupa digital agar dapat dipelajari dengan mudah, salah satu inovasi tersebut adalah repositori. Repositori adalah ruang fisik (bangunan, ruangan, area) yang digunakan untuk cadangan penyimpanan permanen atau penyimpanan sementara (Aulia, Khairani, & Hakiem, 2017).

Selain dengan menciptakan inovasi dalam penyajian informasi, *string matching* (pencocokan string) menjadi kebutuhan dalam pemrosesan informasi, khusunya dalam pencarian teks. Sehingga dengan menggunakan satu kata kunci atau lebih dari isi hadits, dapat dengan cepat memperoleh informasi mengenai hadits tersebut. Oleh karena itu, dibutuhkan Algoritma *string matching* yang digunakan untuk mencari satu atau lebih *string* dalam kumpulan string (teks). Algoritma adalah susunan langkah penyelesaian suatu masalah secara sistematika dan logis (Sitorus, 2015). Algoritma yang baik adalah suatu langkah-langkah yang

menghasilkan *output* yang efektif dalam waktu yang relatif singkat dan penggunaan memori yang relatif sedikit (efisien) dengan langkah-langkah yang berhingga dan prosedurnya berakhir baik dalam keadaan diperoleh suatu solusi ataupun tidak ada solusinya (Kurniadi, 2013). Algoritma untuk pencarian *string* sudah semakin berkembang dari hari ke hari. Algoritma pencarian *string* dianggap memiliki hasil paling baik dalam praktiknya, yaitu algoritma yang bergerak mencocokkan *string* dari arah kanan ke kiri. Algoritma Boyer-Moore merupakan salah satu contoh algoritma yang menggunakan arah dari kanan ke kiri (Ardi, Andreswari, & Setiawan, 2017).

Pada jurnal yang berjudul Implementasi Algoritma Zhu-Takaoka pada Aplikasi Kamus Istilah Musik Berbasis Android menjelaskan bahwa Algoritma Zhu-Takaoka merupakan pengembangan dari Algoritma Boyer-Moore (Togatorop, Aan, & Coastera 2017). Selain itu, pada jurnal yang berjudul Perbandingan Penggunaan Algoritma Boyer-Moore dan Algoritma Horspool pada Pencarian *String* dalam Bahasa Medis menerangkan bahwa Algoritma Horspool merupakan penyederhaaan dari Algoritma Boyer-Moore, karena hasil dari seorang peneliti yang bernama R. Nigel Horspool mengemukakan gagasan tambahanya terhadap Algoritma Boyer-Moore (Tambun, 2011).

Ada lima penelitian yang penulis jadikan sebagai penelitian sejenis pada penelitian ini. Pertama, penelitian yang dilakukan oleh (Geraldy, Sediyono, & Beeh, 2016), mereka melakukan perbandingan tiga Algoritma *String Matching*, yaitu Algoritma Brute Force, Algoritma Knuth Morris Pratt, dan Algoritma Boyer Moore dengan waktu pemrosesan sebagai parameternya. Selanjutnya, penelitian kedua, yaitu penelitian yang dilakukan oleh (Fau, Mesran, & Ginting, 2017) membandingkan Algoritma Boyer Moore dan Algoritma Knuth Morris Pratt (KMP) dengan menggunakan metode perbandingan eksponensial (MPE) dan parameter pembandingnya adalah jumlah memori dan besarnya waktu yang dibutuhkan dari setiap proses pencocokan. Penelitian ketiga, yaitu penelitian yang dilakukan oleh (Adhi & Khrisnandi, 2017), mereka melakukan analisis perbandingan Algoritma Horspool dan Zhu-Takaoka dengan mengukur performa waktu dalam melakukan pencarian *pattern* dalam suatu *text* file. Penelitian keempat

yang dilakukan oleh (Aulia, 2017) yaitu Pengembangan Repositori Hadits Terjemahan Bahasa Indonesia (Studi Kasus: Hadits Bukhori). Penelitian Kelima yang dilakukan oleh (Handrizal, Budiman, & Ardani, 2017) mereka melakukan analisis perbandingan Algoritma Zhu Takaoka dan Algoritma Knuth Morris Pratt yang diimplementasikan pada kamus berbasis android dengan waktu pemrosesan sebagai parameternya.

Dalam penelitian-penelitian yang dilakukan oleh peneliti sebelumnya, dapat diketahui bahwa setiap Algoritma *string matching* memiliki keunggulan dan kekurangan masing-masing, baik dalam konsep pencarian *string*, waktu dan memori yang dibutuhkan untuk pemrosesannya. Dengan melakukan analisis perbandingan performansi maka dapat diketahui kecepatan dalam waktu proses pencarian yang digunakan dan memori yang terpakai, serta ketepatan kata yang dicari dalam satu kali proses dari masing-masing algoritma tersebut.

Dari kelima penelitian sejenis yang telah penulis paparkan sebelumnya, penulis membuat suatu penelitian yang membahas tentang kinerja Algoritma Boyer Moore dan pengembangan dari algoritma tersebut yaitu Algoritma Hoorspol dan Algoritma Zhu Takaoka pada repositori Hadits Bukhori terjemahan Bahasa Indonesia. Setelah itu, penulis akan membandingkan performansi ketiga algoritma tersebut saat melakukan proses pencarian kata berdasarkan kecepatan waktu proses (*runtime*) dan jumlah memori yang digunakan (*memory consumption*) sebagai parameternya. Selain itu, penulis akan menyajikan ketepatan kata yang dicari pada pemrosesan masing-masing algoritma. Oleh sebab itu, penulis memberi judul penelitian ini "Analisis Perbandingan Kinerja Algoritma Boyer Moore, Horspool, dan Zhu Takaoka pada Repositori Hadits Bukhori Terjemahan Bahasa Indonesia".

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka penulis merumuskan masalah penelitian sebagai berikut : Bagaimana kinerja Algoritma Boyer Moore, Horspool, dan Zhu Takaoka dalam proses pencarian *string* berdasarkan parameter *runtime* dan *memory consumption* pada repositori Hadits Bukhori terjemahan Bahasa Indonesia?

1.3. Tujuan Penelitian

Berdasarkan permasalahan yang telah disebutkan di atas, maka tujuan penelitian ini adalah untuk mengetahui hasil perbandingkan kinerja Algoritma Boyer Moore, Horspool, dan Zhu Takoaka pada repositori Hadits Bukhori terjemahan Bahasa Indonesia dengan waktu pemrosesan pencarian kata (*runtime*) dan memori yang digunakan (*memory consumption*) sebagai parameternya.

1.4. Manfaat Penelitian

Setelah melaksanakan skripsi ini diharapkan dapat memberikan manfaat antara lain sebagai berikut:

1.4.1. Bagi Penulis

- Dapat lebih memahami algoritma dalam pencarian string, khususnya Algoritma Boyer Moore, Algoritma Horspool, dan Algoritma Zhu Takaoka beserta perbedaan kinerja ketiganya.
- Dapat lebih memahami cara untuk membandingkan kinerja suatu algoritma.
- Untuk memenuhi salah satu syarat dalam meraih gelar sarjana dalam Fakultas Sains dan Teknologi jurusan Teknik Informatika Universitas Islam Negeri Syarif Hidayatullah Jakarta.

1.4.2. Bagi Universitas

- Mengetahui kemampuan mahasiswa dalam menguasai materi teori yang telah diperoleh pada masa kuliah ataupun materi yang sesuai dengan program studinya.
- Mengetahui kemampuan mahasiswa dalam menerapkan ilmunya dan melakukan perbandingan kinerja dari penerapan algoritma tersebut.

1.4.3. Bagi Pembaca

- Mengetahui cara dan hasil dari perbandingan Algoritma Boyer Moore, Algoritma Horspool, dan Algoritma Zhu Takaoka pada repositori pencarian kata.
- Penelitian ini dapat dijadikan referensi untuk penelitian sejenisnya.

1.5. Batasan Masalah

Di dalam melakukan suatu penelitian diperlukan adanya pembatasan suatu masalah agar penelitian tersebut lebih terarah dan memudahkan dalam pembahasan sehingga tujuan penelitian akan tercapai. Beberapa batasan masalah dalam penelitian ini adalah sebagai berikut:

- Parameter perbandingan kinerja algoritma yang digunakan, yaitu waktu pemrosesan (*runtime*) dan memori yang digunakan untuk pemrosesan (*memory consumption*).
- Skenario yang dibuat sebanyak delapan model dengan lima kali pengulangan pada setiap algoritma-nya.
- Skenario yang diambil dalam simulasi berdasarkan rentang karakter.
- Akurasi (*accuracy*) hanya digunakan sebagai *output* untuk menunjukkan ketepatan pencarian *string* dengan sumber data yang ada.
- Metode pengambilan keputusan menggunakan Metode Perbandingan Eksponensial.
- *Output* banyaknya hasil pencarian akan dikelompokkan berdasarkan hadits.

1.6. Metodologi Penelitian

Untuk mencapai tujuan penelitian ini, maka dalam penelitian ini penulis menggunakan metode-metode sebagai berikut.

1.6.1. Metode Pengumpulan Data

Penulis melakukan pengumpulan data dengan melakukan studi pustaka dan wawancara. Studi pustaka dilakukan dengan mencari informasi melalui buku, jurnal, dan *website* yang bersangkutan terhadap pembahasan penelitian. Hal ini ditujukan untuk data pada latar belakang, landasan teori, literatur sejenis, serta pembelajaran dalam pembahasan penelitian. Selain itu, penulis juga melakukan wawancara kepada narasumber yang ahli dalam bidangnya untuk menentukan bobot kriteria yang penulis gunakan pada metode pengambilan keputusan, yaitu Metode Perbandingan Eksponensial.

1.6.2. Metode Implementasi

Penulis melakukan implementasi penelitian dengan menggunakan metode simulasi yang terbagi dalam beberapa tahap secara berurut, yaitu :

- Formulasi Masalah (*Problem Formulation*)
- Model Pengkonsepan (*Conceptual Model*)
- Data Masukan/Keluaran (*Input/Output Data*)
- Pemodelan (*Modelling*)
- Simulasi (*Simulation*)
- Verifikasi dan Validasi (*Verification and Validation*)
- Eksperimentasi (*Experimentation*)
- Analisis Keluaran (*Output Analysis*)

1.6.3. Metode Pengambilan Keputusan

Penulis menggunakan metode pengambilan keputusan, yaitu Metode Perbandingan Eksponensial yang digunakan untuk perankingan hasil dari masing-masing algoritma, sehingga dapat diketahui algoritma yang terbaik.

1.7. Sistematika Penulisan

Dalam penulisan skripsi ini, penulis membagi sistematika penulisan skripsi ke dalam enam bab yang secara singkat akan peneliti uraikan sebagai berikut.

BAB I PENDAHULUAN

Pada bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi penelitian, dan sistematika penulisan dari penelitian ini.

BAB II LANDASAN TEORI

Pada bab ini berisi tentang deskripsi teori-teori dari pembahasan penelitian dan studi pustaka yang menjadi landasan dalam melakukan penelitian ini.

BAB III METODOLOGI PENELITIAN

Pada bab ini berisi tentang langkah-langkah metodologi penelitian yang penulis gunakan, yaitu metode simulasi dan kerangka berpikir.

BAB IV IMPLEMENTASI SIMULASI DAN EKSPERIMEN

Pada bab ini berisi tentang pelaksanaan implementasi metode simulasi dari tahapan formulasi masalah sampai tahapan simulasi.

BAB V HASIL DAN PEMBAHASAN

Pada bab ini berisi tentang pelaksanaan tahapan verifikasi dan validasi sampai analisis keluaran, yang merupakan hasil dan pembahasan dari penelitian ini.

BAB VI PENUTUP

Pada bab ini memuat penutup yang berisi kesimpulan dari hasil penelitian serta saran untuk penelitian lebih lanjut.

BAB II
LANDASAN TEORI

2.1. Definisi Analisis

Analisis menurut Kamus Besar Bahasa Indonesia adalah penyelidikan terhadap suatu peristiwa (karangan, perbuatan, dsb) untuk mengetahui keadaan yang sebenarnya (sebab-musabab, duduk perkaranya, dsb). Analisis juga bisa diartikan sebagai penguraian dari suatu pokok atas berbagai bagiannya dan penelaahan bagian itu sendiri serta hubungan antarbagian untuk memperoleh pengertian yang tepat dan pemahaman arti keseluruhan (Mulyani, 2016).

Jadi, dapat disimpulkan bahwa analisis adalah melakukan penilaian terhadap suatu masalah atau kejadian untuk mnghasilkan sebuah kesimpulan.

2.2 Studi Pustaka

Studi kepustakaan merupakan studi terhadap kajian teoritis dan referensi lain yang berkaitan dengan nilai, budaya dan norma yang berkembang pada situasi sosial yang diteliti, selain itu studi kepustakaan sangat penting dalam melakukan penelitian, hal ini dikarenakan penelitian tidak akan lepas dari literatur-literatur ilmiah. Penulis menggunakan metode studi pustaka dalam pengumpulan data karena metode ini sangat cocok dilakukan untuk mengumpulkan data dan informasi sebagai bahan dasar peneliti dan acuan dalam penelitian (Sugiyono, 2013).

2.3. Wawancara

Menurut (Rhosa A.S & M. Shahaludin, 2014), pengumpulan data dengan menggunakan wawancara mempunyai beberapa keuntungan sebagai berikut.

- Lebih mudah dalam menggali bagian sistem mana yang dianggap baik dan bagian mana yang dianggap kurang baik.
- Jika ada bagian tertentu yang menurut anda perlu untuk digali lebih dalam, maka dapat langsung menanyakan kepada narasumber.
- Dapat menggali kebutuhan *user* secara lebih bebas.
- *User* dapat mengungkapkan kebutuhannya secara lebih bebas.

Selain memiliki kelebihan, metode wawancara juga memiliki beberapa kelemahan. Berikut ini kelemahan dari metode wawancara.

- Wawancara akan sulit dilakukan jika narasumber kurang dapat mengungkapkan kebutuhannya.
- Pertanyaan dapat menjadi tidak terarah, terlalu fokus pada hal-hal tertentu dan mengabaikan bagian lainnya.

2.4. Metode Simulasi

Menurut (Saputra, 2015) yang dikutip dari skripsi (Fadly F., Masruroh, S. U., & Suseno, 2017) metode simulasi merupakan metode untuk melakukan simulasi dan pemodelan yang diadaptasi dari penelitian yang dilakukan oleh Sajjad A. Madani, Jawad Kazmi, dan Stefan Mahlknecht pada tahun 2010 dengan karya publikasi yang berjudul *"Wireless Sensor Networks: Modelling and Simulation"*. Dalam penelitian tersebut metode simulasi digunakan untuk melakukan pemodelan dan simulasi terhadap *Wireless Sensor Network* (WSN).

Menurut (Sajjad, 2010) yang dikutip dari skripsi (Fikri, Nurhayati, & Masruroh, 2016) metode simulasi terdiri dari beberapa tahapan yang terdiri dari:

2.4.1. Formulasi Masalah (*Problem Formulation*)

Proses simulasi dimulai dengan masalah praktis yang memerlukan pemecahan atau pemahaman. Sebagai contoh sebuah perusahaan kargo ingin mencoba untuk mengembangkan strategi baru untuk pengiriman truk, contoh lain yaitu astronom mencoba memahami bagaimana sebuah nebula terbentuk. Pada tahap ini kita harus memahami perilaku dari sistem, mengatur operasi sistem sebagai objek untuk percobaan. Maka kita perlu menganalisa berbagai solusi dengan menyelidik hasil sebelumnya dengan masalah yang sama. Solusi yang paling diterima yang harus dipilih.

2.4.2. Model Pengkonsepan (*Conceptual Model*)

Langkah ini terdiri dari deskripsi tingkat tinggi dari struktur dan perilaku sebuah sistem dan mengidentifikasi semua benda dengan atribut dan *interface* mereka. Kita juga harus menentukan variabel *state*-nya, bagaimana cara mereka berhubungan, dan mana yang penting untuk penelitian. Pada tahap ini dinyatakan aspek-aspek kunci dari *requirement*.

Selama definisi model konseptual, kita perlu mengungkapkan fitur yang penting. Kita juga harus mendokumentasikan informasi non-fungsional, misalnya seperti perubahan pada masa yang akan datang, perilaku non-*intuitive* atau non-formal, dan hubungan dengan lingkungan.

2.4.3. Data Masukan/ Keluaran (*Input/ Output Data*)

Pada tahap ini kita mempelajari sistem untuk mendapatkan data *input* dan *output.* Untuk melakukannya kita harus mengumpulkan dan mengamati atribut yang telah ditentukan pada tahap sebelumnya. Ketika entitas sistem yang dipelajari, maka dicoba mengaitkannya dengan waktu. Isu penting lainnya pada tahap ini adalah pemilihan ukuran sampel yang valid secara statistik dan format data yang dapat diproses dengan komputer. Kita harus memutuskan atribut mana yang stokastik dan deterministik. Dalam beberapa kasus, tidak ada sumber data yang dapat dikumpulkan (misalnya pada sistem yang belum ada). Dalam kasus tersebut kita perlu mencoba untuk mendapatkan *set* data dari sistem yang ada (jika tersedia). Pilihan lain yaitu dengan menggunakan pendekatan stokastik untuk menyediakan data yang diperlukan melalui generasi nomor acak.

2.4.4. Pemodelan (*Modelling*)

Pada tahap pemodelan, kita harus membangun representasi yang rinci dari sistem berdasarkan model konseptual dan *input/output* data yang dikumpulkan. Model ini dibangun dengan mendefinisikan objek, atribut, dan metode menggunakan paradigma yang dipilih. Pada tahap ini spesifikasi model dibuat, termasuk *set* persamaan yang mendefinisikan perilaku dan struktur. Setelah menyelesaikan definisi ini, kita harus membangun struktur awal model (mungkin berkaitan sistem dan metrik kerja).

2.4.5. Simulasi (*Simulation*)

Pada tahap simulasi, kita harus memilih mekanisme untuk menerapkan model (dalam banyak kasus menggunakan komputer dan bahasa pemrograman dan alat-alat yang memadai), dan model simulasi yang

dibangun. Selama langkah ini, mungkin perlu untuk mendefinisikan algoritma simulasi dan menerjemahkannya ke dalam program komputer.

2.4.6. Verifikasi dan Validasi (*Verification and Validation*)

Pada tahap sebelumnya, tiga model yang berbeda yang dibangun yaitu model konseptual (spesifikasi), sistem model (desain), dan model simulasi (*executable program*). Kita perlu memverifikasi dan memvalidasi model ini. Verifikasi terkait dengan konsistensi internal antara tiga model. Validasi difokuskan pada korespondensi antara model dan realitas yaitu hasil simulasi yang konsisten dengan sistem yang dianalisis.

2.4.7. Eksperimentasi (*Experimentation*)

Kita harus menjalankan model simulasi, menyusul tujuan yang dinyatakan pada model konseptual. Selama fase ini kita harus mengevaluasi *output* dari simulator menggunakan korelasi statistik untuk menentukan tingkat presisi untuk metrik kerja. Fase ini dimulai dengan desain eksperimen, menggunakan teknik yang berbeda. Beberapa teknik ini meliputi analisis sensitivitas, optimasi, dan seleksi (dibandingkan dengan sistem alternatif).

2.4.8. Analisa Keluaran (*Output Analysis*)

Pada tahap analisa keluaran, keluaran simulasi dianalisis untuk memahami perilaku sistem. Keluaran ini digunakan untuk mendapatkan tanggapan tentang perilaku sistem yang asli. Pada tahap ini, alat visualisasi dapat digunakan untuk membantu proses tersebut.

2.5. Repositori

2.5.1. Pengertian Repositori

Dalam buku yang berjudul *Dictionary for Library and Information Science* menyatakan bahwa repositori adalah ruang fisik (bangunan, ruangan, area) yang digunakan untuk cadangan penyimpangan permanen atau penyimpanan sementara (Aulia, Khairani, & Hakiem, 2017).

2.5.2. Fungsi Repositori

Fungsi repositori adalah sebagai berikut (Reitz, 2014) :

- Tempat menyimpan *Structured Information* yang dikumpulkan dari berbagai sumber informasi.
- Sumber referensi bagi proses pembelajaran di *Discussion Forum* dan *Structured Knowledge Creation*.
- Tempat menyimpan pengetahuan yang dihasilkan pada proses pembelajaran di *Discussion Forum* dan *Structured Knowledge Creation*.

Pendapat lain, fungsi dari repositori, yaitu sebagai berikut (Joaqin, 2016) :

1. Fungsi penyimpanan; menyimpan data
2. Fungsi organisasi informasi; mengelola repositori informasi yang dijelaskan dengan skema informasi yang mencakup beberapa unsur berikut :
   - Modifikasi dan pembaruan skema informasi.
   - Peng-*query*-an repositori dengan menggunakan bahasa *query*.
   - Modifikasi dan pembaruan repositori.
3. Fungsi relokasi; mengelola lokasi repositori untuk antarmuka, termasuk lokasi dari fungsi-fungsi manajemen yang mendukung.
4. Fungsi jenis repositori; mengelola spesifikasi jenis repositori dan tipe hubungan
5. Fungsi perdagangan; menangani iklan dan penemuan antarmuka.

2.6. *Corpus Analysis*

Menurut Pusat Bahasa Kemendikbud, berdasarkan kesusastraan, korpus adalah himpunan karangan dengan tema, masalah, pengarang, atau bentuk yang sama. Sedangkan menurut linguistik, korpus merupakan kumpulan ujaran yang tertulis atau lisan yang digunakan untuk menyokong atau menguji hipotesis tentang struktur bahasa, atau korpus dapat juga berarti data yang dipakai sebagai sumber bahan penelitian (Aulia, 2017).

Corpus analysis adalah sebuah bentuk analisis teks yang memungkinkan kita untuk membuat perbandingan antara objek tekstual pada skala besar. Hal ini memungkinkan kita untuk melihat hal-hal yang tidak perlu ketika kita membaca sebagai seorang manusia. Jika kita mempunyai beberapa koleksi dokumen, kita mungkin menemukan pola penggunaan tata bahasa atau frase yang sering berulang di korpus. Kita juga mungkin ingin mencari frase seperti atau tidak seperti secara statistik untuk penulis atau jenis teks tertentu, jenis tertentu dari struktur tata bahasa atau banyak contoh dari konsep tertentu di sejumlah besar dokumen di dalam konteks. Analisis *Corpus* sangat berguna untuk menguji intuisi tentang teks dan/ atau hasil triangulasi dari metode digital lainnya (Aulia, Khairani, Bahaweres, & Hakiem, 2017).

2.7. *String Matching*

String Matching adalah proses pencarian semua kemunculan *query* yang selanjutnya disebut *pattern* ke dalam *string* lebih panjang atau teks (Siahaan & Mesran, 2018). Prinsip kerja algoritma *String Matching* (Effendi, 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 pergesaran 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, 2013), yaitu :

1. *Pattern*, yaitu deretan karakter yang akan dicocokkan dengan teks, dinyatakan dengan x[0... m – 1], panjang *pattern* dinyatakan dengan m.
2. Teks, yaitu tempat pencocokkan *pattern* dilakukan. Dinyatakan dengan y[0... n – 1], panjang teks dinyatakan dengan n.

3. Alfabet, berisi semua simbol yang digunakan oleh bahasa pada teks dan *pattern* dinyatakan dengan ∑ dengan ukuran dinyatakan ASIZE.

2.7.1. Cara Kerja *String Matching*

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.

Dengan sebuah nilai karakter (m < n) yang akan dicari dalam teks. Dalam algoritma pencocokan *string*, teks diasumsikan berada di dalam memori, sehingga bila kita mencari *string* di dalam sebuah arsip, maka semua isi arsip perlu dibaca terlebih dahulu kemudian disimpan di dalam memori. Jika *pattern* muncul lebih dari sekali di dalam teks, maka pencarian hanya akan memberikan keluaran berupa lokasi *pattern* ditemukan pertama kali (Wulan, 2011).

2.7.2. Teknik Algoritma *String Matching*

Menurut (Singh & Verma, 2011), ada dua teknik utama dalam algoritma *string matching*, 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 atau sama persis dengan string masukan. Beberapa algoritma *exact string matching* antara lain :

a. Knuth Morris Pratt

Metode ini mencari kehadiran sebuah kata dalam teks dengan melakukan observasi awal (*preprocessing*) dengan cara mengecek ulang kata sebelumnya. Algoritma ini melakukan pencocokan dari kiri ke kanan.

b. Boyer Moore

Algoritma Boyer Moore adalah algoritma *string matching* yang paling efisien dibandingkan algoritma *string matching* lainnya. Sebelum melakukan pencarian *string*, algoritma melakukan proses terlebih dahulu pada *pattern*, bukan pada *string* pada teks tempat pencarian. Algoritma ini melakukan pencocokan karakter yang dimulasi dari kanan ke kiri. Karena sifatnya yang sangat efisien, Boyer Moore memiliki banyak variasi penyederhanaan.

2. *Approximate String Matching* atau *Fuzzy String Matching*

Fuzzy String Matching merupakan pencocokan *string* secara samar, maksudnya pencocokan *string* dimana *string* yang dicocokkan memiliki kemiripan susunan karakter yang berbeda (mungkin jumlah atau urutannya), tetapi *string* tersebut memiliki kemiripan baik kemiripan tekstual/penulisan (*approximate string matching)* atau kemiripan ucapan (*phonetic string matching*).

2.8. Algoritma

Algoritma adalah susunan langkah penyelesaian suatu masalah secara sistematika dan logis (Sitorus, 2015). Algoritma yang baik adalah suatu langkah-langkah yang mengahasilkan *output* yang efektif dalam waktu yang relatif singkat dan penggunaan memori yang relatif sedikit (efisien) dengan langkah-langkah yang berhingga dan prosedurnya berakhir baik dalam keadaan diperoleh suatu solusi ataupun tidak ada solusinya (Kurniadi, 2013). Dalam buku Algoritma dan

Pemrograman oleh (Munir, 2016) , algoritma adalah urutan langkah-langkah untuk memecahkan suatu masalah.

Jadi dapat disimpulkan, algoritma adalah langkah-langkah penyelesaian masalah berupa instruksi yang sistematis dan logis untuk menghasilkan suatu keluaran.

2.8.1. Algoritma Boyer Moore

Algoritma Boyer-Moore adalah salah satu algoritma untuk mencari suatu *string* di dalam teks, dibuat oleh R.M Boyer dan J.S Moore. Algoritma melakukan perbandingan dimulai dari kanan ke kiri, tetapi pergesaran *window* tetap dari kiri ke kanan. Jika terjadi kecocokan maka dilakukan perbandingan karakter teks dan karakter pola yang sebelumnya, yaitu sama-sama mengurangi indeks teks dan pola masing-masing sebanyak satu (Argakusumah & Hansun, 2016).

- Langkah-langkah Algoritma Boyer Moore sebagai berikut :
  1. Buat tabel pergesaran *string* yang dicari dengan pendekatan *Match Heuristic* (MH) dan *Occurence Heuristic* (OH), untuk menentukan jumlah pergesaran yang akan dilakukan jika mendapat karakter tidak cocok pada proses pencocokan dengan *string*.
  2. Jika dalam proses perbandingan terjadi ketidakcocokan antara pasangan karakter pada S dan karakter pada T, pergesaran dilakukan dengan memilih salah satu nilai pergesaran dari dua tabel analisa *string* yang memiliki nilai pergesaran paling besar.
  3. Dua kemungkinan penyelesaian dalam melakukan pergesaran S, jika sebelumnya belum ada karakter yang cocok adalah dengan melihat nilai pergesaran hanya pada tabel *Occurance Heuristic*, jika karakter yang tidak cocok tidak ada pada S, maka pergeseran adalah sebanyak jumlah karakter pada S dan jika karakter yang tidak cocok ada pada S, maka banyaknya pergesaran bergantung dari nilai pada tabel.

4. Jika karakter pada teks yang sedang dibandingkan cocok dengan karakter pada S, maka posisi karakter pada S dan T diturunkan sebanyak 1 posisi, kemudian dilanjutkan dengan pencocokan pada posisi tersebut dan seterusnya. Jika kemudian terjadi ketidakcocokan karakter S dan T, maka dipilih nilai pergesaran terbesar dari dua tabel analisa *pattern*, yaitu nilai dari tabel *Match Heuristic* dan nilai tabel *Occurance Heuristic* dikurangi dengan jumlah karakter yang telah cocok.
5. Jika semua karakter telah cocok, artinya S telah ditemukan di dalam T, selanjutnya geser *pattern* sebesar 1 karakter.
6. Lanjutkan sampai akhir *string* T (Argakusumah & Hansun, 2016).

- Cara menghitung *Bad-Character Shift (Occurance Heuristic)*

Untuk menghitung tabel *Occurance Heuristic* harus menggunakan langkah-langkah sebagai berikut (Argakusumah & Hansun, 2016).

Contoh *string* : manaman

Panjang : 7 karakter

Tabel 2.1 *Occurance Heuristic*

Posisi	1	2	3	4	5	6	7
String	M	A	N	A	M	A	N
Pergeseran (OH)	2	1	0	1	2	1	0

1. Lakukan pencacahan mulai dari posisi terakhir *string* sampai ke posisi awal, dimulai dengan nilai 0 karena terletak pada jarak terakhir, catat karakter yang sudah ditemukan (dalam contoh ini karakter “n”).
2. Mundur ke posisi sebelumnya, nilai pencacah ditambah 1, jika karakter pada posisi ini belum pernah ditemukan, maka nilai pergesarannya adalah sama dengan nilai pencacah (dalam

contoh ini, karakter “a” belum pernah ditemukan sehingga nilai pergesarannya adalah sebesar nilai pencacah yaitu 1).

3. Mundur keposisi sebelumnya karakter “m” nilai pergeserannya adalah 2.
4. Mundur lagi, karakter “a”. Karakter “a” sudah pernah ditemukan sebelumnya sehingga nilai pergesarannya sama dengan nilai pergeseran karakter “a” yang sudah ditemukan paling awal yaitu 1.
5. Begitu seterusnya sampai posisi awal *string*.

- Cara menghitung *Good-Suffiz Shift (Match Heuristic)*

Tabel *Match Heuristic* sering disebut juga *Good-Suffix Shift*, dimana pergesarannya dilakukan berdasarkan posisi ketidakcocokan karakter yang terjadi (Argakusumah & Hansun, 2016).

Contoh *string* : manaman

Panjang : 7 karakter

Tabel 2.2 Tabel *Match Heuristic*

Posisi	1	2	3	4	5	6	7
String	M	A	N	A	M	A	N
Pergeseran (MH)	4	4	4	4	7	7	1

1. Jika karakter pada posisi 7 bukan “n” maka geser 1 posisi, berlaku untuk semua *pattern* yang dicari.
2. Jika karakter “n” sudah cocok, tetapi karakter sebelum “n” bukan “a”, maka geser sebanyak 7 posisi, sehingga posisi *pattern* melewati teks, karena sudah pasti “manambn” bukan “manaman”.
3. Jika karakter “an” sudah cocok, tetapi karakter sebelum “an” bukan “m” maka geser sebanyak 7 posisi, sehingga posisi *pattern* melewati teks, karena sudah pasti “manaban” bukan “manaman”.

4. Jika karakter “man” sudah cocok, tetapi karakter sebelum “man” bukan “a” maka geser sebanyak 4 posisi, sehingga posisi *pattern* berada atau bersesuaian dengan akhiran “man” yang sudah ditemukan sebelumnya, karena bisa saja akhiran “man” yang sudah ditemukan sebelumnya merupakan awalan dari *pattern* “manaman” yang berikutnya.
5. Jika karakter “aman” sudah cocok, tetapi karakter sebelum “aman” bukan “n” maka geser sebanyak 4 posisi, sehingga posisi *pattern* berada/bersesuaian dengan akhiran “man” yang sudah ditemukan sebelumnya, karena bisa saja akhiran “man” yang sudah ditemukan sebelumnya merupakan awalan dari *pattern* “manaman” yang berikutnya.

   Selanjutnya sama, pergesaran paling mungkin dan aman dalam tabel *Match Heuristic* adalah pergesaran sebanyak 4 posisi (Argakusumah & Hansun, 2016).

- Contoh Penerapan Algoritma Boyer Moore

Contoh penggunaan algoritma Boyer Moore dalam melakukan pencarian dalam teks (Fau, Mesran, & Ginting, 2017):

Teks (S) : Pengolahan Citra Digital

Pattern (P) : Citra

Tahapan pencarian *pattern* (P) dalam Teks (S) :

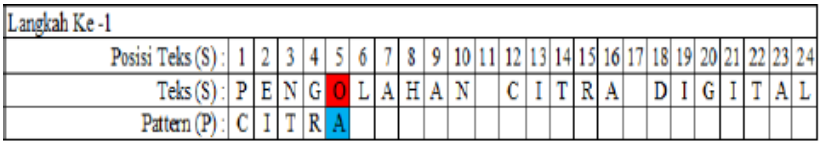
Tabel 2.3 Analisa Penentuan Nilai OH dan MH

Pattern (P)	C	I	T	R	A
Occurcance Heuristic (OH)	4	3	2	1	0
Match Heuristic (MH)	5	5	5	5	1

Langkah – Langkah Proses Algoritma Boyer Moore :

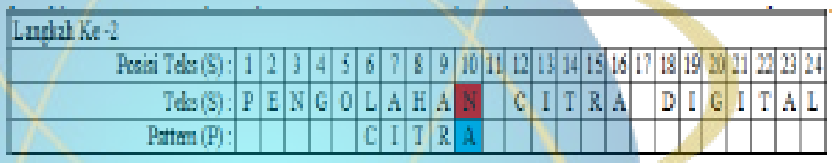
- Pada pergeseran pertama karakter “A” pada *pattern* tidak cocok dengan arakter “O” pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter “O”

tidak terdapat pada tabel, sehingga pergeseran selanjutnya adalah sebanyak jumlah karakter "A" pada *pattern* yaitu 5.



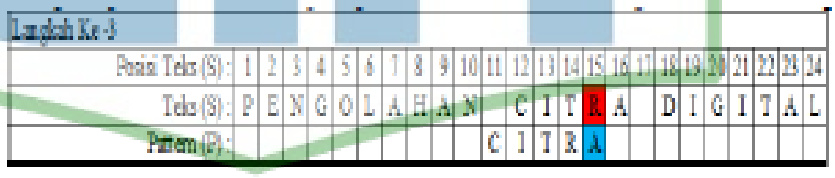
Gambar 2.1 Langkah Ke-1 Pemrosesan

- Pada pergeseran ke-2 karakter "A" pada *pattern* tidak cocok dengan karakter "N" pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter "N" tidak terdapat pada tabel, sehingga pergeseran selanjutnya adalah sebanyak umlah karakter "A" pada *pattern* yaitu 5.



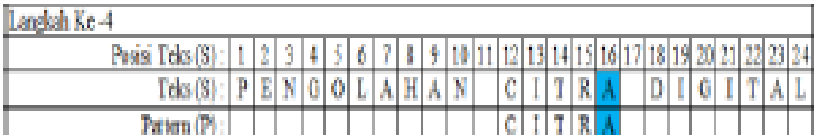
Gambar 2.2 Langkah Ke-2 Pemrosesan

- Pada pergeseran ke-3 karakter "A" pada *pattern* tidak cocok dengan karakter "R" pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter R terdapat pada tabel, sehingga pergeseran selanjutnya adalah sebanyak jumlah karakter "R" pada tabel "OH" yaitu 1.



Gambar 2.3 Langkah Ke-3 Pemrosesan

- Pada pergeseran ke-4 karakter "A" pada *pattern* cocok dengan karakter "A" pada teks, maka pergeseran selanjutnya dimundurkan satu langkah.



Gambar 2.4 Langkah Ke-4 Pemrosesan

- Pada pergeseran selanjutnya dilakukan sampai pada pergeseran ke-12 karakter "C" pada *pattern* dengan karakter "C" pada teks cocok (Fau, Mesran, & Ginting, 2017).

2.8.2. Algoritma Horspool

Algoritma Horspool adalah modifikasi dari algoritma Boyer Moore dengan sedikit perubahan. Tidak seperti algoritma Boyer Moore, algoritma Horspool hanya menggunakan satu tabel (*bad character shift*) dimana algoritma Boyer Moore menggunakan dua tabel: *bad character shift* dan *good suffix shift*. Algoritma Horspool mencari *pattern* dari kiri ke kanan dan untuk *shift value* berdasarkan ukuran dari *pattern* yang dicari dalam *bad character shift* tabel (Adhi & Khrisnandi, 2017).

- Langkah-langkah yang pencarian dengan Algoritma Horspool

  Terhadap dua tahap pada pencocokkan *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 :

     - Algoritma Horspool melakukan pencocokan karakter ter-kanan pada *pattern*
     - Setiap karakter pada *pattern* ditambah ke dalam tabel *bad-match* dan dihitung nilai *shift*-nya
     - Karakter yang berada pada ujung *pattern* tidak dihitung dan tidak dijadikan karakter ter-kanan dari karakter yang sama dengannya

- Apabila terdapat dua karakter yang sama dan salah satunya bukan karakter ter-kanan, maka karakter dengan indeks terbesar yang dihitung nilai *shift*-nya
- Algoritma horspool menyimpan panjang dari *pattern* sebagai panjang nilai *shift* secara *default* apabila karakter pada teks tidak ditemukan dalam *pattern*
- Nilai (*value*) *shift* yang akan digunakan dapat dicari dengan perhitungan panjang dari *pattern* dikurang indeks terakhir karakter dikurang 1, untuk masing-masing karakter, *value* = m – i – 1

  Sebagai contoh, dapat dilihat pada Tabel 2.4 berikut.

  Pattern : KARTIKA

K	A	R	T	I	K	A
0	1	2	3	4	5	6

Tabel 2.4 *Bad Match* pada Praproses

Karakter	Index	*Value*
K	5	1
A	1	5
R	2	4
T	3	3
I	4	2
*	-	1

Value = 7 – 5 – 1 = 1

Value = 7 – 1 – 1 = 5

Value = 7 – 2 – 1 = 4

Value = 7 – 3 – 1 = 3

Value = 7 – 4 – 1 = 2

* : karakter yang tidak dikenali

2. Tahap pencarian

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

- Dilakukan perbandingan karakter paling kanan terdapat *window.*
- Tabel *bad-match* digunakan untuk melewati karakter ketika ketidakcocokan terjadi.
- Ketika ada ketidakcocokan, maka karakter paling kanan pada *window* berfungsi sebagai landasan untuk menentukan karakter *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.

- Contoh Penerapan algoritma Horspool

Untuk menggambarkan rincian algoritma, akan diberikan contoh kasus dimana *pattern* P = "KARTIKA" dan T = 'ADEMUTIARA KARTIKA". Inisialisasi awal dan pembuatan *bad-match* terlihat pada tabel 2.5 dan tabel 2.6 berikut (Singh & Verma, 2011) :

Tabel 2.5 Inisialisasi Awal *Bad Match*

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T	A	D	E	M	U	T	I	A	R	A		K	A	R	T	I	K	A
P	K	A	R	T	I	K	A											
i	0	1	2	3	4	5	6											

Tabel 2.6 Pembuatan *Bad Match*

P	A	R	T	I	K	*
I	1	2	3	4	5	*
V	5	4	3	2	1	7

Seperti yang terlihat pada Tabel 2.5 di atas, inisialisasi awal *bad-match* dilakukan. Setiap teks dan *pattern* masing-masing diberi nilai *m* dan *i* dimana m sebagai panjang *pattern* dan i sebagai indeks. Tabel 2.5 menunjukkan nilai pergeseran *bad-match* dengan menghitung nilai v seperti yang telah dilakukan pada 2.6. Pada tahap awal pencarian, dilakukan perbandingan karakter paling kanan *pattern* terhadap *window*. Apabila terjadi ketidakcocokan, akan dilakukan pergeseran ke kanan untuk melewati karakter yang tidak cocok dimana nilai pergeserannya terdapat pada tabel *bad-match*. Karakter paling kanan teks pada *window* berfungsi sebagai landasan untuk menentukan jarak geser yang akan dilakukan. Hal ini terlihat pada 2.7.

Tabel 2.7 Iterasi Algoritma Horspool Pertama

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T	A	D	E	M	U	T	I	A	R	A		K	A	R	T	I	K	A
P	K	A	R	T	I	K	A											
i	0	1	2	3	4	5	6											

Terjadi ketidakcocokan seperti yang terlihat pada tabel 2.7. Karakter "I" adalah karakter paling kanan teks pada *window*. Pada tabel *bad-match*, nilai geser karakter "I" adalah 2. Maka, dilakukan pergeseran ke kanan pada *window* sebanyak 2 kali. Hal ini terlihat pada tabel 2.8.

Tabel 2.8 Iterasi Algoritma Horspool Kedua

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T	A	D	E	M	U	T	I	A	R	A		K	A	R	T	I	K	A
P			K	A	R	T	I	K	A									
i			0	1	2	3	4	5	6									

Pada tabel 2.8 terdapat ketidakcocokan kembali antara karakter "R" dan "A". Pada tabel *bad-match*, nilai geser karakter "R" adalah 4. Maka, dilakukan pergeseran ke kanan pada *window* sebanyak 4 kali. Hal ini terdapat pada tabel 2.9.

Tabel 2.9 Iterasi Algoritma Horspool Ketiga

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T	A	D	E	M	U	T	I	A	R	A		K	A	R	T	I	K	A
P							K	A	R	T	I	K	A					
I							0	1	2	3	4	5	6					

Pada iterasi ketiga yang terlihat pada tabel 2.9 , kecocokan *pattern* dan teks terjadi pada karakter "A" dan karakter "K". Namun, kembali terjadi ketidakcocokan antara karakter "I" dan "(spasi)". Maka, karakter paling kanan teks pada *window* menentukan karak geser yang dilakukan. Sebagaimana yang terlihat pada tabel 2.9, karakter "A" berfungsi sebagai landasan nilai geser. Nilai geser karakter "A" yang terdapat pada tabel *bad-match* adalah 5. Maka dilakukan pergeseran ke kanan pada *window* sebanyak 5 kali dan dilanjutkan dengan iterasi keempat seperti yang terlihat pada tabel 2.10.

Tabel 2.10 Iterasi Algoritma Horspool Keempat

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T	A	D	E	M	U	T	I	A	R	A		K	A	R	T	I	K	A
P												K	A	R	T	I	K	A
I												0	1	2	3	4	5	6

Pada tabel 2.10, window telah berada pada akhir teks dan semua *pattern* cocok dengan teks. Seluruh pencocokan karakter menggunakan algoritma Horspool telah selesai dan berhenti di iterasi keempat (Singh & Verma, 2011).

2.8.3. Algoritma Zhu Takaoka

Menurut (Togatorop, Aan, & Coastera, 2017) Algoritma Zhu-Takaoka merupakan modifikasi dari algoritma pencocokan *string Boyer-Moore Algorithm* yang dibuat oleh Boyer R.S dan Moore J.S tahun 1977. Algoritma Zhu Takaoka dipublikasikan dan dikembangkan oleh Zhu Rui Feng dan Tadao Takaoka pada tahun 1986.

- Langkah-langkah Algoritma Zhu Takaoka sebagai berikut.
    1. Menjalankan prosedur preZTBc dan preBmGs untuk mendapatkan inisialisasi.
        a. Menjalankan prosedur preZTBc. Fungsi dari prosedur ini adalah untuk menentukan berapa besar pergeseran yang dibutuhkan untuk mencapai karakter tertentu pada *pattern* dari dua karakter *pattern* terakhir/terkanan. Hasil dari prosedur preZTBc disimpan pada tabel ZTBc.
        b. Menjalankan prosedur preBmGs. Sebelum menjalankan isi prosedur ini, prosedur *suffix* dijalankan terlebih dulu pada *pattern*. Fungsi dari prosedur *suffix* adalah memeriksa kecocokan sejumlah karakter yang dimulai dari karakter terakhir/terkanan dengan sejumlah karakter yang dimulai dari setiap karakter yang lebih kiri dari karakter terkanan tadi. Hasil dari prosedur *suffix* disimpan pada tabel *suff*. Jadi *suff*[i] mencatat panjang dari *suffix* yang cocok dengan segmen dari *pattern* yang diakhiri karakter ke-i.
        c. Dengan prosedur preBmGs, dapat diketahui berapa banyak langkah pada *pattern* dari sebeuah segmen ke segmen lain yang sama yang letaknya lebih kiri dengan karakter di sebelah kiri segmen yang berbeda. Prosedur preBmGs menggunakan tabel *suff* untuk mengetahui semua pasangan segmen yang sama. Hasil dari prosedur preBmGs disimpan pada tabel BmGs.
    2. Dilakukan proses pencarian *string* dengan menggunakan hasil dari prosedur preBmBc dan preBmGs yaitu tabel BmBc dan BmGs (Togatorop, Aan, & Coastera, 2017)
- Contoh Pencarian dalam Zhu Takaoka

    Langkah pertama yang dilakukan adalah tahapan *preprocessing* yaitu meciptakan dua buah tabel *shif*/pergesaran ztBc (*Zhu-Takaoka Bad Character)* dan bmGs *(Boyer-Moore Good Suffixes).* Kedua tabel

berikut ini ini diciptakan merujuk kepada *pattern* yang akan dicari sehingga *pattern* berubah maka tabel juga akan berubah. Hasil *preprocessing* untuk *pattern* ABEG terlihat pada tabel 2.11 dan tabel 2.12.

Tabel 2.11 Zhu Takaoka *Bad Character Table*

ZtBc	A	B	E	G	*
A	3	2	4	4	4
B	3	4	1	4	4
E	3	4	4	4	4
G	3	4	4	4	4
*	3	4	4	4	4

Tabel *ztBc* berbentuk *array* dua dimensi yang baris dan kolom diisi sesuai dengan karakter yang ada pada *pattern*, tanda *(star) mewakili seluruh karakter yang tidak ada pada *pattern*.

Tabel 2.12 Boyer Moore *Suffixes Table*

I	0	1	2	3
x [i]	A	B	E	G
Suff [i]	0	0	0	4
bmGs [i]	3	4	4	1

Tahapan selanjutnya adalah tahapan pencarian yaitu dengan menggunakan teknik *right-to-left scan rule*. Pencarian dilakukan dengan membandingkan karakter demi karakter dari mulai karakter paling kanan menuju karakter paling kiri. Jika terjadi ketidakcocokan karakter, pergeseran akan dilakukan dengan mencari nilai *max* antara *ztBc* dan *bmGs*, dan apabila semua *pattern* cocok pergeseran menggunakan nilai dari *bmGs*.

- Langkah-langkah pencarian dengan algoritmaa Zhu Takaoka adalah sebagai berikut (Togatorop, Aan, & Coastera, 2017).

- Langkah 1

Tabel 2.13 Pencarian pada Teks Langkah Ke-1

Window			*	*											
Teks	V	A	R	I	A	T	I	O	N		A	B	E	G	G
Pattern	A	B	E	G											
I	0	1	2	3											

[*][*] = 4

[] = [3] = 1

Pergeseran dilakukan sebanyak 4 (pergeseran maksimal)

- Langkah 2

Tabel 2.14 Pencarian pada Teks Langkah Ke-2

Window							*	*							
Teks	V	A	R	I	A	T	I	O	N		A	B	E	G	G
Pattern					A	B	E	G							
I					0	1	2	3							

[*][*] = 4

[] = [3] = 1

Pergeseran dilakukan sebanyak 4

- Langkah 3

Tabel 2.15 Pencarian pada Teks Langkah Ke-3

Window											A	B			
Teks	V	A	R	I	A	T	I	O	N		A	B	E	G	G
Pattern									A	B	E	G			
I									0	1	2	3			

[*][*] = 2

[] = [3] = 1

Pergeseran dilakukan sebanyak 2

- Langkah 4

Tabel 2.16 Pencarian pada Teks Langkah Ke - 4

Window													E	G	
Teks	V	A	R	I	A	T	I	O	N		A	B	E	G	G
Pattern											A	B	E	G	
I											0	1	2	3	

Karakter cocok

Pergeseran tidak dilakukan karena panjang karakter teks tidak memenuhi.

2.9. Hadits atau As-Sunnah

Menurut (Idri, 2017) di kalangan ulama Hadits (*al-muhadditsun*), kata Hadits sering diartikan dengan kata Sunnah, khabar, atsar. Kata Hadits lebih sering digunakan baik kalangan ulama maupun umat Islam umumnya daripada kata Sunnah, Khabar, dan atsar. Secara bahasa, kata Hadits (*al-hadits*) berarti baru, yaitu sesuatu yang baru. Menurut Ibn Hajar Al-'Asqalani, sebagaimana dikutip oleh Subhi al-Salih, yang dimaksud dengan Hadits dalam tradisi *syara'* adalah sesuatu yang disandarkan kepada Nabi SAW seolah-olah dimaksudkan sebagai bandingan dengan Al-Quran yang bersifat *qadim*. Di samping berarti baru, *al-hadits* juga mengandung arti dekat yaitu sesuatu yang dekat, yang belum lama terjadi dan juga berarti berita yang sama dengan *hiddit*, yaitu sesuatu yang dipercakapkan dan dipindahkan dari seseorang pada orang lain.

2.10. Hadits Bukhori

Nama lengkap Imam Bukhori adalah Abu Abdullah Muhammad bin Ismail bin Ibrohim bin al-Mughiroh bin Bardizbahal al-Bukhori. Ia lahir di Bukhoro. Sejak kecil Bukhori sudah menampakkan kepribadian yang mulia, serta memiliki kecerdasan dan daya hafal yang luar biasa. Ia sudah mempelajari hadits sebelum genap berusia 10 tahun. Pada usia 11 tahun, ia telah mampu mengoreksi kesalahan *sanad* hadits. Ketika berusia 15 tahun, ia sudah hafal itab Ibnu Mubarok dan Waqi, serta mampu memahami pendapat *ahlu ro'yi* (rasionalis), *ushul*, dan *mazhab* mereka. Dan, pada usia 18 tahun, ia telah menulis kitab *Qoda ya as-Sahabt wa at-Tabi'in* (Bustamin & Hasanuddin, 2010).

Untuk memperluas dan memperdalam pengetahuannya tentang Hadits, Bukhori melawat ke berbagai negeri Irak, Syiria, Meisr, dan Kuffah, Basroh, Khurasa, dan lain-lain. Di negeri-negeri tersebut ia menuntut ilmu pada banyak ulama hadits terkemuka. Dalam pengembaraannya ia selalu mengumpulkan dan menulis hadits. Bahkan ia sempat menghafal 100.000 hadits shohih dan 200.000 hadits tidak shohih (Bustamin & Hasanuddin, 2010).

Kitab shohih Bukhori (Al-Jami' as Shohih Bukhori), berisi 7.275 hadits yang merupakan seleksi dari 600.000 hadits. Kitab ini juga memuat fatwa sahabat dan tabi'in sebagai penjelasan terhadap hadits yang diketengahkan.Sebagian ulama berpendapat bahwa nilai Shohih Bukhori lebih tinggi dibanding Shohih Muslim. Alasan mereka karena syarat yang ditetapkan oleh Imam Bukhori lebih ketat dibandingkan dengan syarat yang ditetapkan oleh Imam Muslim. Syarat yang dimaksud, diantaranya, Bukhori menetapkan *liqoo* (bertemunya antara rawi yang menyampaikan dengan rawi yang menerima). Sebaliknya, bagi Imam Muslim, syarat tersebut cukuplah *mu'aasarob* (perawi yang menyampaikan dan perawi yang menerima itu hidup dalam satu masa) (Bustamin & Hasanuddin, 2010).

2.11. Metode Perbandingan Eksponensial

Metode perbandingan eksponensial (MPE) merupakan salah satu metode pengambilan keputusan yang mengkualifikasikan pendapat seseorang atau lebih dalam skala tertentu. Metode ini mampu menentukan urutan prioritas alternatif keputusan dengan menggunakan beberapa kriteria (Kriteria Majemuk) (Sari, 2018).

2.11.1. Tahapan Metode Perbandingan Eksponensial

Menurut (Pratiwi, 2016), tahapan metode perbandingan eksponensial sebagai berikut:

1. Menyusun alternatif-alternatif keputusan yang akan dipilih
2. Menentukan kriteria atau perbandingan relatif kriteria keputusan yang penting untuk dievaluasi dengan menggunakan skala konversi tertentu sesuai dengan keinginan pengambil keputusan

3. Menentukan tingkat kepentingan relatif dari setiap kriteria keputusan atau pertimbangan kriteria. Penentuan bobot ditetapkan pada setiap kriteria untuk menunjukkan tingkat kepentingan suatu kriteria
4. Melakukan penilaian terhadap semua alternatif pada setiap kriteria dalam bentuk total skor tiap alternatif.

2.11.2. Formulasi Perhitungan Metode Perbandingan Eksponensial

Formulasi perhitungan total nilai setiap pilihan keputusan adalah sebagai berikut (Pratiwi, 2016) :

$$\text{Total Nilai } (TN_i) = \sum_{j=1}^{m} (RKij)^{TKKj}$$

Keterangan :

TN_i = Total nilai alternatif ke-i

Rkij = Derajat kepentingan relatif kriteria ke-j pada pilihan keputusan i

TKKj = Derajat kepentingan kriteria keputusan ke-j; TKKj > 0; bulat

n = Jumlah pilihan keputusan

m = Jumlah kriteria keputusan

Penentuan tingkat kepentigan kriteria dilakukan dengan cara wawancara dengan si pengambil keputusan atau melalui kesepakatan curah pendapat. Sedangkan penentuan skor alternatif pada kriteria tertentu dilakukan dengan memberi nilai setiap alternatif berdasarkan nilai kriterianya. Semakin besar nilai alternatif semakin besar pula skor alternatif tersebut. Total skor masing-masing alternatif keputusan akan relatif berbeda secara nyata karena adanya fungsi eksponensial.

2.11.3. Keuntungan Metode Perbandingan Eksponensial

Metode Perbandingan Eksponensial dapat mengurangi bias yang mungkin terjadi dalam analisis. Nilai skor yang menggambarkan urutan prioritas menjadi besar dalam hal fungsi eksponensial ini menyebabkan urutan prioritas alternatif keputusan menjadi lebih nyata (Pratiwi, 2016).

2.12. Studi Literatur Sejenis

Dalam penelitian ini, penulis menggunakan literatur penelitian sejenis yang sudah ada sebelumnya. Hal ini dimaksudkan untuk membandingkan studi literatur tersebut. Berikut ini tabel literatur sejenis.

Tabel 2.17 Tabel Studi Literatur Sejenis

No	Judul Penelitian	Algoritma					Basis Aplikasi			Paramater Analisis	
		Hors pool	Zhu Takaoka	Boyer Moore	Knuth Morris Pratt	Brute Force	Desktop	Android	Repositori	Waktu	Memori
1.	Perbandingan Algoritma Horspool dan Algoritma Zhu-Takaoka dalam Pencarian *String* Berbasis Desktop (Adhi Kusnadi, Abraham Khrisnandi Wicaksono, 2017)	✓	✓	-	-	-	✓	-	-	✓	-
2.	Analisa Perbandingan Boyer Moore dan Knuth Morris Pratt dalam Pencarian Judul Buku Menerapkan Metode Perbandingan	-	-	-	✓	✓	✓	-	-	✓	✓

	Eksponensial (Studi Kasus : Perpustakaan STMIK Budi Darma) (Alwin Fau, Mesran, Guidio Leonarde Ginting, 2017)										
3.	Studi Perbandingan Algoritma Brute Force, Algoritma Knuth Morris Pratt, Algortima Boyer Moore untuk Identifikasi Kesalahan Penulisan Teks Berbasis Android (Ditya Geraldy, Eko Sediyono, Yos Richard Beeh, 2016)	-	-	✓	✓	✓	-	✓	-	✓	-
4	Pengembangan Repositori Hadits Terjemahan Bahasa	-	-	-	-	✓	-	-	✓	-	-

	Indonesia (Studi Kasus: Hadits Bukhori) (Atqia Aulia, 2017)										
5.	Implementation and Analysis Zhu Takaoka and Knuth Morris Pratt Algorithm for Dictionary of Computer Application Based on Android (Handrizal, Andri Budiman, Desy Rahayu Ardani, 2017)	-	✓	-	✓	-	-	✓	-	✓	-
6.	Analisis Perbandingan Kinerja Algoritma Boyer Moore, Algoritma Horspool, dan Algoritma Zhu Takaoka pada Repositori Hadits	✓	✓	✓	-	-	-	-	✓	✓	✓

	Bukhori Terjemahan Bahasa Indonesia (Nindy Raisa Hanum, 2018)										

Berdasarkan studi literatur yang telah dijelaskan di atas, penulis melakukan penelitian untuk menganalisis perbandingan algoritma Boyer Moore, Horspool, dan Zhu Takaoka dengan parameter pembandingnya adalah waktu pemrosesan (*runtime*) dan memori yang digunakan dalam pemrosesan (*memory consumption*) yang diimplementasikan pada repositori Hadits Bukhori terjemahan Bahasa Indonesia dengan menggunakan bahasa pemrograman PHP.

.

BAB III
METODOLOGI PENELITIAN

3.1. Metode Pengumpulan Data

Dalam skripsi ini penulis menggunakan dua metode dalam pengumpulan data, yaitu melakukan wawancara dan simulasi untuk memperoleh data primer dan studi pustaka untuk memperoleh data sekunder.

3.1.1. Data Primer

3.1.1.1. Data Simulasi

Data primer penulis peroleh dengan cara melakukan simulasi langsung dengan menggunakan algoritma Boyer Moore, Horspool, dan Zhu Takaoka yang diimplementasikan pada repositori Hadits Bukhori terjemahan Bahasa Indonesia dengan membandingkan hasil kinerja masing-masing algoritma terhadap beban kerja yang diberikan sesuai dengan delapan skenario dengan lima kali pengulangan yang telah penulis tetapkan.

3.1.1.2. Wawancara

Pada metode ini penulis melakukan wawancara kepada pihak ahli dalam data, yaitu dengan pihak PUSTIPANDA UIN Syarif Hidayatullah Jakarta, Bapak Nashrul Hakiem selaku ketua PUSTIPANDA UIN Syarif Hidayatullah Jakarta. Wawancara telah dilaksanakan pada Jumat, 20 Juli 2018 pada pukul 13.20 WIB. Wawancara ini berfungsi untuk menentukan bobot pada parameter yang digunakan penulis dalam metode pengambilan keputusan dengan menggunakan Metode Perbandingan Eksponensial.

3.1.2. Data Sekunder

3.1.2.1. Studi Pustaka

Penulis melakukan pengumpulan data dalam penelitian ini. Pengumpulan tersebut dilakukan dengan mencari data-data pendukung untuk latar belakang dan landasan teori, membandingkan dengan penelitian sejenis, dan memahami teori maupun terapan yang sesuai dengen penelitian ini. Data-data tersebut diperoleh dari berbagai sumber, seperti buku, jurnal, *e-book*, dan skripsi peneliti sebelumnya.

3.2. Metode Simulasi

Dalam penelitian ini, penulis menggunakan metode simulasi dalam membandingkan kinerja dari algoritma *string matching* yang ada pada repositori hadits bukhori terjemahan Bahasa Indonesia. Sehubungan dengan itu, metode simulasi ini dibagi menjadi delapan tahapan yang terdiri dari :

a. Formulasi Masalah (*Problem Formulation*)

Tahap formulasi masalah merupakan langkah awal dalam perancangan pada model motode simulasi. Formulasi masalah merupakan suatu kegiatan untuk memilih satu permasalah yang penting untuk dianalisis dan diselesaikan. Dengan semakin berkembangnya teknologi informasi sehingga informasi semakin mudah di dapat. Salah satunya dengan memasukkan kata yang ingin dicari dari sekumpulkan data yang telah digitalisasi. Setelah melakukan pengumpulan data melalui studi pustaka, penulis dapat memformulasikan sebuah masalah yaitu analisis perbandingan kinerja pada penggunaan repositori Hadits Bukhori terjemahan Bahasa Indonesia dengan menggunakan Algoritma Boyer Moore, Horspool, dan Zhu Takaoka. Hal ini bertujuan untuk memberikan solusi yang lebih baik dalam proses *string matrching* pada repositori tersebut.

b. Model Pengkonsepan (*Conceptual Model*)

Pada tahapan ini, penulis membuat model pengkonsepan yang diklasifikasi menjadi dua, yaitu yang pertama ialah konsep proses internal dari Algoritma Boyer Moore, Horspool, dan Zhu Takaoaka yang kedua ialah konsep proses simulasi. Konsep yang pertama

menjelaskan contoh pencarian yang terjadi di dalam proses internal Algoritma Boyer Moore, Horspool, dan Zhu Takaoka secara manual dalam suatu kata. Untuk konsep yang kedua, menjelaskan bagaimana konsep pertama dikemas berdasarkan formulasi masalah yang dihasilkan dari tahapan sebelumnya ke dalam bentuk simulasi yang nantinya akan dijalankan. Dalam hal ini, penulis menggambarkan alur dari proses pencarian kata yang akan dijalankan untuk mendapatkan perhitungan waktu pemrosesan yang terlibat dalam proses *string matching* tersebut maupun pengambilan data waktu pemrosesannya.

c. Data Masukan/ Keluaran (*Input/ Output Data*)

Langkah selanjutnya yaitu menentukan *input* yang akan diproses dan *output* yang akan didapat. *Input* dibutuhkan pada simulasi ini yaitu kata kunci hadits yang ingin dicari pada repositori. *Output* yang didapat dalam simulasi ini yaitu isi hadits, data *runtime*, *memory consumption*, dan ketepatan (*accuracy*) kata yang dicari.

d. Pemodelan (*Modelling*)

Pada tahap ini, penulis menentukan model skenario yang akan digunakan dalam tahap simulasi. Terdapat delapan model skenario berdasarkan kombinasi dari data masukan ukuran *pattern*. Tiap algoritma baik Algoritma Boyer Moore, Algoritma Horspool, dan Algoritma Zhu Takaoka masing-masing melakukan delapan model skenario tersebut.

e. Simulasi (*Simulation*)

Pada tahap ini, sistem akan dijalankan untuk mensimulasikan kinerja masing-masing algoritma sesuai dengan konsep dan juga skenario yang telah ditentukan sebelumnya. Dan hasil simulasi akan dicatat dan kemudian akan dilakukan tahap verifikasi.

f. Verifikasi atau Validasi (*Verification and Validation*)

Pada tahap ini, penulis melakukan verifikasi dan validasi terhadap simulasi yang telah dilakukan pada tahapan sebelumnya. Verifikasi dilakukan untuk memastikan ada atau tidaknya kesalahan

(*error*) yang terjadi ketika perhitungan waktu pemrosesan pada Algoritma Boyer Moore, Horspool, ataupun Zhu Takaoka sedangkan validasi dilakukan untuk memastikan kesesuaian simulasi yang dibuat berdasarkan model pengkonsepan dengan formulasi masalah yang sudah dibuat. Jika validasi tidak terpenuhi, maka penulis kembali ke tahapan model pengkonsepan untuk membuat model pengkonsepan baru.

g. Eksperimentasi (*Experimentation*)

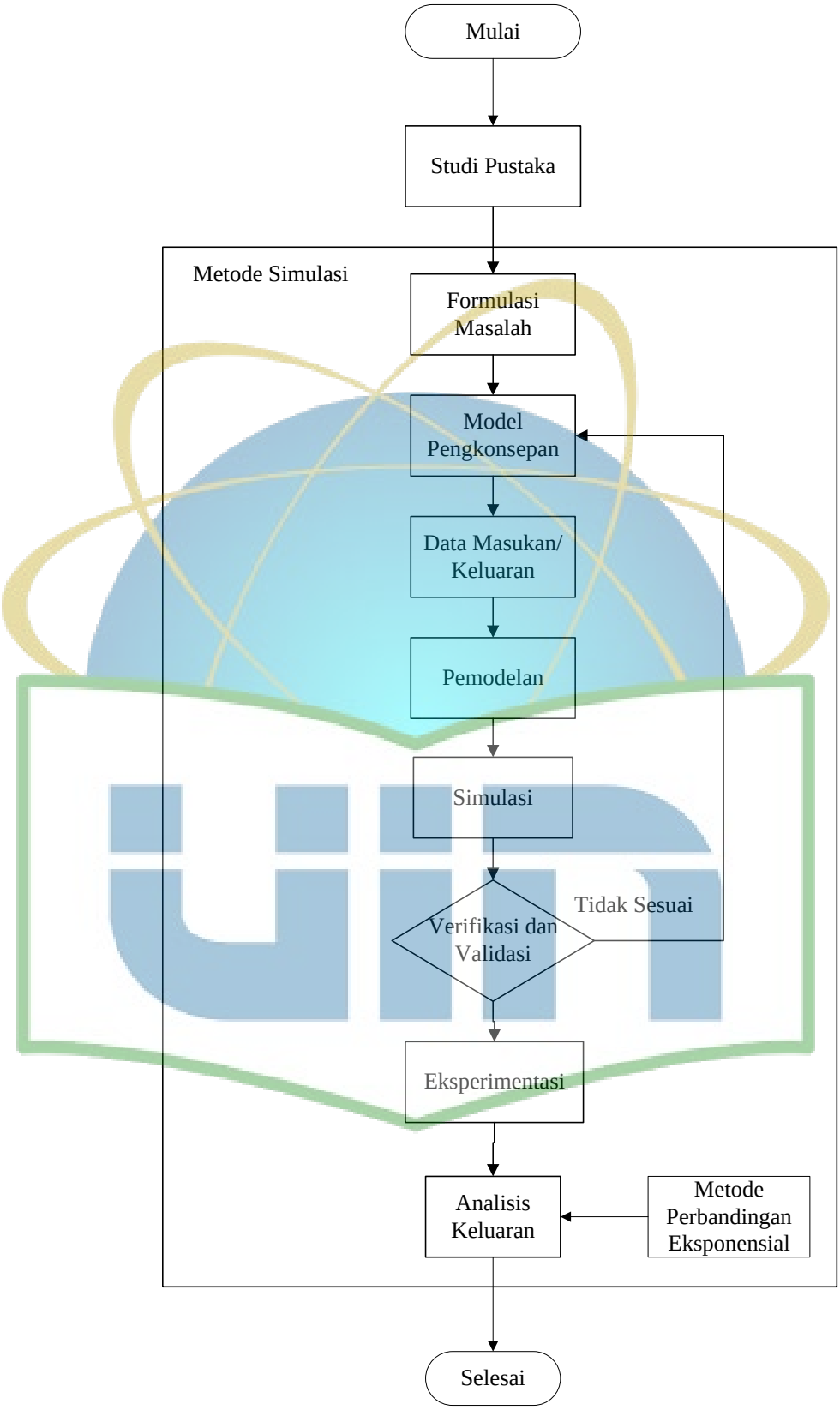
Pada tahapan ini, penulis melakukan eksperimentasi sesuai model skenario yang dibuat pada saat tahapan pemodelan. Tiap model skenario dilakukan percobaan masing-masing sebanyak lima kali pada masing-masing algoritma.

h. Analisis Keluaran (*Output Analysis*)

Pada tahapan terakhir ini, peneliti menganalisa keluaran dan simulasi yang dilakukan pada saat eksperimentasi. Keluaran tersebut direpresentasikan dalam bentuk tabel dan grafik yang menyatakan nilai waktu pemrosesan (*runtime*), memori yang digunakan (*memory consumption*), dan ketepatan (*accuracy*) dari masing-masing algoritma pada saat setiap skenarionya. Pada pembahasan terakhirnya akan membahas analisa keseluruhan dari hasil seluruh skenario simulasi. Selain itu, dalam analisisnya membahas keterhubungan antara nilai-nilai data masukan dan pengaruh dari setiap variabelnya.

3.3. Kerangka Berfikir

Dalam melakukan penyusunan skripsi ini, penulis melakukan tahapan-tahapan kegiatan dengan mengikuti rencana kegiatan yang terdapat pada kerangka berpikir berikut ini:



Gambar 3.1 Kerangka Berpikir

BAB IV
IMPLEMENTASI SIMULASI DAN EKSPERIMEN

4.1. Formulasi Masalah (*Problem Formulation*)

Aplikasi ini merupakan repositori Hadits Bukhori terjemahan Bahasa Indonesia. Pada kasus ini, diperlukan sebuah algoritma *string matching* untuk mencari kata kunci dalam hadits tersebut. Sehingga *user* dapat memperoleh informasi hadits Bukhori dengan mudah dan cepat.

Setelah melakukan studi literarur mengenai algoritma *string matching*, terdapat salah satu solusi yang ditawarkan yaitu Algoritma Boyer Moore. Beberapa jenis Algorima Boyer Moore yang dapat digunakan diantaranya Algoritma Horspool dan Algoritma Zhu Takaoka. Untuk mengetahui kinerja dari algoritma-algoritma tersebut terdapat dua parameter yang dilihat, yaitu *runtime* dan juga *memory consumption*.

Ketepatan (*accuracy*) tidak menjadi parameter yang dibandingkan dalam penelitian ini. Namun, *accuracy* menjadi hal yang terpenting untuk membuktikan hasil pencarian dalam setiap algoritma yang digunakan sesuai dengan sumber data pada repositori penelitian ini.Oleh karena itu, penulis juga menampilkan keakuratan hasil dari pencarian kata pada algoritma yang digunakan.

4.2. Pengkonsepan Model (*Conseptual Model*)

Pada fase ini, penulis membuat konsep yang akan diterapkan pada simulasi yang akan dijalani. Berikut penjelasan dari komponen-komponen aritektur sistem tersebut

1. Server

   Pada simulasi ini server yang digunakan adalah Windows 8.1 Pro dengan spesifikasi memori 4 GB.

2. Algoritma
   - Boyer Moore

     Proses simulasi pada Algoritma Boyer Moore dilakukan melakukan *source code* dengan menggunakan parameter rentang

waktu (*runtime*) dan memori yang digunakan (*memory consumption*). Dengan tambahan akurasi (*accuracy*) sebagai *output*nya.

- Horspool

  Proses simulasi pada Algoritma Horspool dilakukan melakukan *source code* dengan menggunakan parameter rentang waktu (*runtime*) dan memori yang digunakan (*memory consumption*). Dengan tambahan akurasi (*accuracy*) sebagai *output*nya.

- Zhu Takaoka

  Proses simulasi pada Algoritma Zhu Takaoka dilakukan melakukan *source code* dengan menggunakan parameter rentang waktu (*runtime*) dan memori yang digunakan (*memory consumption*). Dengan tambahan akurasi (*accuracy*) sebagai *output*nya.

4.3. Data Masukan/ Keluaran (*Input/ Output Data*)

4.3.1. Data Masukan (*Input*)

Input merupakan atribut-atribut atau nilai-nilai yang digunakan pada proses simulasi ini. Input yang digunakan adalah kata kunci dari sebuah informasi hadits yang ingin dicari.

4.3.2. Data Keluaran (*Output*)

Output dari hasil simulasi ini adalah sebagai berikut:

1. *Runtime*, yaitu waktu yang diperlukan untuk pemrosesan pencarian menggunakan masing-masing algoritma. *Running Time* dari simulasi ini memiliki satuan detik (*second*).
2. *Memory Consumption*, yaitu jumlah besaran memori yang diperlukan dalam pemrosesan pencarian dari masing-masing algoritma yang disimulasikan. Satuan yang digunakan adalah dalam bentuk kilobyte (kb).
3. *Accuracy*, yaitu ketepatan kata yang ingin dicari dalam repository hadits terjemahan Bahasa Indonesia.

4.4. Pemodelan (*Modelling*)

Pada tahap ini, penulis membuat delapan model skenario yang akan dijalankan pada simulasi. Hal ini berdasarkan pada jumlah *string* yang diproses dalam pencarian. Skenario yang dimaksud antara lain:

Tabel 4.1 Tabel Skenario Simulasi

Skenario	*Range* Banyak Karakter	*Output*	Contoh
1	1-10 Karakter	*Runtime, Memory Consumption, Accuracy*	Wahyu
2	10-20 Karakter	*Runtime, Memory Consumption, Accuracy*	Umar Bin Khatab
3	20-30 Karakter	*Runtime, Memory Consumption, Accuracy*	Tanda-Tanda Orang Munafik
4	30-40 Karakter	*Runtime, Memory Consumption, Accuracy*	Semua Perbuatan Tergantug pada Niatnya
5	40-50 Karakter	*Runtime, Memory Consumption, Accuracy*	Diantara tanda-tanda kiamat adalah diangkatnya ilmu
6	50-60 Karakter	*Runtime, Memory Consumption, Accuracy*	Pertanyaan Jibril kepada Nabi SAW tentang iman, islam, ihsan
7	60-70 Karakter	*Runtime, Memory Consumption, Accuracy*	Bagaimana permulaan wahyu yang diturunkan kepada Rasulullah?
8	70-80 Karakter	*Runtime, Memory Consumption, Accuracy*	Seorang muslim adalah orang yang membuat muslim lain selamat dari lidah dan tangannya

Percobaan pada masing-masing skenario dilakukan sebanyak lima kali percobaan pada repositori Hadits Bukhori terjemahan Bahasa Indonesia diasumsikan dengan panjang karakter yang berbeda. *Output* atau hasil simulasi yang dihasilkan yang terdiri dari *runtime* dan *memory consumption*, dan *accuracy*.

4.5. Simulasi (*Simulation*)

Penulis memakai windows 8.1 Pro sebagai sistem operasi yang digunakan untuk seluruh proses simulasi. Dan menggunakan notepad++ dalam proses peng*coding*an. Berikut tahapan-tahapan yang penulis lakukan untuk mempersiapkan proses simulasi:

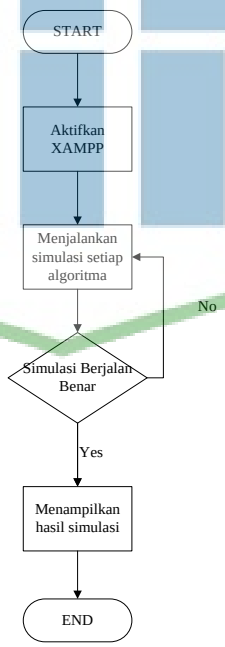
4.5.1. Pembangunan Server

Dalam tahap simulasi penulis menggunakan windows 8.1 Pro sebagai sistem operasi. Secara garis besar, berikut langkah-langkah yang penulis lakukan :

1. Melakukan instalasi appserver, dalam hal ini penulis menggunakan XAMPP
2. Melakukan instalasi editor yang digunakan untuk membuat kode script, dalam hal ini penulis menggunakan notepad++

4.5.2. *Flowchart* Simulasi

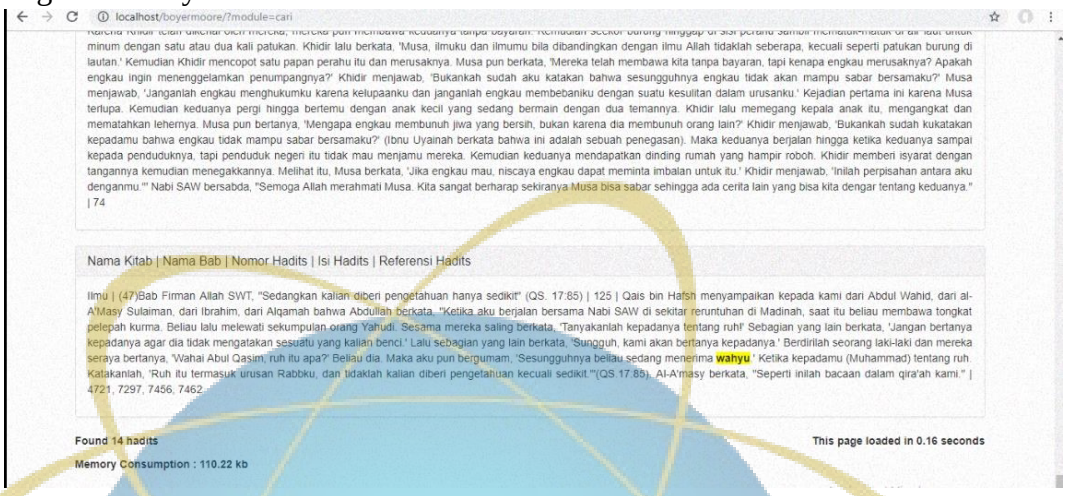
Berikut ini *flowchart* simulasi perbandingan kinerja algoritma Boyer Moore, Horspool, dan Zhu Takaoka yang diimplementasikan pada repositori Hadits terjemahan Bahasa Indonesia:



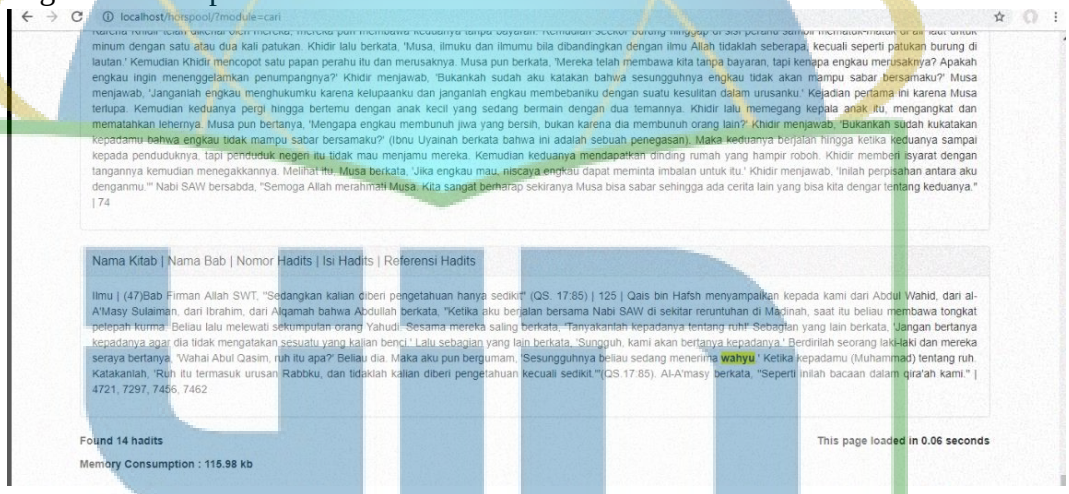
Gambar 4.1 *Flowchart* Simulasi

Berikut contoh simulasi yang dilakukan oleh penulis dengan menggunakan algoritma Boyer Moore, Horspool, dan Zhu Takaoka dalam sekali percobaan :

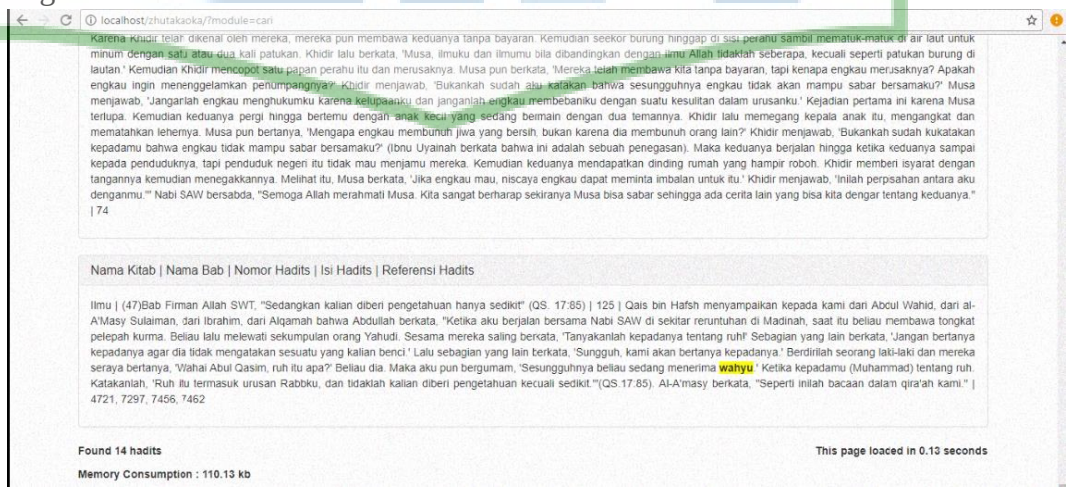
- Algoritma Boyer Moore



- Algoritma Horspool



- Algoritma Zhu Takaoka



4.6. Verifikasi dan Validasi (*Verification and Validation*)

Penjelasan dan pemaparan mengenai verifikasi dan validasi akan dijelaskan pada BAB V skripsi ini, yaitu bab yang membahas hasil pembahasan.

4.7. Eksperimentasi (*Experimentation*)

Penjelasan dan pemaparan mengenai eksperimentasi akan dijelaskan pada BAB V skripsi ini, yaitu bab yang membahas hasil pembahasan.

4.8. Analisa Keluaran (*Output Analysis*)

Penjelasan dan pemaparan mengenai analisis keluaran akan dijelaskan pada BAB V skripsi ini, yaitu bab yang membahas hasil pembahasan.

BAB V
HASIL DAN PEMBAHASAN

5.1. Verifikasi dan Validasi (*Verification and Validation*)

Tahapan ini merupakan tahapan untuk melakukan verifikasi dan validasi dari tahapan-tahap sebelumnya yaitu konseptual model, model sistem dan model simulasi. Pada tahap ini dilakukan koreksi atau perbaikan jika terjadi kesalahan dengan menguji apakah keseluruhan proses simulasi telah berjalan dengan *flowchart* simulasi yang telah dibuat sebelumnya pada tahapan *simulation*. Sedangkan validasi dilakukan dengan menguji apakah keseluruhan proses simulasi telah sesuai dengan ketentuan-ketentuan pada tahapan *conceptual model*, *input output data*, dan *modelling*.

Pengujian akurasi setiap algoritma merupakan pengujian yang dilakukan untuk membandingkan hasil perhitungan dengan cara manual dengan hasil perhitungan algoritma di sistem. Dalam pengujian akurasi hasil algoritma ini hasil yang didapat harus sama untuk keduanya, karena perhitungan manual merupakan acuan dalam menentukan algoritma tersebut benar. Skenario yang digunakan dalam pengujian manual adalah delapan data *sample* yang ada di *corpus* sistem ini.

Tabel 5.1 Tabel Pengujian Algoritma

Skenario	Boyer Moore		Horspool		Zhu Takaoaka		Status
	Manual	Sistem	Manual	Sistem	Manual	Sistem	Sesuai
Wahyu	14	14	14	14	14	14	Sesuai
Umar Bin Khatab	5	5	5	5	5	5	Sesuai
Tanda-tanda orang munafik	2	2	2	2	2	2	Sesuai

Semua perbuatan tergantung dengan niatnya	2	2	2	2	2	2	Sesuai
Di antara tanda-tanda kiamat adalah diangkatnya ilmu	1	1	1	1	1	1	Sesuai
Pertanyaan Jibril kepada Nabi SAW tentang Iman, Islam, Ihsan	1	1	1	1	1	1	Sesuai
Bagaimana permulaan wahyu yang diturukan kepada Rasulullah SAW?	7	7	7	7	7	7	Sesuai
Seorang muslim adalah orang yang membuat	1	1	1	1	1	1	Sesuai

muslim lain selamat dari lidah dan tangannya							

Pengujian Akurasi

Akurasi dihitung dari jumlah yang tepat dibagi dengan jumlah data (Jiawei Han, 2012).

$$Tingkat\ Akurasi = \frac{\Sigma \text{ data uji benar}}{\Sigma \text{total data uji}}$$

$$Akurasi(\%) = \frac{\Sigma \text{ data uji benar}}{\Sigma \text{total data uji}} x\ 100\%$$

$$Akurasi(\%) = \frac{8}{8} x\ 100\% = 100\%$$

5.2. Eksperimentasi (*Experimentation*)

Proses eksperimentasi adalah proses yang dilakukan untuk membandingkan hasil dari *simulator*. Fase ini dimulai dengan desain eksperimen sesuai dengan yang penulis susun pada tahap simulasi, dan dengan teknik tertentu berdasar pada beberapa faktor yang menguji nilai parameter untuk melakukan analisa pada *output* hasil dari proses simulasi. Pada penulisan ini penulis membandingkan perbedaan yang terjadi jika kata yang ada pada proses simulasi tersebut diubah, Penulis menggunakan parameter-parameter, yaitu : *runtime* dan *memory consumption* diujikan ke tiga jenis algoritma, yaitu Algoritma Boyer Moore, Algoritma Horspool, dan Algoritma Zhu Takaoka. Proses analisis *output* akan dijelaskan pada poin selanjutnya.

5.3. Analisis Keluaran (*Output Analysis*)

5.3.1. Skenario 1

5.3.1.1. Skenario Boyer Moore

Tabel 5.2 Hasil Skenario 1 Boyer Moore

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,15	0,16	0,16	0,15	0,17	0,158
Memory Consumption (kb)	110,22	110,22	110,22	110,22	110,22	110,22
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 1 pada Algoritma Boyer Moore. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan satu dan empat, yaitu 0,15 s serta rata-rata sebesar 0,158 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,22 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.1.2. Skenario Horspool

Tabel 5.3 Hasil Skenario 1 Horspool

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,08	0,10	0,08	0,10	0,06	0,084
Memory Consumption (kb)	115,98	115,98	115,98	115,98	115,98	115,98
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 1 pada Algoritma Horspool. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan lima, yaiti 0,06 s serta rata-rata sebesar 0,084 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 115,98 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.1.3. Skenario Zhu Takaoka

Tabel 5.4 Hasil Skenario 1 Zhu Takaoka

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,13	0,12	0,12	0,15	0,17	0,138
Memory Consumption (kb)	110,13	110,13	110,13	110,13	110,13	110,13
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 1 pada Algoritma Zhu Takaoka. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan dua dan tiga, yaitu 0,15 s serta rata-rata sebesar 0,138 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,13 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.2. Skenario 2

5.3.2.1. Skenario Boyer Moore

Tabel 5.5 Hasil Skenario 2 Boyer Moore

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,06	0,13	0,23	0,05	0,12	0,118
Memory Consumption (kb)	110,19	110,19	110.19	110,19	110,19	110,19
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 2 pada Algoritma Boyer Moore. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan keempat, yaitu 0,5 s serta rata-rata sebesar 0,118 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,19 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.2.2. Skenario Horspool

Tabel 5.6 Hasil Skenario 2 Horspool

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,11	0,06	0,16	0,05	0,05	0,086
Memory Consumption (kb)	115,95	115,95	115,95	115,95	115,95	115,95
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 2 pada Algoritma Horspool. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan keempat dan kelima, yaitu 0,05 s serta rata-rata sebesar 0,086 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 115,95 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.2.3. Skenario Zhu Takaoka

Tabel 5.7 Hasil Skenario 2 Zhu Takaoka

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,05	0,06	0,05	0,05	0,05	0,052
Memory Consumption (kb)	110.1	110,1	110,1	110,01	110,01	110,01
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 2 pada Algoritma Zhu Takaoka. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* terlama terdapat pada percobaan kedua, yaitu 0,06 s serta rata-rata sebesar 0,052 s. Untuk percobaan lainnya, masing-masing nilai *runtime* sebesar 0,05 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,01 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.3. Skenario 3

5.3.3.1. Skenario Boyer Moore

Tabel 5.8 Hasil Skenario 3 Boyer Moore

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,19	0,09	0,13	0,03	0,13	0,114
Memory Consumption (kb)	110,14	110,14	110,14	110,14	110,14	110,14
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 3 pada Algoritma Boyer Moore. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan keempat, yaitu 0,03 s serta rata-rata sebesar 0,114 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,14 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.3.2. Skenario Horspool

Tabel 5.9 Hasil Skenario 3 Horspool

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,08	0,13	0,04	0,15	0,09	0,098
Memory Consumption (kb)	115,91	115,91	115,91	115,91	115,91	115,91
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 3 pada Algoritma Horspool. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan ketiga, yaitu 0,04 s serta rata-rata sebesar 0,098 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 115,91 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.3.3. Skenario Zhu Takaoka

Tabel 5.10 Hasil Skenario 3 Zhu Takaoka

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,10	0,16	0,12	0,05	0,17	0,12
Memory Consumption (kb)	110,05	110,05	110,05	110,05	110,05	110,05
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 3 pada Algoritma Zhu Takaoka. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan keempat, yaitu 0,05 s serta rata-rata sebesar 0,12 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,05 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.4. Skenario 4

5.3.4.1. Skenario Boyer Moore

Tabel 5.11 Hasil Skenario 4 Boyer Moore

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,16	0,10	0,12	0,08	0,10	0,112
Memory Consumption (kb)	110,13	110,13	110,13	110,13	110,13	110,13
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 4 pada Algoritma Boyer Moore. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan keempat, yaitu 0,08 s serta rata-rata sebesar 0,112 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,13 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.4.2. Skenario Horspool

Tabel 5.12 Hasil Skenario 4 Horspool

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,17	0,04	0,18	0,04	0,10	0,106
Memory Consumption (kb)	115,88	115,88	115,88	115,88	115,88	115,88
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 4 pada Algoritma Horspool. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan kedua dan keempat, yaitu 0,04 s serta rata-rata sebesar 0,106 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 115,88 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.4.3. Skenario Zhu Takaoka

Tabel 5.13 Hasil Skenario 4 Zhu Takaoka

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,04	0,16	0,15	0,15	0,04	0,108
Memory Consumption (kb)	110,04	110,04	110,04	110,04	110,04	110,04
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 4 pada Algoritma Zhu Takaoka. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan pertama dan kelima, yaitu 0,04 s serta rata-rata sebesar 0,108 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,04 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.5. Skenario 5

5.3.5.1. Skenario Boyer Moore

Tabel 5.14 Hasil Skenario 5 Boyer Moore

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,10	0,12	0,11	0,12	0,08	0,106
Memory Consumption (kb)	110,06	110,06	110,06	110,06	110,06	110,06
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 5 pada Algoritma Boyer Moore. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan kelima, yaitu 0,08 s serta rata-rata sebesar 0,106 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,06 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.5.2. Skenario Horspool

Tabel 5.15 Hasil Skenario 5 Horspool

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,11	0,14	0,12	0,11	0,05	0,106
Memory Consumption (kb)	115,84	115,84	115,84	115,84	115,84	115,84
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 5 pada Algoritma Horspool. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan kelima, yaitu 0,05 s serta rata-rata sebesar 0,106 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 115,84 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.5.3. Skenario Zhu Takaoka

Tabel 5.16 Hasil Skenario 5 Zhu Takaoka

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,03	0,12	0,15	0,08	0,04	0,084
Memory Consumption (kb)	109,97	109,97	109,97	109,97	109,97	109,97
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 5 pada Algoritma Zhu Takaoka. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan pertama, yaitu 0,03 s serta rata-rata sebesar 0,084 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 109,97 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.6. Skenario 6

5.3.6.1. Skenario Boyer Moore

Tabel 5.17 Hasil Skenario 6 Boyer Moore

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,08	0,16	0,04	0,15	0,04	0,094
Memory Consumption (kb)	110,05	110,05	110,05	110,05	110,05	110,05
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 6 pada Algoritma Boyer Moore. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan ketiga dan kelima, yaitu 0,04 s serta rata-rata sebesar 0,094 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,05 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.6.2. Skenario Horspool

Tabel 5.18 Hasil Skenario 6 Horspool

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,17	0,05	0,14	0,10	0,11	0,114
Memory Consumption (kb)	115,8	115,8	115,8	115,8	115,8	115,8
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 6 pada Algoritma Horspool. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan kedua, yaitu 0,05 s serta rata-rata sebesar 0,114 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 115,8 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.6.3. Skenario Zhu Takaoka

Tabel 5.19 Hasil Skenario 6 Zhu Takaoka

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,09	0,17	0,12	0,16	0,14	0,136
Memory Consumption (kb)	109,97	109,97	109,97	109,97	109,97	109,97
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 6 pada Algoritma Zhu Takaoka. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan pertama, yaitu 0,09 s serta rata-rata sebesar 0,136 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 109,97 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.7. Skenario 7

5.3.7.1. Skenario Boyer Moore

Tabel 5.20 Hasil Skenario 7 Boyer Moore

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,06	0,15	0,07	0,07	0,10	0,09
Memory Consumption (kb)	110,02	110,02	110,02	110,02	110,02	110,02
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 7 pada Algoritma Boyer Moore. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan pertama, yaitu 0,06 s serta rata-rata sebesar 0,09 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 110,02 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.7.2. Skenario Horspool

Tabel 5.21 Hasil Skenario 7 Horspool

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,17	0,20	0,05	0,06	0,11	0,118
Memory Consumption (kb)	115,78	115,78	115,78	115,78	115,78	115,78
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 7 pada Algoritma Horspool. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan ketiga, yaitu 0,05 s serta rata-rata sebesar 0,118 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 115,78 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.7.3. Skenario Zhu Takaoka

Tabel 5.22 Hasil Skenario 7 Zhu Takaoka

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,17	0,09	0,05	0,06	0,18	0,11
Memory Consumption (kb)	109,94	109,94	109,94	109,94	109,94	109,94
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 7 pada Algoritma Zhu Takaoka. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan ketiga, yaitu 0,05 s serta rata-rata sebesar 0,11 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 109,94 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.8. Skenario 8

5.3.8.1. Skenario Boyer Moore

Tabel 5.23 Hasil Skenario 8 Boyer Moore

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,05	0,11	0,05	0,05	0,10	0,072
Memory Consumption (kb)	109,98	109,98	109,98	109,98	109,98	109,98
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 8 pada Algoritma Boyer Moore. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* terlama terdapat pada percobaan kedua, yaitu 0,11 s serta rata-rata sebesar 0,072 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 109,98 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.8.2. Skenario Horspool

Tabel 5.24 Hasil Skenario 8 Horspool

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,12	0,11	0,19	0,05	0,13	0,120
Memory Consumption (kb)	115,74	115,74	115,74	115,74	115,74	115,74
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 8 pada Horspool. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan keempat, yaitu 0,05 s serta rata-rata sebesar 0,120 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 115,74 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.3.8.3. Skenario Zhu Takaoka

Tabel 5.25 Hasil Skenario 8 Zhu Takaoka

Output	Percobaan Ke -					Rata-Rata
	1	2	3	4	5	
Runtime (s)	0,16	0,08	0,05	0,18	0,05	0,104
Memory Consumption (kb)	109,89	109,89	109,89	109,89	109,89	109,89
Accuracy (%)	100	100	100	100	100	100

Tabel di atas menunjukkan hasil skenario 8 pada Algoritma Zhu Takaoka. Percobaan dilakukan sebanyak lima kali. Hasil *runtime* tercepat terdapat pada percobaan ketiga dan kelima, yaitu 0,05 s serta rata-rata sebesar 0,104 s. *Memory consumption* pada percobaan ini tidak berubah, yaitu sebesar 109,89 kb dari percobaan satu sampai percobaan lima. Sedangkan, untuk nilai dari *accuracy* nya sebesar 100%.

5.4. Hasil Perbandingan

Setelah setiap skenario dijalankan dan data-data *output* pada setiap percobaan didapatkan, maka hasil *output* tersebut digunakan untuk melakukan analisis kinerja masing-masing algoritma berdasarkan parameter *runtime* tercepat dengan melihat waktu terkecil serta memperhatikan *memory consumption* dengan nilai-nilai terkecil.

Output analisis kinerja dijabarkan dengan menggunakan tabel dari setiap skenario yang sudah dilakukan sebanyak delapan skenario hingga sampai hasil akhir penelitian analisis kinerja algoritma Boyer Moore, Horspool, dan Zhu Takaoka.

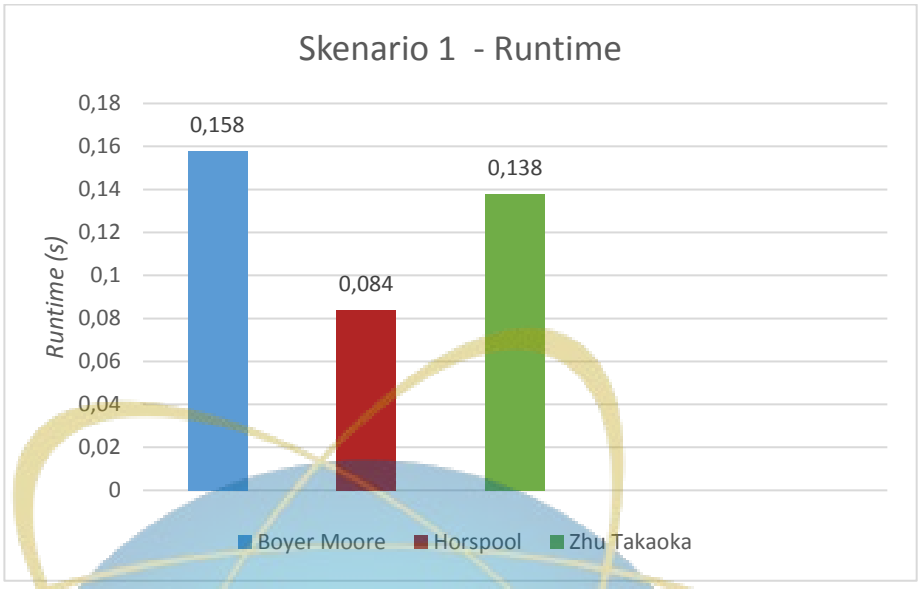
5.4.1. Skenario 1

Hasil dari masing-masing percobaan pada skenario 1 dilakukan untuk menghitung rata-rata. Berikut ini rata-rata hasil dari skenario 1 pada algoritma Boyer Moore, Horspool, dan Zhu Takaoka :

Tabel 5.26 Hasil Perbandingan Skenario 1

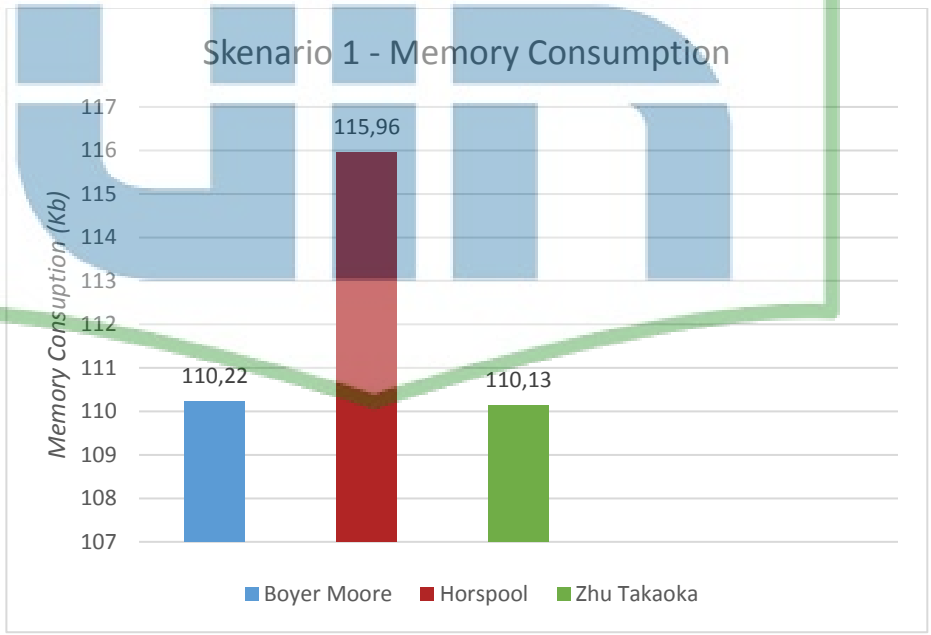
Output	Boyer Moore	Horspool	Zhu Takoka
Runtime (s)	0,158	0,084	0,138
Memory Consumption (kb)	110,22	115,96	110,13
Accuracy (%)	100	100	100

Pada tabel di atas menunjukkan hasil simulasi pada skenario 1 yang dilakukan terhadap tiga algoritma. Boyer Moore memiliki nilai rata-rata *runtime* 0,158 s dan nilai rata-rata *memory consumption* 110,22 kb. Horspool memiliki nilai rata-rata runtime tercepat dengan nilai 0,084 s dan memilki nilai rata-rata *memory consumption* terbesar yaitu 115,96. Zhu Takaoka memilki nilai rata-rata *runtimw* 0,138 s dan memiliki nilai rata-rata *memory consumption* terkecil yaitu 110,13 s. Untuk *accuracy*, masing-masing memilki nilai sebesar 100 %.



Gambar 5.1 Hasil Perbandingan Skenario 1 *Runtime*

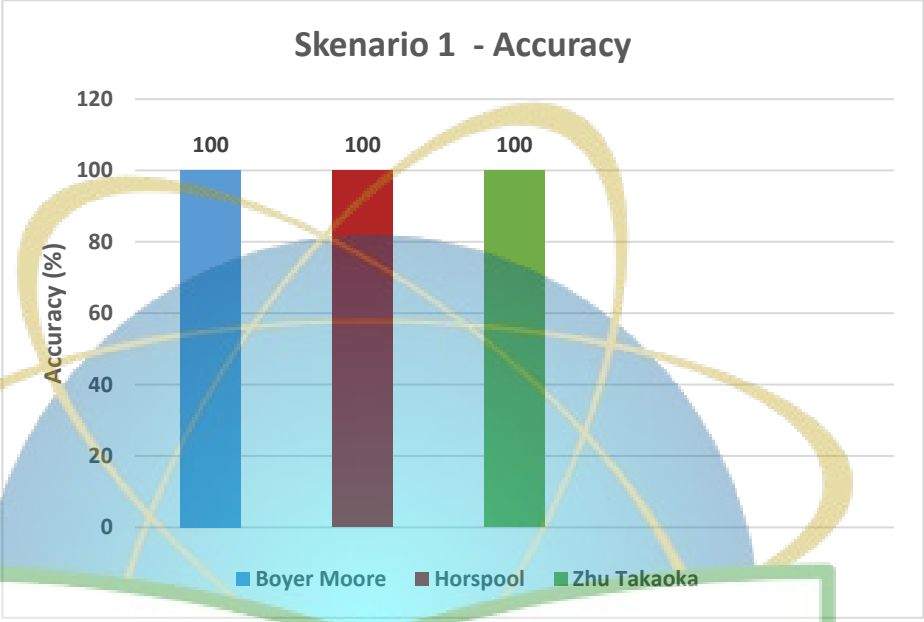
Grafik di atas menunjukkan perbandingan nilai rata-rata *runtime* untuk masing-masing algoritma. Semakin kecil nilai *runtime* pada proses di suatu algoritma maka hasil ini semakin bagus. Pada hasil skenario 1 , algoritma Horspool memiliki nilai *runtime* yang terbaik.



Gambar 5.2 Hasil Perbandingan Skenario 1 *Memory Consumption*

Grafik di atas menunjukkan perbandingan nilai rata-rata *memory consumption* untuk masing-masing algoritma. Semakin kecil nilai *memory*

consumption pada proses pencarian di suatu algoritma maka hal ini bisa dikatakan semakin bagus. Sebab akan membutuhkan *resource* yang lebih kecil. Pada skenario 1, algoritma Zhu Takaoka memilki *memory consumption* yang terbaik.



Gambar 5.3 Hasil Perbandingan Skenario 1 *Accuracy*

Grafik di atas menunjukkan nilai *accuracy* masing-masing algoritma. Nilai *accuracy* merupakan sebuah pencocokan banyaknya *output* kata dengan perhitungan manual dan yang ada di sistem. Pada skenario 1, setiap algoritma memiliki nilai *accuracy* sebesar 100 %.

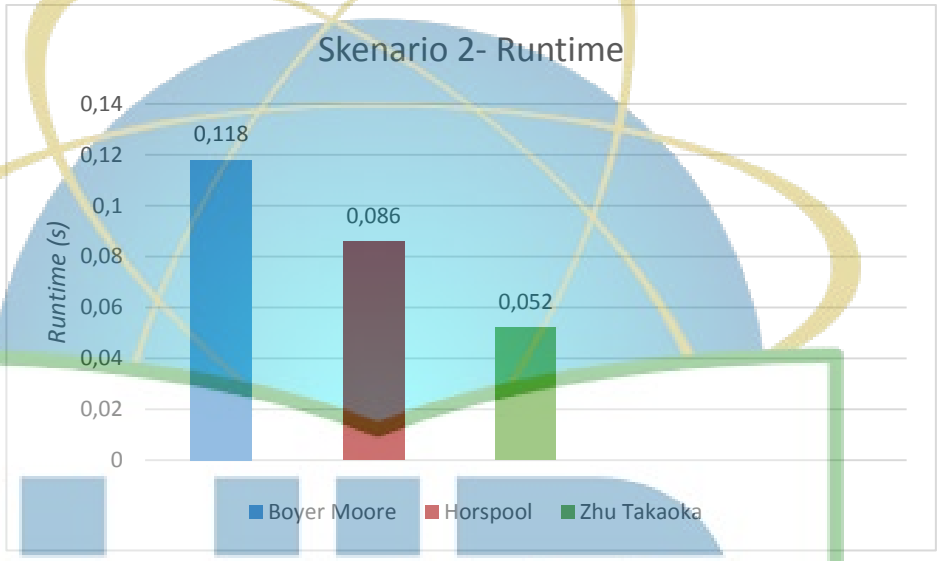
5.4.2. Skenario 2

Hasil dari masing-masing percobaan pada skenario 2 dilakukan untuk menghitung rata-rata. Berikut ini rata-rata hasil dari skenario 2 pada algoritma Boyer Moore, Horspool, dan Zhu Takaoka :

Tabel 5.27 Hasil Perbandingan Skenario 2

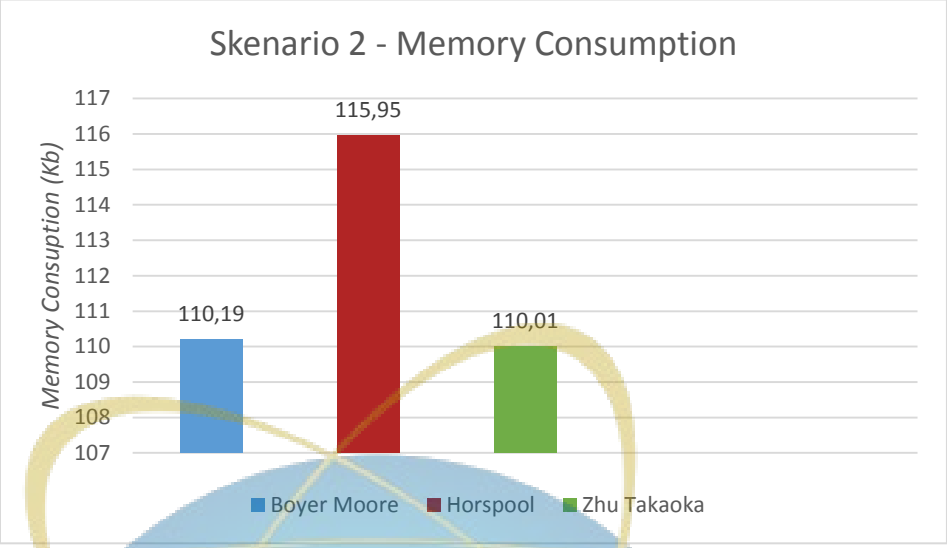
Output	Boyer Moore	Horspool	Zhu Takoka
Runtime (s)	0,118	0,086	0,052
Memory Consumption (kb)	110,19	115,95	110,01
Accuracy (%)	100	100	100

Pada tabel di atas menunjukkan hasil simulasi pada skenario 2 yang dilakukan terhadap tiga algoritma. Boyer Moore memiliki nilai rata-rata *runtime* 0,118 s dan nilai rata-rata *memory consumption* 110,19 kb. Horspool memiliki nilai rata-rata runtime dengan nilai 0,086 s dan memilki nilai rata-rata *memory consumption* terbesar yaitu 115,95 kb. Zhu Takaoka memilki nilai rata-rata *runtime* 0,052 s dan memiliki nilai rata-rata *memory consumption* terkecil yaitu 110,13 kb. Untuk *accuracy*, masing-masing memilki nilai sebesar 100 %.



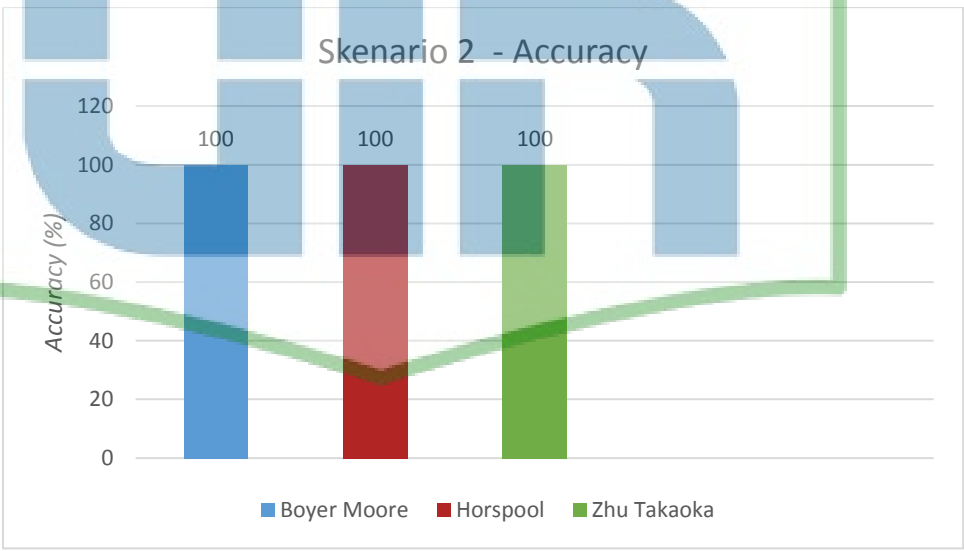
Gambar 5.4 Hasil Perbandingan Skenario 2 *Runtime*

Grafik di atas menunjukkan perbandingan nilai rata-rata *runtime* untuk masing-masing algoritma. Semakin kecil nilai *runtime* pada proses di suatu algoritma maka hasil ini semakin bagus. Pada hasil skenario 2, algoritma Zhu Takaoka memiliki nilai *runtime* yang terbaik.



Gambar 5.5 Hasil Perbandingan Skenario 2 *Memory Consumption*

Grafik di atas menunjukkan perbandingan nilai rata-rata *memory consumption* untuk masing-masing algoritma. Semakin kecil nilai *memory consumption* pada proses pencarian di suatu algoritma maka hal ini bisa dikatakan semakin bagus. Sebab akan membutuhkan *resource* yang lebih kecil. Pada skenario 2, algoritma Zhu Takaoka memilki *memory consumption* yang terbaik.



Gambar 5.6 Hasil Perbandingan Skenario 2 *Accuracy*

Grafik di atas menunjukkan nilai *accuracy* masing-masing algoritma. Nilai *accuracy* merupakan sebuah pencocokan banyaknya

output kata dengan perhitungan manual dan yang ada di sistem. Pada skenario 2, setiap algoritma memiliki nilai *accuracy* sebesar 100 %.

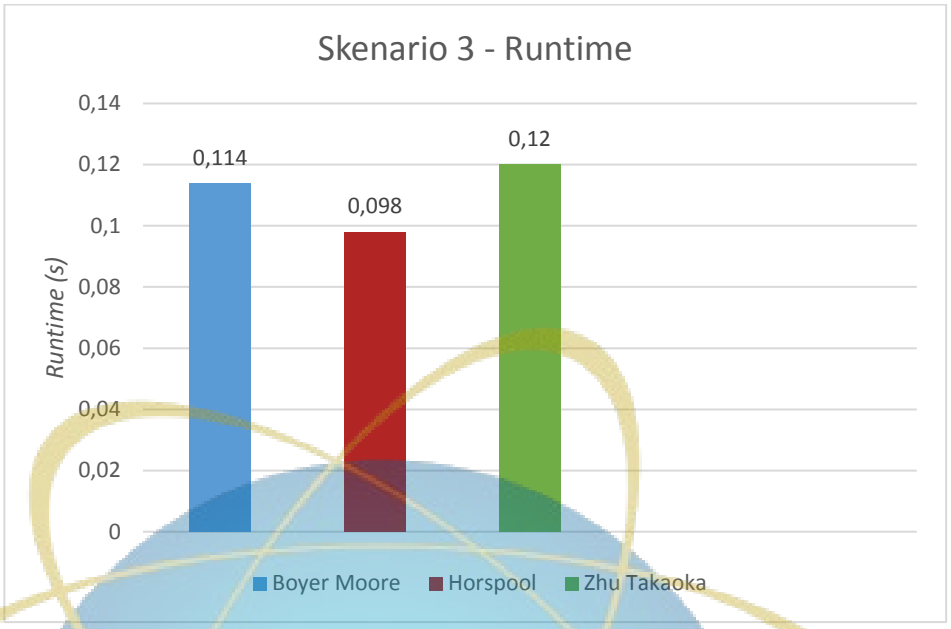
5.4.3. Skenario 3

Hasil dari masing-masing percobaan pada skenario 3 dilakukan untuk menghitung rata-rata. Berikut ini rata-rata hasil dari skenario 3 pada algoritma Boyer Moore, Horspool, dan Zhu Takaoka :

Tabel 5.28 Hasil Perbandingan Skenario 3

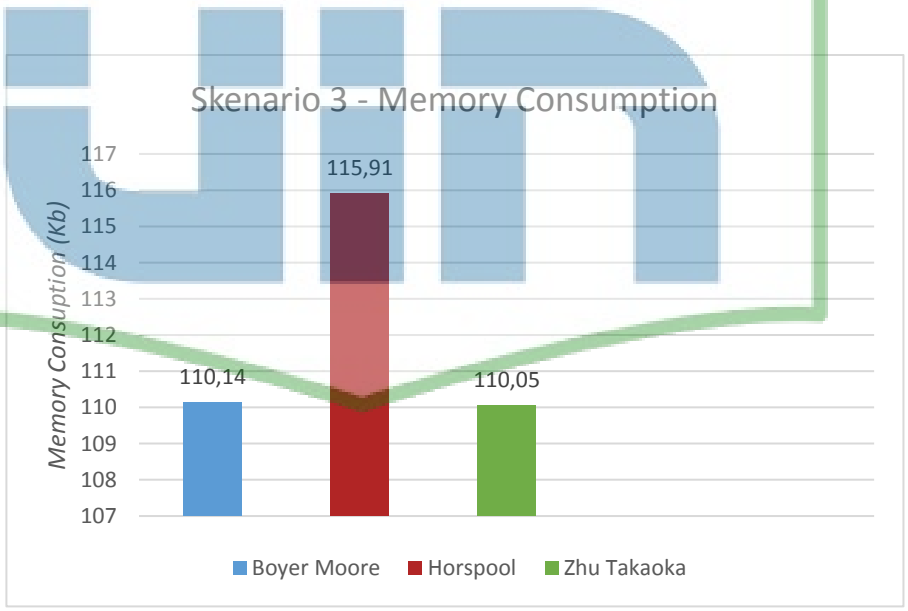
Output	Boyer Moore	Horspool	Zhu Takoka
Runtime (s)	0,114	0,098	0,12
Memory Consumption (kb)	110,14	115,91	110,05
Accuracy (%)	100	100	100

Pada tabel di atas menunjukkan hasil simulasi pada skenario 3 yang dilakukan terhadap tiga algoritma. Boyer Moore memiliki nilai rata-rata *runtime* 0,114 s dan nilai rata-rata *memory consumption* 110,14 kb. Horspool memiliki nilai rata-rata *runtime* tercepat dengan nilai 0,098 s dan memilki nilai rata-rata *memory consumption* terbesar yaitu 115,91 kb. Zhu Takaoka memilki nilai rata-rata *run timw* 0,12 s dan memiliki nilai rata-rata *memory consumption* terkecil yaitu 110,05 kb. Untuk *accuracy*, masing-masing memilki nilai sebesar 100 %.



Gambar 5.7 Hasil Perbandingan Skenario 3 *Runtime*

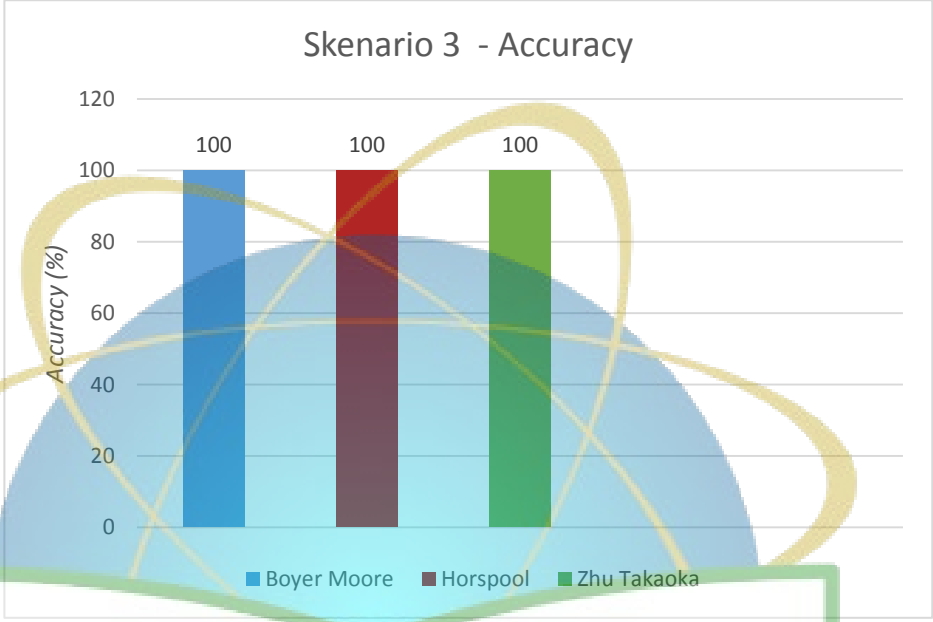
Grafik di atas menunjukkan perbandingan nilai rata-rata *runtime* untuk masing-masing algoritma. Semakin kecil nilai *runtime* pada proses di suatu algoritma maka hasil ini semakin bagus. Pada hasil skenario 3 , algoritma Horspool memiliki nilai *runtime* yang terbaik.



Gambar 5.8 Hasil Perbandingan Skenario 3 *Memory Consumption*

Grafik di atas menunjukkan perbandingan nilai rata-rata *memory consumption* untuk masing-masing algoritma. Semakin kecil nilai *memory*

consumption pada proses pencarian di suatu algoritma maka hal ini bisa dikatakan semakin bagus. Sebab akan membutuhkan *resource* yang lebih kecil. Pada skenario 3, algoritma Zhu Takaoka memilki *memory consumption* yang terbaik.



Gambar 5.9 Hasil Perbandingan Skenario 3 *Accuracy*

Grafik di atas menunjukkan nilai *accuracy* masing-masing algoritma. Nilai *accuracy* merupakan sebuah pencocokan banyaknya *output* kata dengan perhitungan manual dan yang ada di sistem. Pada skenario 3, setiap algoritma memiliki nilai *accuracy* sebesar 100 %.

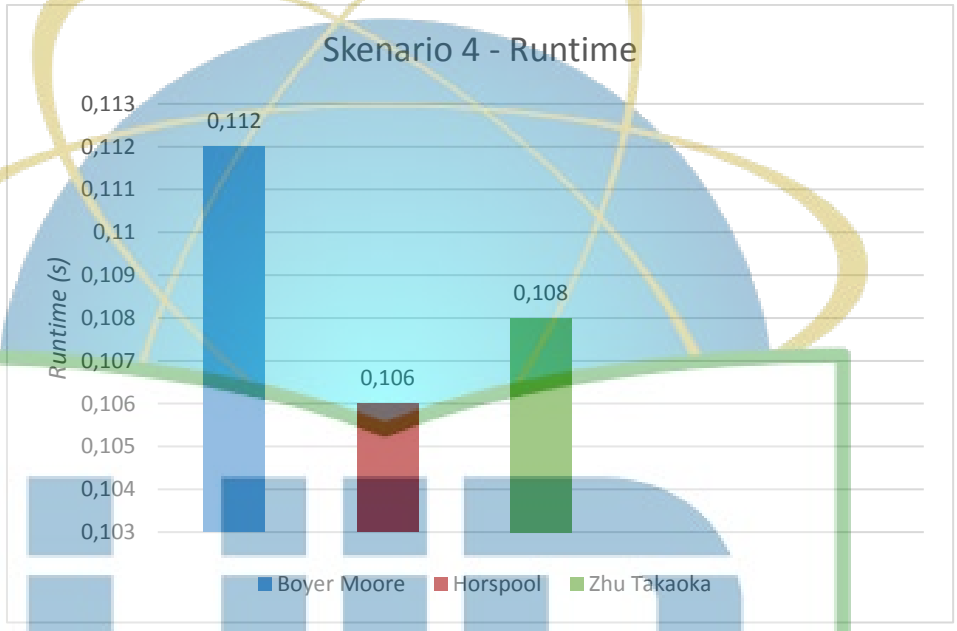
5.4.4. Skenario 4

Hasil dari masing-masing percobaan pada skenario 4 dilakukan untuk menghitung rata-rata. Berikut ini rata-rata hasil dari skenario 4 pada algoritma Boyer Moore, Horspool, dan Zhu Takaoka :

Tabel 5.29 Hasil Perbandingan Skenario 4

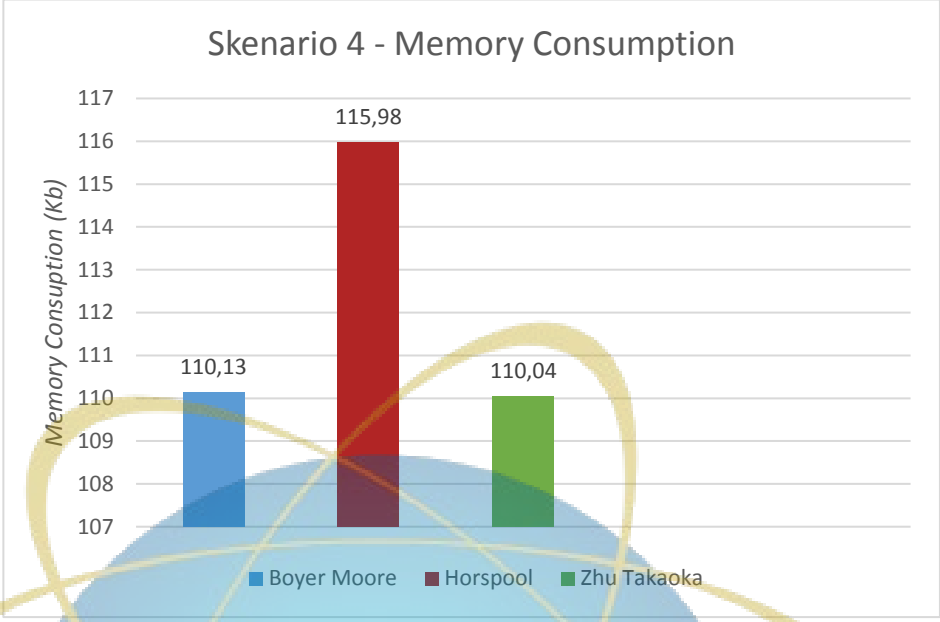
Output	Boyer Moore	Horspool	Zhu Takoka
Runtime (s)	0,112	0,106	0,108
Memory Consumption (kb)	110,13	115,98	110,04
Accuracy (%)	100	100	100

Pada tabel di atas menunjukkan hasil simulasi pada skenario 4 yang dilakukan terhadap tiga algoritma. Boyer Moore memiliki nilai rata-rata *runtime* 0,112 s dan nilai rata-rata *memory consumption* 110,13 kb. Horspool memiliki nilai rata-rata *runtime* tercepat dengan nilai 0,106 s dan memilki nilai rata-rata *memory consumption* terbesar yaitu 115,98 kb. Zhu Takaoka memilki nilai rata-rata *runtime* 0,108 s dan memiliki nilai rata-rata *memory consumption* terkecil yaitu 110,04 kb. Untuk *accuracy*, masing-masing memilki nilai sebesar 100 %.



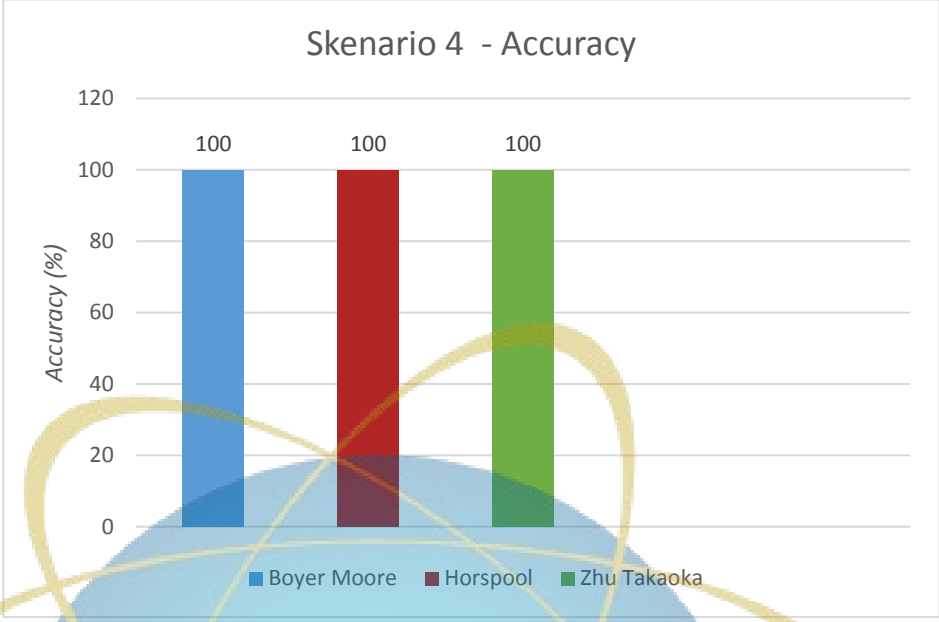
Gambar 5.10 Hasil Perbandingan Skenario 4 *Runtime*

Grafik di atas menunjukkan perbandingan nilai rata-rata *runtime* untuk masing-masing algoritma. Semakin kecil nilai *runtime* pada proses di suatu algoritma maka hasil ini semakin bagus. Pada hasil skenario 4 , algoritma Horspool memiliki nilai *runtime* yang terbaik.



Gambar 5.11 Hasil Perbandingan Skenario 4 *Memory Consumption*

Grafik di atas menunjukkan perbandingan nilai rata-rata *memory consumption* untuk masing-masing algoritma. Semakin kecil nilai *memory consumption* pada proses pencarian di suatu algoritma maka hal ini bisa dikatakan semakin bagus. Sebab akan membutuhkan *resource* yang lebih kecil. Pada skenario 4, algoritma Zhu Takaoka memilki *memory consumption* yang terbaik.



Gambar 5.12 Hasil Perbandingan Skenario 4 *Accuracy*

Grafik di atas menunjukkan nilai *accuracy* masing-masing algoritma. Nilai *accuracy* merupakan sebuah pencocokan banyaknya *output* kata dengan perhitungan manual dan yang ada di sistem. Pada skenario 4, setiap algoritma memiliki nilai *accuracy* sebesar 100 %.

5.4.5. Skenario 5

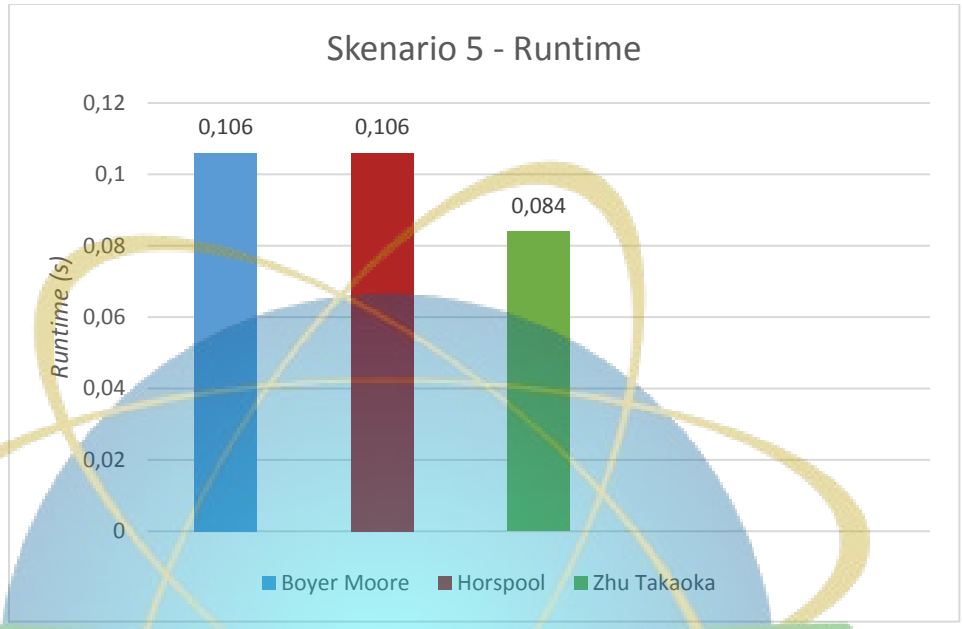
Hasil dari masing-masing percobaan pada skenario 5 dilakukan untuk menghitung rata-rata. Berikut ini rata-rata hasil dari skenario 5 pada algoritma Boyer Moore, Horspool, dan Zhu Takaoka :

Tabel 5.30 Hasil Perbandingan Skenario 5

Output	Boyer Moore	Horspool	Zhu Takoka
Runtime (s)	0,106	0,106	0,084
Memory Consumption (kb)	110,13	115,98	110,04
Accuracy (%)	100	100	100

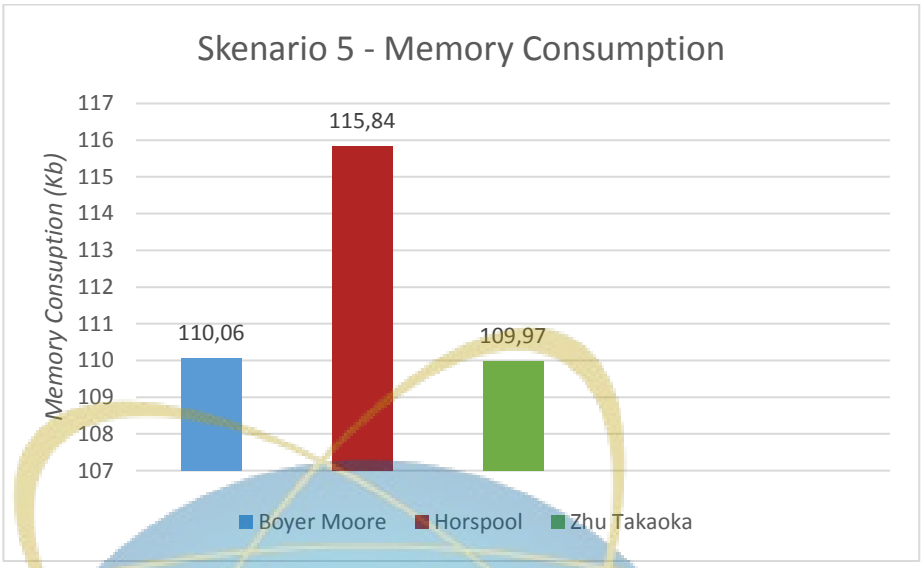
Pada tabel di atas menunjukkan hasil simulasi pada skenario 5 yang dilakukan terhadap tiga algoritma. Boyer Moore memiliki nilai rata-rata *runtime* 0,106 s dan nilai rata-rata *memory consumption* 110,13 kb. Horspool memiliki nilai rata-rata *runtime* dengan nilai 0,106 s dan memilki nilai rata-rata *memory consumption* terbesar yaitu 115,98 kb. Zhu Takaoka

memilki nilai rata-rata *runtime* 0,084 s dan memiliki nilai rata-rata *memory consumption* terkecil yaitu 110,04 kb. Untuk *accuracy*, masing-masing memilki nilai sebesar 100 %.



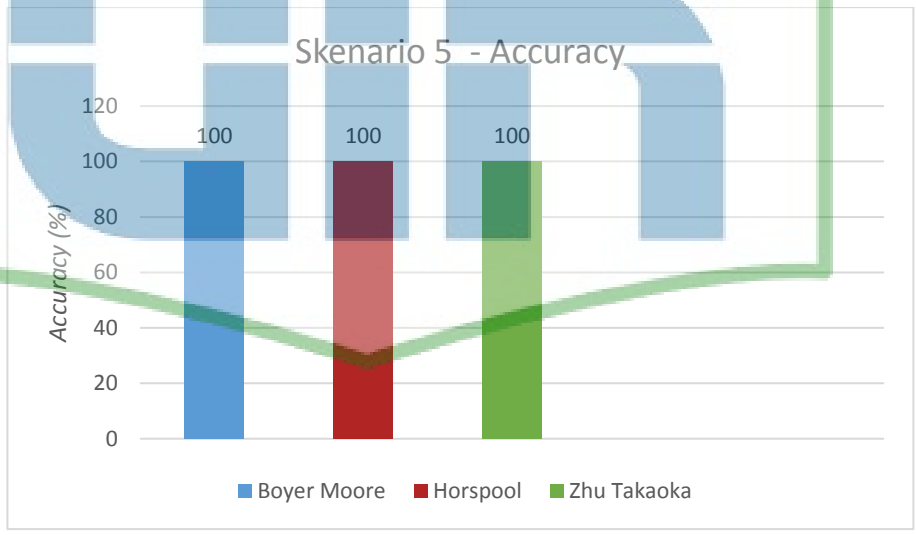
Gambar 5.13 Hasil Perbandingan Skenario 5 *Runtime*

Grafik di atas menunjukkan perbandingan nilai rata-rata *runtime* untuk masing-masing algoritma. Semakin kecil nilai *runtime* pada proses di suatu algoritma maka hasil ini semakin bagus. Pada hasil skenario 5 , algoritma Zhu Takaoka memiliki nilai *runtime* yang terbaik.



Gambar 5.14 Hasil Perbandingan Skenario 5 *Memory Consumption*

Grafik di atas menunjukkan perbandingan nilai rata-rata *memory consumption* untuk masing-masing algoritma. Semakin kecil nilai *memory consumption* pada proses pencarian di suatu algoritma maka hal ini bisa dikatakan semakin bagus. Sebab akan membutuhkan *resource* yang lebih kecil. Pada skenario 5, algoritma Zhu Takaoka memilki *memory consumption* yang terbaik.



Gambar 5.15 Hasil Perbandingan Skenario 5 *Accuracy*

Grafik di atas menunjukkan nilai *accuracy* masing-masing algoritma. Nilai *accuracy* merupakan sebuah pencocokan banyaknya

output kata dengan perhitungan manual dan yang ada di sistem. Pada skenario 5, setiap algoritma memiliki nilai *accuracy* sebesar 100 %.

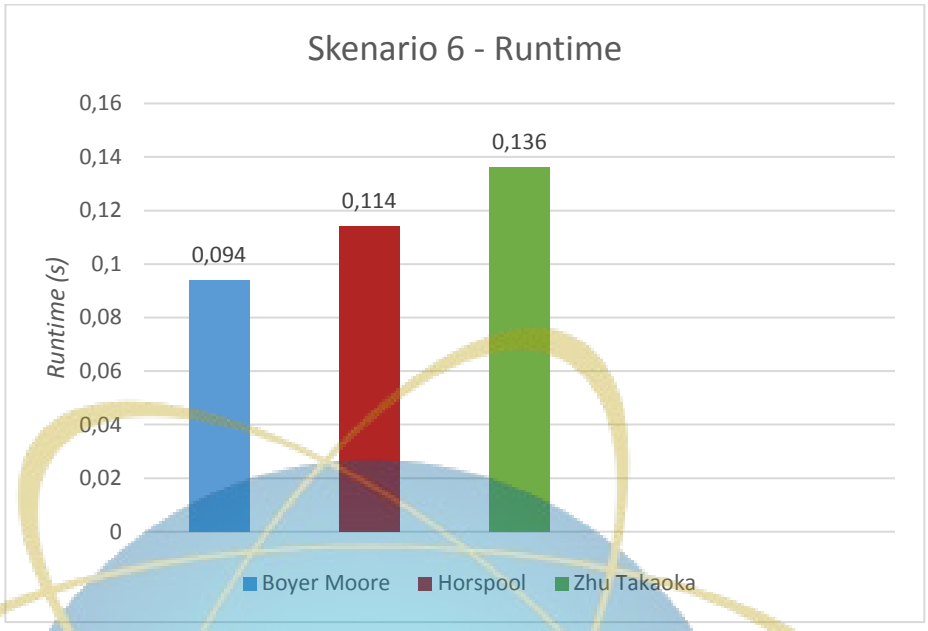
5.4.6. Skenario 6

Hasil dari masing-masing percobaan pada skenario 6 dilakukan untuk menghitung rata-rata. Berikut ini rata-rata hasil dari skenario 6 pada algoritma Boyer Moore, Horspool, dan Zhu Takaoka :

Tabel 5.31 Hasil Perbandingan Skenario 6

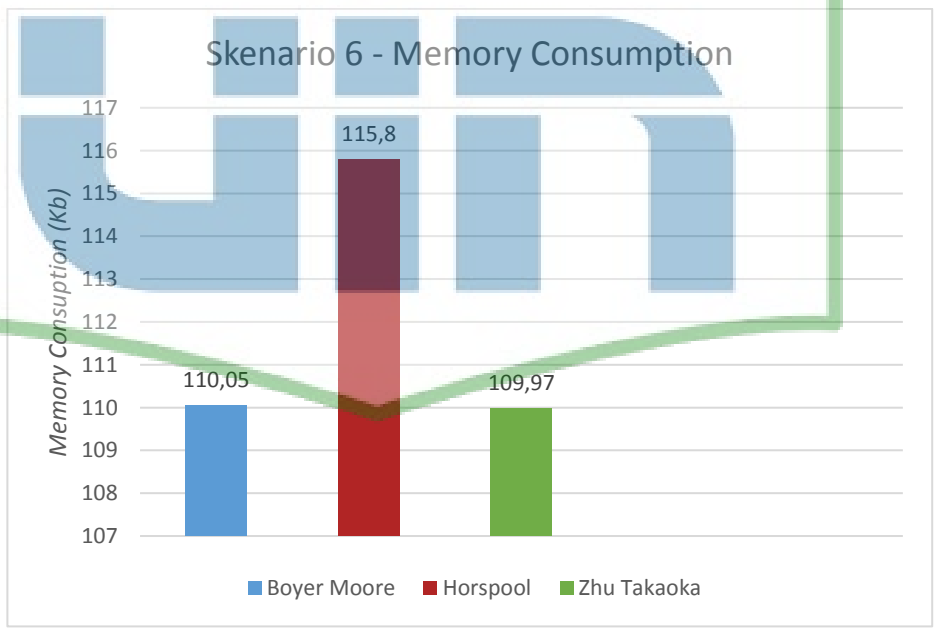
Output	Boyer Moore	Horspool	Zhu Takoka
Runtime (s)	0,094	0,114	0,136
Memory Consumption (kb)	110,05	115,8	109,97
Accuracy (%)	100	100	100

Pada tabel di atas menunjukkan hasil simulasi pada skenario 6 yang dilakukan terhadap tiga algoritma. Boyer Moore memiliki nilai rata-rata *runtime* terkecil yaitu 0,094 s dan nilai rata-rata *memory consumption* 110,05 kb. Horspool memiliki nilai rata-rata *runtime* tercepat dengan nilai 0,114 s dan memilki nilai rata-rata *memory consumption* terbesar yaitu 115,80 kb. Zhu Takaoka memilki nilai rata-rata *runtime* 0,136 s dan memiliki nilai rata-rata *memory consumption* terkecil yaitu 109,97 kb. Untuk *accuracy*, masing-masing memilki nilai sebesar 100 %.



Gambar 5.16 Hasil Perbandingan Skenario 6 *Runtime*

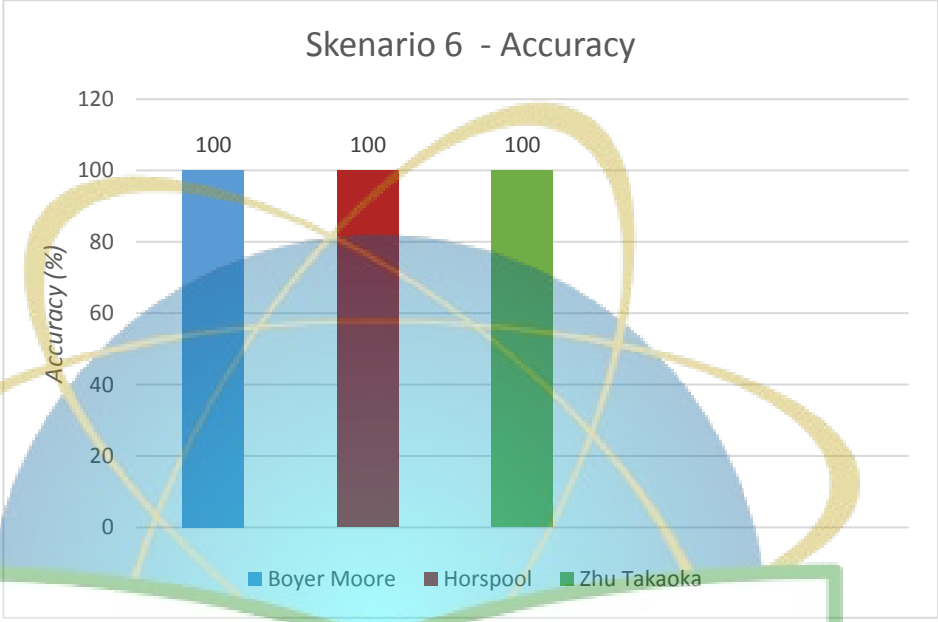
Grafik di atas menunjukkan perbandingan nilai rata-rata *runtime* untuk masing-masing algoritma. Semakin kecil nilai *runtime* pada proses di suatu algoritma maka hasil ini semakin bagus. Pada hasil skenario 6 , algoritma Boyer Moore memiliki nilai *runtime* yang terbaik.



Gambar 5.17 Hasil Perbandingan Skenario 6 *Memory Consumption*

Grafik di atas menunjukkan perbandingan nilai rata-rata *memory consumption* untuk masing-masing algoritma. Semakin kecil nilai *memory*

consumption pada proses pencarian di suatu algoritma maka hal ini bisa dikatakan semakin bagus. Sebab akan membutuhkan *resource* yang lebih kecil. Pada skenario 6, algoritma Zhu Takaoka memilki *memory consumption* yang terbaik.



Gambar 5.18 Hasil Perbandingan Skenario 6 *Accuracy*

Grafik di atas menunjukkan nilai *accuracy* masing-masing algoritma. Nilai *accuracy* merupakan sebuah pencocokan banyaknya *output* kata dengan perhitungan manual dan yang ada di sistem. Pada skenario 6, setiap algoritma memiliki nilai *accuracy* sebesar 100 %.

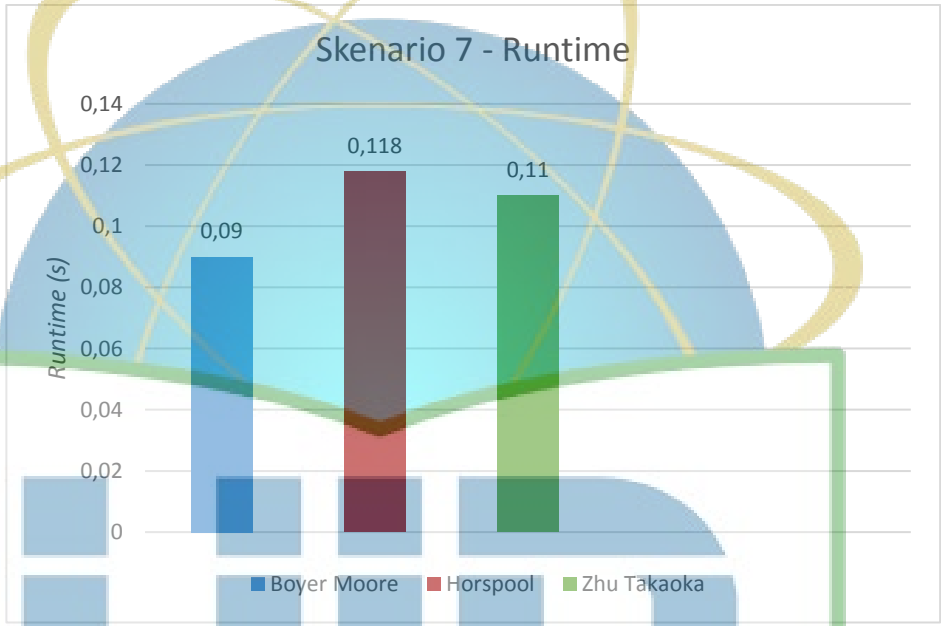
5.4.7. Skenario 7

Hasil dari masing-masing percobaan pada skenario 7 dilakukan untuk menghitung rata-rata. Berikut ini rata-rata hasil dari skenario 7 pada algoritma Boyer Moore, Horspool, dan Zhu Takaoka :

Tabel 5.32 Hasil Perbandingan Skenario 7

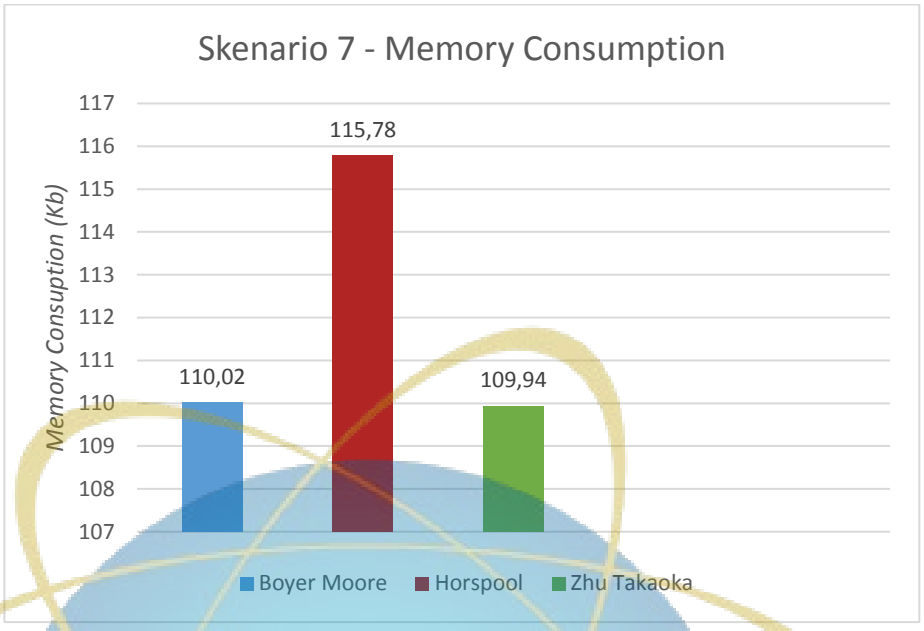
Output	Boyer Moore	Horspool	Zhu Takoka
Runtime (s)	0,09	0,118	0,11
Memory Consumption (kb)	110,02	115,78	109,94
Accuracy (%)	100	100	100

Pada tabel di atas menunjukkan hasil simulasi pada skenario 7 yang dilakukan terhadap tiga algoritma. Boyer Moore memiliki nilai rata-rata *runtime* terkecil 0,09 s dan nilai rata-rata *memory consumption* 110,02 kb. Horspool memiliki nilai rata-rata *runtime* dengan nilai 0,118 s dan memilki nilai rata-rata *memory consumption* terbesar yaitu 115,78 kb. Zhu Takaoka memilki nilai rata-rata *runtime* 0,11 s dan memiliki nilai rata-rata *memory consumption* terkecil yaitu 109,94 kb. Untuk *accuracy*, masing-masing memilki nilai sebesar 100 %.



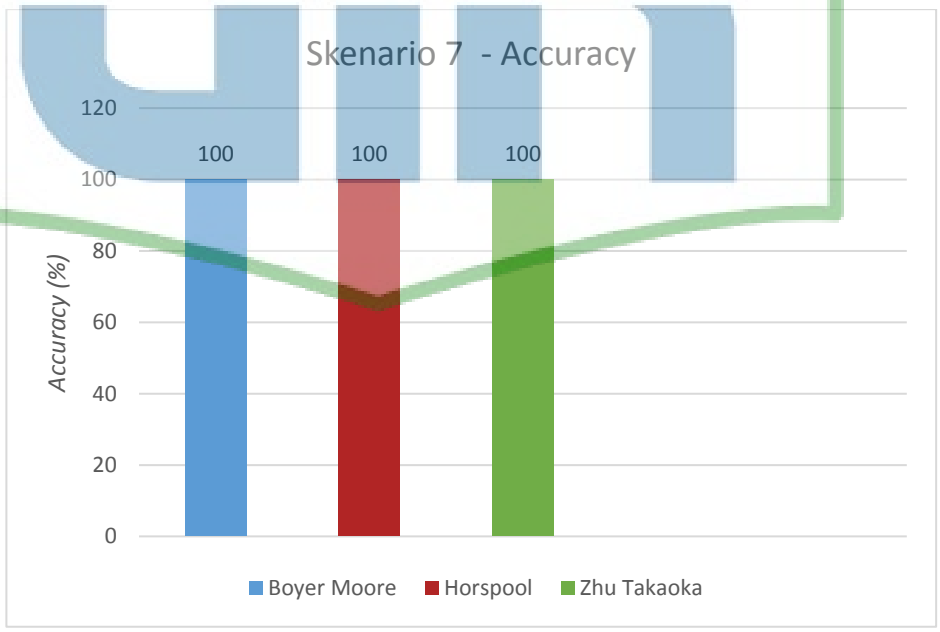
Gambar 5.19 Hasil Perbandingan Skenario 7 *Runtime*

Grafik di atas menunjukkan perbandingan nilai rata-rata *runtime* untuk masing-masing algoritma. Semakin kecil nilai *runtime* pada proses di suatu algoritma maka hasil ini semakin bagus. Pada hasil skenario 7, algoritma Boyer Moore memiliki nilai *runtime* yang terbaik.



Gambar 5.20 Hasil Perbandingan Skenario 7 *Memory Consumption*

Grafik di atas menunjukkan perbandingan nilai rata-rata *memory consumption* untuk masing-masing algoritma. Semakin kecil nilai *memory consumption* pada proses pencarian di suatu algoritma maka hal ini bisa dikatakan semakin bagus. Sebab akan membutuhkan *resource* yang lebih kecil. Pada skenario 7, algoritma Zhu Takaoka memilki *memory consumption* yang terbaik.



Gambar 5.21 Hasil Perbandingan Skenario 7 *Accuracy*

Grafik di atas menunjukkan nilai *accuracy* masing-masing algoritma. Nilai *accuracy* merupakan sebuah pencocokan banyaknya *output* kata dengan perhitungan manual dan yang ada di sistem. Pada skenario 7, setiap algoritma memiliki nilai *accuracy* sebesar 100 %.

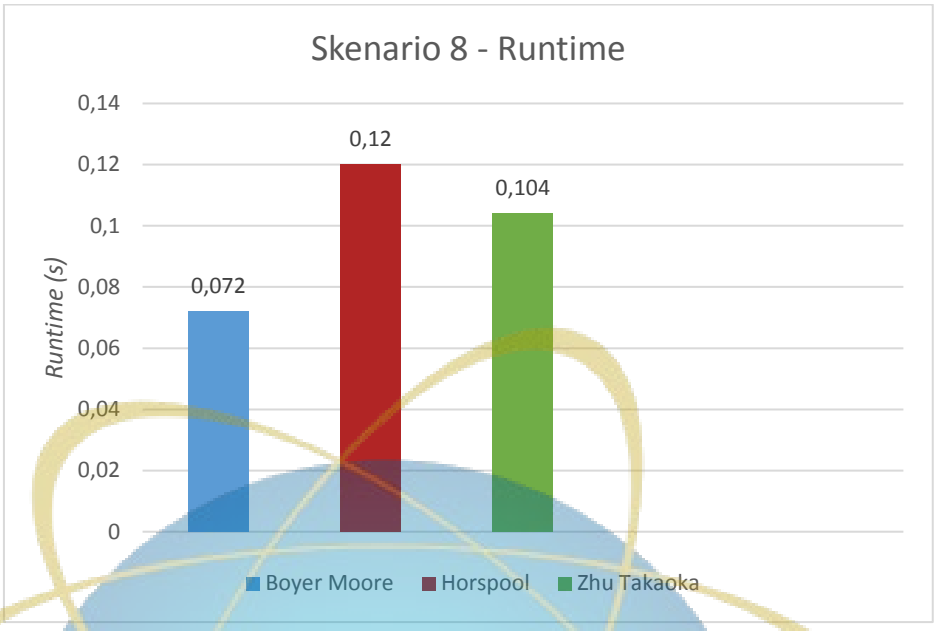
5.4.8. Skenario 8

Hasil dari masing-masing percobaan pada skenario 8 dilakukan untuk menghitung rata-rata. Berikut ini rata-rata hasil dari skenario 8 pada algoritma Boyer Moore, Horspool, dan Zhu Takaoka :

Tabel 5.33 Hasil Perbandingan Skenario 8

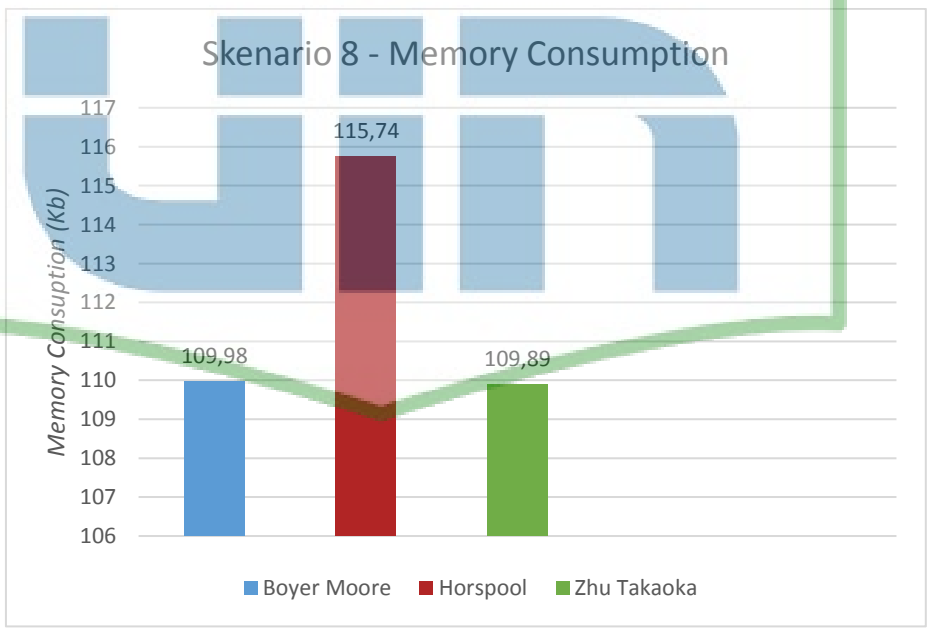
Output	Boyer Moore	Horspool	Zhu Takoka
Runtime (s)	0,072	0,12	0,104
Memory Consumption (kb)	109,98	115,74	109,89
Accuracy (%)	100	100	100

Pada tabel di atas menunjukkan hasil simulasi pada skenario 8 yang dilakukan terhadap tiga algoritma. Boyer Moore memiliki nilai rata-rata *runtime* tercepat yaitu 0,072 s dan nilai rata-rata *memory consumption* 109,98 kb. Horspool memiliki nilai rata-rata *runtime* dengan nilai 0,12 s dan memilki nilai rata-rata *memory consumption* terbesar yaitu 115,74 kb. Zhu Takaoka memilki nilai rata-rata *runtime* 0,104 s dan memiliki nilai rata-rata *memory consumption* terkecil yaitu 109,89 kb. Untuk *accuracy*, masing-masing memilki nilai sebesar 100 %.



Gambar 5.22 Hasil Perbandingan Skenario 8 *Runtime*

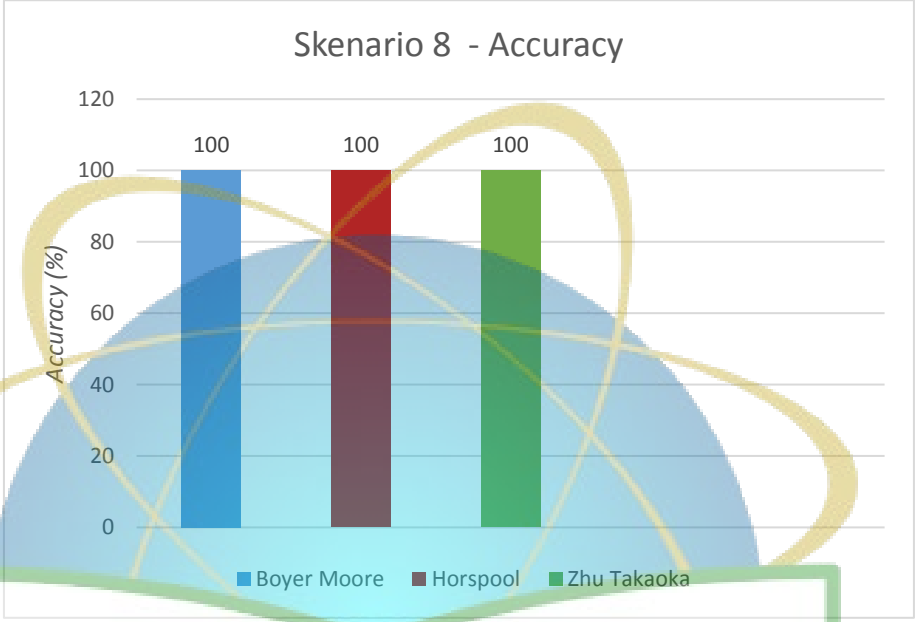
Grafik di atas menunjukkan perbandingan nilai rata-rata *runtime* untuk masing-masing algoritma. Semakin kecil nilai *runtime* pada proses di suatu algoritma maka hasil ini semakin bagus. Pada hasil skenario 8, algoritma Boyer Moore memiliki nilai *runtime* yang terbaik.



Gambar 5.23 Hasil Perbandingan Skenario 8 *Memory Consumption*

Grafik di atas menunjukkan perbandingan nilai rata-rata *memory consumption* untuk masing-masing algoritma. Semakin kecil nilai *memory*

consumption pada proses pencarian di suatu algoritma maka hal ini bisa dikatakan semakin bagus. Sebab akan membutuhkan *resource* yang lebih kecil. Pada skenario 8, algoritma Zhu Takaoka memilki *memory consumption* yang terbaik.



Gambar 5.24 Hasil Perbandingan Skenario 8 *Accuracy*

Grafik di atas menunjukkan nilai *accuracy* masing-masing algoritma. Nilai *accuracy* merupakan sebuah pencocokan banyaknya *output* kata dengan perhitungan manual dan yang ada di sistem. Pada skenario 8, setiap algoritma memiliki nilai *accuracy* sebesar 100 %.

5.5. Analisis *Output* dengan Metode Perbandingan Eksponensial

Dalam menghitung dan membandingkan proses pencarian dari ketiga algoritma tersebut sebagai berikut:

1. Menentukan alternatif

   Dalam penelitian ini, parameter yang digunakan untuk membandingkan algoritma yang satu dengan yang lainnya adalah *runtime* dan *memory consumption*. Dari hasil analisis perbandingan kecepatan (*runtime*) dan memori yang digunakan (*memory consumption*) antara algoritma Boyer Moore, Horspool, dan Zhu Takaoka dalam melakukan pencarian maka perlu dilakukan penentuan

algoritma yang mana yang akan digunakan sebagai algoritma pencarian paling efisien.

2. Menentukan kriteria

Untuk dapat membandingkan kedua alternatif tersebut, maka selanjutnya perlu dilakukan penentuan kriteria dalam menganalisa proses dan cara kerjanya. Untuk kriterianya dapat dilihat pada tabel berikut :

Tabel 5.34 Penentuan Kriteria

Kriteria	Keterangan
Jumlah waktu yang digunakan dalam melakukan pencarian	Perhitungan waktu yang diperoleh pada saat algoritma melakukan pencocokan *string* dari awal pencocokan sampai selesai
Besar memori yang digunakan saat melakukan pencarian	Perhitungan pemakaian memori terjadi pada saat algoritma melakukan pencocokan *string*

3. Menentukan bobot kriteria

Penentuan bobot merupakan salah satu komponen yang sangat berpengaruh terhadap nilai analisis, untuk itu menetukan bobot kriteria berdasarkan tingkatan pengaruh dalam menentukan kecepatan dalam melakuakn pencarian.

Berdasarkan hasil wawancara yang penulis lakukan dengan kepala bidang PUSTIPANDA UIN Syarif Hidayatullah Jakarta, Bapak Nahsrul Hakiem, Ph.D sebagai narasumber pakar dalam penentuan masing-masing algoritma, kecepatan eksekusi dan konsumsi memori merupakan suatu hal yang penting dalam suatu proses pencarian karakter dan dapat dijadikan sebagai perbandingan untuk menentukan performa setiap algoritma. Di dapatkan hasil rata-rata pembobotan penilaian *runtime* dan *memory consumption* adalah 0,5.

Selain itu, berdasarkan studi literatur yang penulis gunakan dalam penilitian ini dalam jurnal yang berjudul *Analisis Perbandingan Boyer Moore dan Knuth Morris Pratt dalam Pencarian Judul Buku Menerapkan Metode Perbandingan Eksponensial* (Fau, Mesran, & Ginting, 2017) didapatkan pemberian bobot pada *runtime* dan *memory consumption*, masing-masing sebesar 0,5.

Tabel 5.35 Pembobotan Masing-Masing Kriteria

Kriteria	Presentase Pengaruh Kriteria	Bobot *Range* (0-1)	Keterangan
Runtime (s)	50%	0,5	Penilaian terhadap waktu dalam melakukan proses pencarian merupakan komponen yang dapat memberikan suatu nilai terhadap algoritma dalam melakukan pencarian
Memory Consumption (Kb)	50 %	0,5	Tingkat pengaruh penggunaan memori sangat berpengaruh dalam menentukan kecepatan sebuah algoritma dalam melakukan pencarian karena semakin banyak kapasitas memori yang digunakan saat melakukan pencarian, maka akan semakin lambat juga suatu algoritma menyelesaikan masalah

4. Pemberian Nilai pada Setiap Kriteria

Pada kriteria yang telah dibentuk harus diberikan nilai. Nilai tersebut dapat dilihat pada contoh di bawah ini yang dimana nilainya diambil berdasarkan analisa algoritma Boyer Moore, Horspool, dan Zhu Takaoka sebelumnya.

Tabel 5.36 Pemberian Nilai Kriteria

Alternatif	Proses Ke-	*Pattern*	Kriteria	
			Runtime (s)	*Memory Consumption* (kb)
Algoritma Boyer Moore	1	Wahyu	0,158	110,22
	2	Umar Bin Khatab	0,118	110,19
	3	Tanda-tanda orang munafik	0,114	110,14
	4	Semua perbuatan tergantung pada niatnya	0,112	110,13
	5	Di antara tanda-tanda kiamat adalah diangkatnya ilmu	0,106	110,06
	6	Pertanyaan Jibril kepada Nabi SAW tentang Iman, Islam, Ihsan	0,094	110,05
	7	Bagaimana permulaan wahyu yang diturunkan kepada Rasullah SAW?	0,09	110,02
	8	Seorang muslim adalah orang yang membuat muslim lain selamat dari lidah dan tangannya	0,072	109,98
Algoritma Horspool	1	Wahyu	0,084	115,96
	2	Umar Bin Khatab	0,086	115,95
	3	Tanda-tanda orang munafik	0,098	115,91
	4	Semua perbuatan tergantung pada niatnya	0,106	115,98
	5	Di antara tanda-tanda kiamat adalah diangkatnya ilmu	0,106	115,84

	6	Pertanyaan Jibril kepada Nabi SAW tentang Iman, Islam, Ihsan	0,114	115,8
	7	Bagaimana permulaan wahyu yang diturunkan kepada Rasullah SAW?	0,118	115,78
	8	Seorang muslim adalah orang yang membuat muslim lain selamat dari lidah dan tangannya	0,12	115,74
Algoritma Zhu Takaoka	1	Wahyu	0,138	110,13
	2	Umar Bin Khatab	0,052	110,01
	3	Tanda-tanda orang munafik	0,12	110,05
	4	Semua perbuatan tergantung pada niatnya	0,108	110,04
	5	Di antara tanda-tanda kiamat adalah diangkatnya ilmu	0,084	109,97
	6	Pertanyaan Jibril kepada Nabi SAW tentang Iman, Islam, Ihsan	0,136	109,97
	7	Bagaimana permulaan wahyu yang diturunkan kepada Rasullah SAW?	0,11	109,94
	8	Seorang muslim adalah orang yang membuat muslim lain selamat dari lidah dan tangannya	0,104	109,89

5. Menghitung Nilai

Setelah melakukan pengisian nilai terhadap masing-masing kriteria, maka proses berikutnya adalah melakukan perhitungan dengan menggunakan rumus dari Metode Perbandingan Eksponensial (MPE). Proses perhitungannya sebagai berikut :

Tabel 5.37 Hasil Proses Algoritma

Skenario	Parameter	Bobot	Boyer Moore	Horspool	Zhu Takaoka
1	*Runtime (s)*	0,5	0,158	0,084	0,138
2		0,5	0,118	0,086	0,052

3		0,5	0,114	0,098	0,12
4		0,5	0,112	0,106	0,108
5		0,5	0,106	0,106	0,084
6		0,5	0,094	0,114	0,136
7		0,5	0,09	0,118	0,11
8		0,5	0,072	0,12	0,104
Total			**0,864**	**0,832**	**0,852**
1	*Memory Consumption*	0,5	110,22	115,96	110,13
2		0,5	110,19	115,95	110,01
3		0,5	110,14	115,91	110,05
4		0,5	110,13	115,98	110,04
5		0,5	110,06	115,84	109,97
6		0,5	110,05	115,8	109,97
7		0,5	110,02	115,78	109,94
8		0,5	109,98	115,74	109,89
Total			**880,79**	**926,96**	**880**

Langkah-langkah/proses perhitungan adalah sebagai berikut:

a. Proses perhitungan total nilai pada proses ke – 1 :

Nilai Boyer Moore $= (0{,}158)^{0,5} + (110{,}22)^{0,5}$

$= 0{,}397492 + 10{,}49857$

$= 10{,}89606$

Nilai Horspool $= (0{,}084)^{0,5} + (115{,}96)^{0,5}$

$= 0{,}289828 + 10{,}768473$

$= 11{,}0583$

Nilai Zhu Takaoka $= (0{,}138)^{0,5} + (110{,}13)^{0,5}$

$= 0{,}371484 + 10{,}49428$

$= 10{,}86577$

b. Proses perhitungan total nilai pada proses ke – 2 :

Nilai Boyer Moore $= (0{,}118)^{0,5} + (110{,}19)^{0,5}$

$= 0{,}343511 + 10{,}49714$

$= 10{,}84065$

Nilai Horspool $= (0{,}086)^{0,5} + (115{,}95)^{0,5}$

= 0,293258 + 10,768008

= 11,06127

Nilai Zhu Takaoka = $(0{,}052)^{0{,}5}$ + $(110{,}01)^{0{,}5}$

= 0,228035 + 10,48857

= 10,7166

c. Proses perhitungan total nilai pada proses ke – 3 :

Nilai Boyer Moore = $(0{,}114)^{0{,}5}$ + $(110{,}14)^{0{,}5}$

= 0,337639 + 10,49476

= 10,8324

Nilai Horspool = $(0{,}098)^{0{,}5}$ + $(115{,}91)^{0{,}5}$

= 0,31305 + 10,766151

= 11,0792

Nilai Zhu Takaoka = $(0{,}12)^{0{,}5}$ + $(110{,}05)^{0{,}5}$

= 0,34641 + 10,49047

= 10,83688

d. Proses perhitungan total nilai pada proses ke – 4 :

Nilai Boyer Moore = $(0{,}112)^{0{,}5}$ + $(110{,}13)^{0{,}5}$

= 0,334664 + 10,49428

= 10,82895

Nilai Horspool = $(0{,}106)^{0{,}5}$ + $(115{,}98)^{0{,}5}$

= 0,325576 + 10,769401

= 11,09498

Nilai Zhu Takaoka = $(0{,}108)^{0{,}5}$ + $(110{,}04)^{0{,}5}$

= 0,328634 + 10,49

= 10,81863

e. Proses perhitungan total nilai pada proses ke – 5 :

Nilai Boyer Moore = $(0{,}106)^{0{,}5}$ + $(110{,}06)^{0{,}5}$

= 0,325576 + 10,49095

= 10,81652

Nilai Horspool = $(0{,}106)^{0{,}5}$ + $(115{,}84)^{0{,}5}$

= 0,325576 + 10,762899

= 11,08848

Nilai Zhu Takaoka = $(0{,}084)^{0{,}5}$ + $(109{,}97)^{0{,}5}$

= 0,289828 + 10,48666

= 10,77649

f. Proses perhitungan total nilai pada proses ke – 6 :

Nilai Boyer Moore = $(0{,}094)^{0{,}5}$ + $(110{,}05)^{0{,}5}$

= 0,306594 + 10,49047

= 10,79707

Nilai Horspool = $(0{,}114)^{0{,}5}$ + $(115{,}8)^{0{,}5}$

= 0,337639 + 10,761041

= 11,09868

Nilai Zhu Takaoka = $(0{,}136)^{0{,}5}$ + $(109{,}97)^{0{,}5}$

= 0,368782 + 10,48666

= 10,85544

g. Proses perhitungan total nilai pada proses ke – 7 :

Nilai Boyer Moore = $(0{,}09)^{0{,}5}$ + $(110{,}02)^{0{,}5}$

= 0,3 + 10,48904

= 10,78904

Nilai Horspool = $(0{,}118)^{0{,}5}$ + $(115{,}78)^{0{,}5}$

= 0,343511 + 10,760112

= 11,10362

Nilai Zhu Takaoka = $(0{,}11)^{0{,}5}$ + $(109{,}94)^{0{,}5}$

= 0,331662 + 10,48523

= 10,81689

h. Proses perhitungan total nilai pada proses ke – 8 :

Nilai Boyer Moore = $(0{,}072)^{0{,}5}$ + $(109{,}98)^{0{,}5}$

= 0,268328 + 10,48713

= 10,75546

Nilai Horspool = $(0{,}12)^{0{,}5} + (115{,}74)^{0{,}5}$

= 0,34641 + 10,758253

= 11,10466

Nilai Zhu Takaoka = $(0{,}104)^{0{,}5} + (109{,}89)^{0{,}5}$

= 0,32249 + 10,48284

= 10,80533

i. Menghitung nilai prioritas keputusan

Total Nilai Boyer Moore = 10,89606 + 10,84065 + 10,8324 + 10,82895 + 10,81652 + 10,79707 + 10,78904 +10,75546

= 86,55015

Total Nilai Horspool = 11,0583 + 11,06127 + 11,0792 + 11,09498 + 11,08848 + 11,09868 + 11,10362 + 11,10466

= 88,68919

Total Nilai Zhu Takaoka = 10,86577 + 10,7166 + 10,83688 + 10,81863 + 11,08848 + 10,85544 + 10,81689 + 10,80533

= 86,80402

6. Menentukan Hasil atau Prioritas Keputusan

Setelah diperoleh nilai akhir atau total nilai dari masing-masing, alternatif, maka tahapan selanjutnya yang perlu dilakukan adalah menentukan prioritas keputusan berdasarkan nilai dari masing-masing alternatif. Hasil prioritas keputusan dapat dilihat pada tabel dibawah ini :

Tabel 5.38 Prioritas Keputusan

Alternatif	Total Nilai	Ranking
Algoritma Boyer Moore	86,55015	1
Algoritma Zhu Takaoka	86,80402	2
Algoritma Horspool	88,68919	3

Dari hasil perhitungan dengan menggunakan Metode Perbandingan Eksponensial, dapat diketahui bahwa Algoritma yang paling efektif dengan parameter *runtime* dan *memory consumption* adalah Algoritma Boyer Moore. Selanjutnya, posisi yang kedua adalah Algoritma Zhu Takaoka dan posisi terakhir adalah Algoritma Horspool. Namun, setiap algoritma ini memiliki kelebihan dan kekurangannya masing-masing. Dari analisis kinerja yang telah penulis lakukan, untuk parameter *runtime*, Algoritma Boyer Moore adalah algoritma yang paling baik digunakan untuk pencarian kata dengan jumlah karakter yang banyak, sedangkan algoritma Horspool adalah algoritma yang paling baik digunakaan jika ingin mencari kata dengan jumlah karakter yang sedikit.

Untuk paramater *memory consumption*, semakin banyak karakter yang dicari, maka masing-masing algoritma akan membutuhkan memori yang semakin sedikit. Namun, dari hasil analisis yang telah penulis lakukan, *memory consumption* terkecil adalah algoritma Zhu Takaoka dan diikuti dengan algoritma Boyer Moore.

BAB VI
PENUTUP

6.1. Kesimpulan

Hasil perbandingan Algoritma Boyer Moore, Algoritma Horspool, dan Algoritma Zhu Takaoka pada repositori hadits terjemahan Bahasa Indonesia dengan menggunakan metode simulasi yang terdiri dari tahapan *problem formulation, conceptual model, input and output data, modeling, simulation, verification and validation, experimentation, dan output analysis* dan setelah melakukan perhitungan dengan Metode Perbandingan Eksponensial (MPE) dengan parameter *runtime* dan *memory consumption* menunjukkan bahwa Algoritma Boyer Moore memiliki nilai terbaik, dilanjutkan dengan Algoritma Zhu Takaoka dan terakhir Algoritma Horspool.

6.2. Saran

Setelah melakukan penelitian ini, ada beberapa hal yang kedepannya dapat dikembangkan guna proses pengembangan selanjutnya. Adapun saran dari penulis antara lain :

1. Algoritma yang digunakan atau yang dibandingkan dapat ditambahkan dengan algoritma-algoritma *String Matching* lainnya, seperti Turbo Boyer Moore atau Tuned Boyer Moore.
2. Untuk mengetahui hasil pembanding yang lebih luas dapat menggunakan bahasa pemrograman berbeda dengan implementasi ke objek lainnya dari yang penulis gunakan.

DAFTAR PUSTAKA

Adhi, K., & Khrisnandi, W. A. (2017). *Perbandingan Algoritma Horspool dan Algoritma Zhu-Takaoka dalam Pencarian String Berbasis Desktop* Vol. 6

Ardi, Andreswari, & Setiawan. (2017). *Rancang Bangun Aplikasi Kamus Istilah Kedokteran dengan Menggunakan Algoritma Boyer Moore Berbasis Android.* Vol. 5.

Argakusumah, K. W. & Hansun, S. (2016). *Implementasi Algoritma Boyer-Moore pada Aplikasi Kamus Kedokteran Berbasis Android*, Vol.6.

Aulia, A. (2017). *Pengembangan Repositori Hadits Terjemahan Bahasa Indonesia (Studi Kasus : Hadits Bukhori)*. UIN Syarif Hidayatullah Jakarta.

Aulia, A., Khairani, D., Bahaweres, R. B., & Hakiem, N. (2017). *WatsaQ : Repository of Al Hadith in Bahasa (Case Study : Hadith Bukhari). International Conference on Electrical Engineering, Computer Science and Informatics*.

Aulia, A., Khairani, D., & Hakiem, N. (2017). *Development of a Retrieval System for Al Hadith in Bahasa (Case Study : Hadith Bukhari). Conference on Cyber and IT Service Management*.

Bustamin & Hasanuddin. (2010). *Membahas Kitab Hadis*. Lembaga Penelitian UIN Syarif Hidayatullah Jakarta.

Effendi. (2013). Pengembangan Algoritma Boyer Moore pada Translator Bahasa Pemrograman. Vol.7.

Fadly F., Masruroh, S. U., & Suseno, H. B. (2017). *Evaluasi Kinerja Enkripsi Terotentifikasi AES-GSM dan AES-AEX pada Aplikasi Instant Messenger*.

Fau, Mesran, & Ginting. (2017). *Analisa Perbandingan Boyer Moore Dan Knuth Morris Pratt Dalam Pencarian Judul Buku Menerapkan Metode Perbandingan Eksponensial (Studi Kasus : Perpustakaan STMIK Budi Darma).* Vol. 6.

Fikri, Nurhayati, & Masruroh, S, U. (2016). *Analisa Proses File Carving Menggunakan Photorec Dan Foremost*. UIN Syarif Hidayatullah Jakarta.

Geraldy, Sediyono, & Beeh. (2016). *Studi Perbandingan Algoritma Brute Force,*

Algoritma Knuth Morris Pratt, Algoritma Boyer Moore untuk Identifikasi Kesalahan Penulisan Teks Berbasis Android.

Handrizal, Budiman, & Ardani. (2017). *Implementation and Analysis Zhu-Takaoka Algorithm and Knuth-Morris-Pratt Algorithm for Dictionary of Computer Application Based on Android*. Vol. 1.

Idri. (2017). *Hadis dan Orinetalis*. Jakarta: Elsevier.

Jiawei Han, M. K. (2012). *Data Mining Concept and Tecniques*. Jakarta: Elsevier.

Joaqin. (2016). *Buku Pintar Hadits Edisi Revisi*. Jakarta: Qiblajoa.

Kurniadi, I. (2013). *Logika dan Algoritma Dasar Menggunakan Bahasa C++*. Jakarta: Mitra Wacana Media.

Mulyani, S. (2016). *Metode Analisis dan Perancangan Sistem*. Bandung: Informatika.

Munir, R. (2016). *Algoritma dan Pemograman*. Bandung: Informatika.

Pratiwi, H. (2016). *Buku Ajar Sistem Pendukung Keputusan*. Bandung: Abdi Sistematika.

Reitz, J. (n.d.). Online Dictionary of Library and Information Science. (Diakses pada 27 Mei 2018).

Rhosa A.S dan M. Shahaludin. (2014). *Rekayasa Perangkat Lunak*. Bandung: Informatika.

Sari, F. (2018). *Metode dalam Pengambilan Keputusan*. Yogyakarta: Deepublish Publisher.

Siahaan Ayu Permatasari dan Mesran. (2018). *Implementasi Algoritma Boyer Moore pada Aplikasi Kamus Nama Bayi Beserta Maknanya Berbasis Android*. Vol. 17.

Singh & Verma. (2011). *A Fast String Matching Algorithm*. *International Journal of Computer Technology and Applications*, *2*.

Sitorus, L. (2015). *Algoritma dan Pemograman*. Yogyakarta: CV Andi Publisher.

Sugiyono. (2013). *Metode Penelitian Kuantitatif, Kualitatif dan R&D*. Bandung: Alfabeta.

Tambun, E. D. (2011). *Perbandingan Penggunaan Algoritma BM dan Algoritma Horspool pada Pencarian String dalam Bahasa Medis.*

Togatorop, Aan, & Coastera. (2017). Implementasi Algoritma Zhu Takaoka pada Aplikasi Kamus Istilah Musik Berbasis Android, *5*(2), 147–153.

Wulan. (2011). *Analisis Penerapan String Matching dalam Komparasi Data Kepesertaan Jaminan Kesehatan Masyarakat (JAMKESMAS)*. UIN Syarif Hidayatullah Jakarta.