ISSN 2085-4552

Implementasi Algoritma Simon Pada Aplikasi Kamus Perubahan Fi'il (Kata Kerja Bahasa Arab) Berbasis Android

Rahmad Akbar[1], Bambang Pramono[2], Rizal Adi Saputra[3]
[1,2,3]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari, 93231, Indonesia
[1]akbarbelajar@gmail.com, [2]bambang.pramono@uho.ac.id, [3]rizaladisaputraa@gmail.com

Diterima 29 November 2020
Disetujui 19 Mei 2021

***Abstract*— Shorof science or Tashrif is the scientific field of word derivation in Arabic, one focus of the discussion in this field is the process of changing verbs or also known as Fi'il into several other types of words, such as Fi'il Mudhori ', Fi'il Madhi , Fi'il Amr, Fi'il Nahi, Isim Fa'il, Isim Maf'ul, Isim Zaman, Isim Makan, Isim Alat, Masdar or Masdar mim. The process of learning Shorof science is still mostly carried out in traditional ways, especially in the pesantren environment by memorizing the derivatives of these words and their translations. While one of the basic books that is often used is the book Amtsilah At-Tashrifiyah written by KH.Ma'shum bin Ali as a reference for the process of changing words, while looking for a translation in Indonesian must use an Arabic-Indonesian dictionary. This study aims to simplify the word search process by making an android-based dictionary of Fi'il change and utilizing the Simon Algorithm as a word search method, so as to simplify the learning process of Shorof's knowledge. Simon's algorithm is a string matching algorithm where the matching phase is carried out from left to right by initializing each index on a given pattern. After testing, the word search process can be carried out with an average running time of 23.8447333 milli second for searching Arabic words.**

***Index Terms*— String matching algorithm, Simon algorithm, Android Fi'il Dictionary**

I. PENDAHULUAN

Bahasa Arab merupakan bahasa resmi di dunia Internasional dan merupakan salah satu bahasa yang kaya akan derivasi atau kata turunan, sebuah kata dalam Bahasa Arab terkadang bisa dikembangkan menjadi puluhan bahkan ratusan. Bidang keilmuan derivasi dalam Bahasa Arab disebut juga dengan ilmu *Shorof* atau *Tashrif*, sedangkan kata dalam Bahasa Arab terbagi menjadi tiga kelompok yaitu *Isim*(kata benda), *Fi'il*(kata kerja) dan *Khurf*(huruf yang memiliki arti) [1]. Diantara ketiga kelompok kata tersebut *Fi'il* menjadi kelompok kata yang memiliki banyak turunan dan perubahan, sehingga perlu adanya pembahasan tersendiri seperti yang terangkum dalam kitab *Tashrif* yang berjudul *Amtsilah At-Tashrifiyah* yang ditulis oleh KH.Ma'shum bin Ali, buku tersebut kusus membahas tentang perubahan-perubahan *Fi'il,* antara lain *Fi'il Mudhore'*(kata kerja bentuk sekarang dan yang akan datang), *Fi'il Madhi*(kata kerja lampau), *Masdar*(kata kerja yang dibendakan) dan masih banyak lagi. Akan tetapi daftar kata pada kitab *Amtsilah At-Tashrifiyah* tersebut tidak memiliki terjemahan secara langsung dalam bahasa Indonesia, sehingga pengguna harus memiliki kamus bahasa Arab-Indonesia untuk mengetahui arti dari daftar kata tersebut beserta turunan-turunanya.

Berdasarkan beberapa uraian di atas, menegaskan bahwa perlunya sebuah aplikasi yang dapat membantu memudahkan proses pembelajaran Bahasa Arab, dan yang lebih penting lagi dapat memudahkan proses pencarian *Fi'il* beserta turunan-turunan dan terjemahanya dalam bahasa Indonesia. Dengan adanya solusi pembuatan tersebut, dibutuhkan sebuah metode yang dapat digunakan dalam proses pencarian kata pada aplikasi yang akan dibangun, salah satu metode yang dapat digunakan antara lain ialah Algoritma *Simon.*

Algoritma *Simon* merupakan sebuah algoritma pencocokan *string* dengan fase pencarianya dilakukan dari kiri ke kanan dengan tahapan inisialisasi tiap indeks pada pola yang diberikan [9]. Berdasarkan beberapa penelitian Algoritma *Simon* terdahulu, yaitu penelitian yang telah dilakukan oleh Putri Chaliska pada tahun 2017 dengan membandingkan Algoritma *Simon* dengan Algoritma *On Ordered Alphabets* dan memperoleh kesimpulan bahwa Algoritma *Simon* membutuhkan waktu yang lebih cepat dengan rata-rata waktu 1,5134 ms dan kompleksitas algoritma yang diperoleh Θ(n), sedangkan Algoritma *On Ordered Alphabets* memiliki rata-rata waktu 1,6928 ms[3]. Kemudian Megawaty pada tahun 2018 juga melakukan penelitian dengan membandingkan Algoritma *Simon* dengan Algoritma *Brute Force* dan memperoleh kesimpulan bahwa *Algoritma Simon* membutuhkan waktu lebih cepat dengan rata-rata waktu 1,8 ms, sedangkan Algoritma *Brute Force* membutuhkan rata-rata waktu 5,5 ms [9]. Adapun *string* yang digunakan pada kedua penelitian tersebut berupa *string* dalam Bahasa Indonesia, sedangkan

pada penelitian ini *string* yang digunakan dalam Bahasa Arab.

II. LANDASAN TEORI

A. *Ilmu Shorof*

Ilmu *Shorof* adalah ilmu yang membahas tentang derivasi atau perubahan kata dalam Bahasa Arab. Perlu diketahui mempelajari Bahasa Arab termasuk didalamnya ilmu Nahwu dan ilmu *Shorof* hukumnya wajib. Karena mempelajari keduanya adalah merupakan sarana untuk dapat memahami serta mendalami makna yang terkandung dalam Al-Qur'an.

Ilmu *Shorof* juga dinamakan dengan *Ummul'ulum* (induknya ilmu) karena dari ilmu *Shorof* itu kita dapat mengetahui berbagai macam bentuk perubahan pecahan-pecahan kata yang antara kata satu dengan kata yang lainnya mempunyai arti yang berbeda. Sebagai contoh lafadz *fatahan* = فَتَحًا mempunyai arti pembukaan, *fatihun* = فَاتِحٌ mempunyai arti orang yang membuka, *miftāhun* = مِفْتَاحٌ mempunyai arti alat untuk membuka. semua ini berasal dari satu kata yaitu *fataha* = فَتَحَ yang artinya membuka.

Menurut arti bahasa (*lugat*) *Shorof/Taṣrif* yaitu berubah atau mengubah dari bentuk aslinya kepada bentuk yang lain. Misalnya merubah bentuk bangunan rumah kuno menjadi bentuk bangunan rumah modern. Sedangkan menurut istilah (menurut kalangan ulama *Shorof*), yaitu berubahnya bentuk asal pertama *Fi'il Maḍhi* menjadi *Fi'il Muḍhori'*, *Fi'il Amr, Fi'il Nahi, Isim Zaman*, *Isim Makan*, dan terakhir sampai pada *Isim Alat* [7].

B. *Algoritma String Matching*

Pencocokan *string* (*string matching*) adalah proses pencarian kata (*Pattern*) dalam sebuah teks. *String matching* bisa digunakan untuk menemukan satu *Pattern* yang pertama kali ditemukan, atau yang lebih umum menampilkan semua *Pattern* yang dapat ditemukan dalam teks. Jenis *string matching* bermacam-macam, dibedakan menurut hasil yang diinginkan [4]. Prinsip kerja algoritma *string matching* adalah sebagai berikut :

- Memindai teks dengan bantuan sebuah *window* yang ukurannya sama dengan *Pattern*.
- Menempatkan *window* pada awal teks
- Membandingkan karakter pada *window* dengan karakter dari *Pattern*.

Setelah pencocokan (baik hasilnya cocok atau tidak cocok), dilakukan *shift* ke kanan pada *window*. Prosedur ini dilakukan berulang-ulang sampai *window* berada pada ahkir teks. Mekanisme ini disebut *sliding-window*.

String matching dibagi menjadi dua, yaitu *exact string matching* dan *heuristic (statistical matching)*. *Exact matching* digunakan untuk menemukan *Pattern* yang berasal dari satu teks. Contoh pencarian *exact matching* adalah pencarian kata "pelajar" dalam kalimat " saya seorang pelajar" atau "saya seorang siswa". Sistem akan memberikan hasil bahwa kata pelajar dan siswa bersinonim. Algoritma *exact matching* diklasifisikan menjadi tiga bagian menurut arah pencariannya, yaitu :

- *Arah pencarian dari kiri ke kanan.*
  Algoritma yang termasuk dalam kategori ini adalah *Brute Force*, *Simon*, *Morris* dan *Pratt* (yang kemudian dikembangkan oleh *Knuth, Morris* dan *Pratt*).
- *Arah pembacaan dari kanan ke kiri.*
  Algoritma yang termasuk dalam kategori ini adalah *Boyer Moore* yang kemudian dikembangkan menjadi algoritma *Turbo Boyer Moore*, *Tuned Boyer Moore* dan *Zhu Takaoka*.
- *Arah pencarian yang ditentukan oleh program.*
  Algoritma yang termasuk dalam kategori ini adalah algoritma *Colussi* dan *Crochemore-Perrin*. *Heuristic matching* adalah teknik yang digunakan untuk menghubungkan dua data terpisah ketika *exact matching* tidak mampu mengatasi karena pembatasan pada data yang tersedia. *Heuristic matching* dapat dilakukan dengan perhitungan distance antara *Pattern* dengan teks. *Exact* dan *heuristic matching* memiliki kemiripan makna tetapi berbeda tulisan.

C. *Algoritma Simon*

Pada Algoritma *Simon* akan dilakukan fase untuk pencocokan string yang dilakukan dari kiri ke kanan dengan tahapan inisialisasi di setiap indeksnya berdasarkan *Pattern* yang akan diberikan, kemudian pada keadaan awal pencarian *string* diberi nilai -1 yang arti nya *mismatch* atau tidak sesuai dengan *Pattern*. Berikut adalah contoh Algortima Simon beserta simulasinya dalam melakukan pencocokan *Pattern* terhadap teks [3].

- *User* akan mencari kata " مَنْصُوْرٌ " dengan memasukan *Pattern* " صُوْرٌ " .
- Kata dan Pattern akan dibalik menggunakan fungsi *reverse* sehingga menjadi " رٌوْرُصُنْمَ " dan " رٌوْرُصُ ".
- Setelah *Pattern* melalui proses pembalikan, maka sistem langsung memproses dengan mendeklarasikan terlebih dahulu yaitu jumlah array *Pattern m = Pattern.lenght()*, jumah array setiap kata yang ada di *database n = text.lenght()*, *Result = False*, *j = 0* dan inisialisasi *x = -1*.
- Kemudian sistem akan melakukan proses pencocokan dengan syarat yang pertama yaitu *j<= n-1*.
- Selanjutnya syarat yang kedua ialah *x+1 < m*.

- Setelah dua syarat diatas terpenuhi, sistem langsung melakukan proses pencocokkan karakter dimulai dari pencocokan karakter array *ke x+1* dari *Pattern* dengan karakter array ke *j* dari kata yang ada pada *database* atau ditulis dengan *Pattern(x+1)==text(j).*
- Apabila terjadi kecocokkan maka nilai *x* akan ditambah 1, namun apabila terjadi ketidak cocokan maka *x* diberi nilai -1, sedangkan nilai *j* selalu ditambah 1.
- Proses akan terus berulang hingga nilai *x = m-1* dengan kata lain semua *array Pattern* telah dicocokan.

Inisialisasi dimulai dari -1.

Keadaan : -1{ص ➡ 0}
Keadaan : 0{ص ➡ 0 ◌ُ ➡ 1}
Keadaan : 1{◌ُ ➡ 1 و ➡ 2}
Keadaan : 2{و ➡ 2 ◌ْ ➡ 3}
Keadaan : 3{◌ْ ➡ 3 ر ➡ 4}
Keadaan : 4{ر ➡ 4 ◌ٌ ➡ 5}
Keadaan : 5{ }
Berhenti di keadaan 5.

Tabel 1. Pencocokan Pertama

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P	ص									

-1

Karna *Pattern* tidak sesuai dengan teks maka م diberi nilai -1 yang berarti *mismatch*.

Tabel 2. Pencocokan Kedua

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P		ص								

-1

Karna *Pattern* tidak sesuai dengan teks maka ◌َ diberi nilai -1 yang berarti *mismatch*.

Tabel 3. Pencocokan Ketiga

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P			ص							

-1

Karna *Pattern* tidak sesuai dengan teks maka ن diberi nilai -1 yang berarti *mismatch*.

Tabel 4. Pencocokan Keempat

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ	م
P				ص							

-1

Karna *Pattern* tidak sesuai dengan teks maka ◌ْ diberi nilai -1 yang berarti *mismatch*.

Tabel 5. Pencocokan Kelima

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P					ص					

0

Karna *Pattern* sesuai dengan teks maka ص diberi nilai 0 yang artinya *match*.

Tabel 6. Pencocokan Keenam

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P						◌ُ				

1

Karna *Pattern* sesuai dengan teks maka ◌ُ diberi nilai 1 yang artinya *match*.

Tabel 7. Pencocokan Ketujuh

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P							و			

2

Karna *Pattern* sesuai dengan teks maka و diberi nilai 2 yang artinya *match.*

Tabel 8. Pencocokan Kedelapan

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P								◌ْ		

3

Karna *Pattern* sesuai dengan teks maka ◌ْ diberi nilai 3 yang artinya *match.*

Tabel 9. Pencocokan Kesembilan

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P									ر	

4

Karna *Pattern* sesuai dengan teks maka ر diberi nilai 4 yang artinya *match*.

Tabel 10. Pencocokan Kesepuluh

T	م	◌َ	ن	◌ْ	ص	◌ُ	و	◌ْ	ر	◌ٌ
P									ر	◌ٌ

5

Karna *Pattern* sesuai dengan teks maka ◌ٌ diberi nilai 5 yang artinya *match*. Keadaan 5 telah terpenuhi, sesuai inisialisasi awal pencarian akan berhenti pada keadaan 5.

D. *JSON(Java Script Object Notation)*

JSON(*Java Script Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat(generate) oleh komputer. Format ini dibuat berdasarkan dari bahasa pemrograman Java Script, Struktur data ini disebut sebagai struktur data universal. Pada dasarnya semua bahasa pemrograman modern mendukung struktur dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasrkan pada stuktur data ini. *JSON* menggunakan bentuk sebagai berikut.

Objek adalah nama/nilai yang tidak terurutkan. Objek dimulai dengan "{" (kurung kurawal buka) dan diakhiri dengan "}" (kurung kurawal tutup). Setiapnama diikuti dengan ":" (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh "," (koma)[6]

E. *Case Folding*

Case Folding adalah sebuah proses penyamaan besar kecilnya huruf pada setiap kata serta pembuangan tanda baca yang masih terdapat pada suatu kalimat [8]. Berikut adalah beberapa proses *Case Folding*

- Ubah kata menjadi huruf kecil
  Pada proses ini semua string yang terdapat pada *Pattern* akan diubah menjadi huruf kecil menggunakan fungsi *toLowerCase()*.
- Buang tanda baca
  Pada tahap ini hasil dari proses penyamaan huruf menjadi huruf kecil diolah kembali untuk menghilangkan *noise* berupa tanda baca dan angka yang masih terdapat pada huruf. Pada proses ini digunakan fungsi *replace()*, delimiter yang digunakan adalah *\d* yang menandakan digit[0-9] dan *\W* yang menandakan simbol atau karakter yang tidak termasuk kedalam huruf dan pengganti merupakan karakter yang digunakan untuk mengganti karakter yang sesuai dengan delimiter, pada kasus ini digunakan spasi sebagai karakter pengganti. Dengan delimiter tersebut fungsi akan menghapus angka dan simbol yang terdapat pada kata yang dimasukan sebagai parameter.

III. IMPLEMENTASI DAN PEMBAHASAN

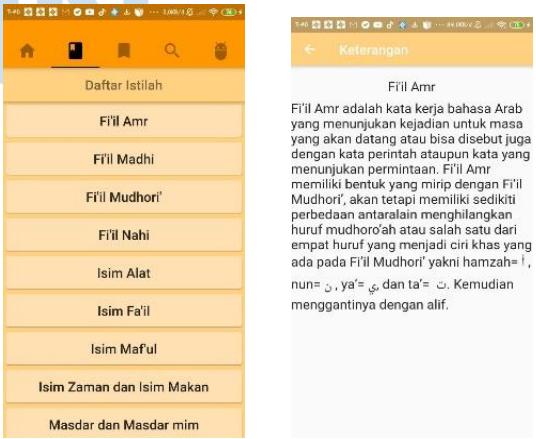
A. *Implementasi Rancangan Antarmuka User*

Implementasi *interface* dari perangkat lunak dilakukan berdasarkan rancangan yang telah dibuat pada bab sebelumnya, namun disesuaikan dengan komponen-komponen yang tersedia pada Flutter. Implementasi antarmuka ditampilkan dalam bentuk screenshoot aplikasi Android.

- *Beranda*
  Beranda merupakan tampilan awal ketika aplikasi pada aplikasi Kamus perubahan Fi'il dan berisi penjelasan singkat tentang aplikasi.

Gambar 1. Tampilan Menu Beranda *User*

- *Menu Istilah*
  Menu istilah menampilkan daftar istilah yang digunakan pada aplikasi kamus perubahan *Fi'il* .



Gambar 2. Tampilan Menu Istilah

- *Menu Wazan*
  Menu Wazan menampilkan daftar wazan yang terdapat pada aplikasi Kamus perubahan Fi'il beserta daftar kata dan turunan-turunanya dengan cara sebagai berikut:
  - Pilih menu daftar *Wazan*, maka daftar Wazan akan ditampilkan.

- Pilih salah satu *Wazan*, maka daftar kata yang terdapat pada *Wazan* tersebut akan ditampilkan dalam bentuk *Fi'il Madhi.*
- Pilih salah satu kata dari daftar kata tersebut, maka akan ditampilkan daftar turunan beserta terjemahan dan *audio* dari kata tersebut.

Gambar 3. Tampilan Menu *Wazan*

- *Menu Pencarian*

Menu Pencarian berisi *form* yang dapat digunakan mencari kata yang terdapat pada Kamus perubahan dengan cra sebagai berikut:

- Pilih menu pencarian.
- Pilih salah satu kategori pencarian yang menggunakan Bahasa Indonesia atau Bahasa Arab.
- Masukan kata yang akan dicari pada form pencarian.
- Apabila kata ditemukan, maka akan ditampilakan daftar kata tersebut dibawah form, baik itu berupa kata dalam bentuk *Fi'il Mudhori', Fi'il Madhi, Fi'il Amr, Fi'il Nahi, Isim Fa'il, Isim Maf'ul, Isim Zaman, Isim Makan, Isim Alat, Masdar* maupun *Masdar mim*.
- Pilih salah satu kata yang ditemukan, maka akan ditampilkan daftar turunan dari kata tersebut.

Gambar 4. Tampilan Menu Pencarian

- *Menu Tentang*

Menu tentang berisi informasi mengenai deskripsi ataupun spesifikasi dari sistem aplikasi Kamus Perubahan *Fi'il.*

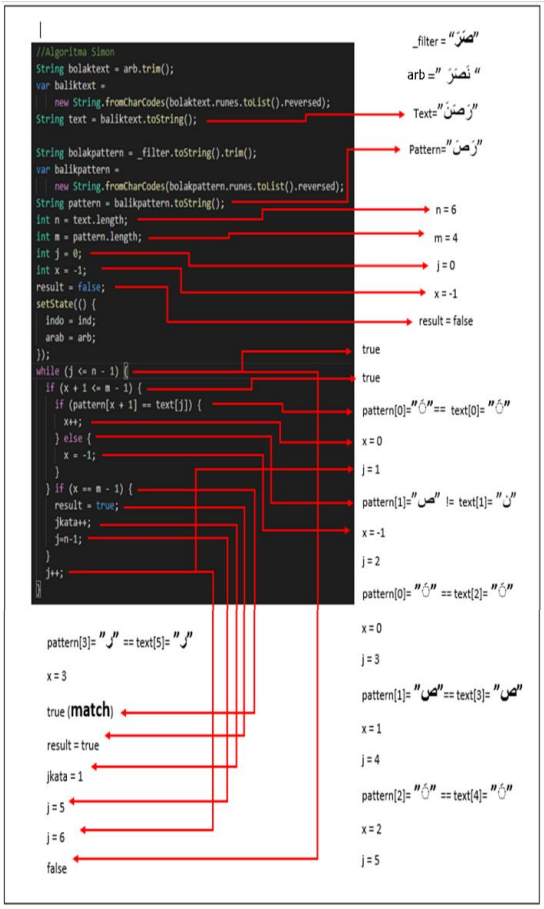
Gambar 5. Tampilan Menu Tentang

B. *Implementasi Coding Algoritma Simon*

Dalam proses coding, dibuat dua algoritma Simon yang masing-masing digunakan dalam proses pencocokan *string* Bahasa Arab. Adapun implementasinya adalah sebagai berikut :

- Deklarasi *String text* sebagai variabel *baliktext* yang berasal dari variabel *arb*, yakni *string* kata yang dipanggil dari data *JSON* dan variabel *arb* tersebut diproses dengan fungsi *runes.toList.reversed()* untuk membalik kata tersebut.
- Deklarasi *String pattern* sebagai *balikpattern* yang berasal dari variabel *_filter*, yakni kata yang dimasukan oleh *user* dan variabel _filter tersebut diproses dengan fungsi *runes.toList.reversed()* untuk membalik kata tersebut.
- Deklarasi *int n = text.length*, sebagai panjang *string* dari variabel *text*, kemudian *int m = pattern.length*, sebagai panjang variabel *pattern.*
- Deklarasi *int j=0;* sebagai awal dari proses pencocokan *string* yang dimulai dari sebelah kiri atau *array* ke-0;
- Deklarasi *int x=-1;* sebagai inisialisasi awal proses pencocokan *string*.
- Deklarasi *result = false;* sebagai keaadaan awal proses dan ketika terjadi ketidak cocokan pada hasil pencocokan *string*.
- Kemudian akan dilakukan perulangan jika kondisi *j<=n-1* dan *j* selalu ditambah satu.
- Kemudian dalam perulangan terdapat dua kondisi, yaitu kondisi pertama *x+1<=m-1* dan kondisi kedua *x == m-1*.
- Dalam kondisi *x+1<=m-1* terdapat sebuah kondisi jika benar *pattern [x+1] == text[j]* maka *x+1*, dan jika tidak maka *x=-1*.

- *Pattern* dinyatakan *match* dengan kata yang dicari apabila kondisi *x == m-1* sama dengan *true*, sehingga *result = true* yang berfungsi untuk menampilkan hasil pencarian dan variabel *jkata++* untuk menghitung jumlah kata yang ditemukan, serta *j=n-1* untuk mengakhiri proses.



Gambar 6. *Coding* Algoritma Simon

C. *Pengujian Algoritma Simon dalam Proses Pencarian Kata*

Pengujian ini dilakukan untuk menguji kecepatan sistem dalam melakukan proses pencarian sebuah kata berdasarkan *Pattern* yang dimasukan. Pengujian ini dibagi dalam dua hasil prosees, antaralain proses pencarian kata dengan *Pattern* dalam Bahasa Arab.

- Pencarian satu sample kata dalam beberapa Pattern

Tabel 11. Pengujian Pencarian Satu Kata

No	*Pattern*	Panjang *Pattern*	Waktu pencarian Ms (*milisecond*)
1	نَ	1	17,186
2	نَ	2	19,33
3	نَص	3	14,697
4	نَصَ	4	16,356
5	نَصَر	5	21,104
6	نَصَرَ	6	13,27
Rata-rata waktu *running time*			**16,9905**

- Pencarian 10 sample kata yang berbeda

Tabel 12. Pengujian Pencarian 10 Kata

No	*Pattern*	Panjang *Pattern*	Waktu pencarian Ms (*milisecond*)
1	مَدَ	4	25,477
2	صَانَ	5	38,474
3	غَزَا	5	22,331
4	أَمَلَ	6	14,357
5	ضَرَبَ	6	26,349
6	فَرَّ	4	28,512
7	وَعَدَ	6	21,013
8	يَسَرَ	6	26,392
9	سَارَ	5	22,605
10	شَوَى	5	20,873
Rata-rata waktu *running time*			**24,638**

- Pencarian 10 sample kata yang tidak ditemukan

Tabel 13. Pengujian Pencarian 10 Kata

No	*Pattern*	Panjang *Pattern*	Waktu pencarian Ms (*milisecond*)
1	يَفَعَ	6	37,784
2	وَجِيَ	6	20,928
3	بَئِسَ	6	30,212
4	حَصُنَ	6	28,014
5	جَبُنَ	6	31,795
6	يَمُنَ	6	34,868
7	سَرُوَ	6	19,043
8	أَدُبَ	6	30,344
9	لَؤُمَ	6	28,156
10	دَحْرَجَ	8	37,913
Rata-rata waktu *running time*			**29,9057**

- Rata-rata yang diperoleh dari pengujian 1-3 string Bahasa Arab

Tabel 14. Rata-rata running time pencocokan *string*

Jenis Pengujian	***Running Time* ms*(mili second)***
Pengujian 1	16,9905
Pengujian 2	24,638
Pengujian 3	29,9057
Rata-rata	**23,8447333**

IV. SIMPULAN

Berdasarkan analisis hasil pengujian maka dapat disimpulkan bahwa Algoritma Simon dapat diimplementasikan sebagai metode pencocokan *string* pada proses pencarian kata dalam kamus perubahan *Fi'il*, baik itu berupa kata dalam Bahasa Arab maupun Bahasa Indonesia. Adapun rata-rata *running time* yang dibutuhkan oleh Algoritma Simon dalam proses pencocokan string dalam penelitian ini adalah 23,8447333 *mili second* , sedangkan proses running time tercepat yang diperoleh proses running time tercepat yang diperoleh 13,27 *mili second* dan *running time* terlama 38,474 *mili second.*

Adapun saran untuk penelitian selanjutnya mengenai "Implementasi Algoritma Simon Pada

Aplikasi Kamus Perubahan *Fi'il* (Kata Kerja Bahasa Arab) Berbasis *Android*" yaitu :

1. Sistem aplikasi Kamus Perubahan *Fi'il* ini masih berbasis *offline*, sehingga ukuran aplikasi cukup besar, sehingga kurang efektif apabila ditambah lebih banyak lagi kata dan *audio* yang dimasukan.
2. Sistem penyimpanan data pada aplikasi ini hanya menggunakan data *JSON* dan tidak menggunakan *database*, sehingga proses *update* maupun *delete* dilakukan secara manual dalam program aplikasi, sehingga harus dilakukan *build* aplikasi kembali.

DAFTAR PUSTAKA

[1] Alkhathib, A. L. bin M. (2016). Al Khathib Ensiklopedia Komplit Menguasai Shorof Tashrif (1st ed.). Mitra Pustaka. pustakapelajar@yahoo.com

[2] Andriani, A. (2015). Urgensi Pembelajaran Bahasa Arab dalam Pendidikan Islam. Ta'allum: Jurnal Pendidikan Islam, 3(1), 39–56.https://doi.org/10.21274/taalum.2015.3.1.39-56.

[3] Chaliska, P. (2017). Perbandingan Algoritma String Matching On Ordered Alphabet dan Algoritma Simon pada Kamus Istilah Arsitektur. USU E-JOURNAL. http://repositori.usu.ac.id/handle/123456789/4112.

[4] Charras, C. and, & Lecroq, T. (2004). Handbook of Exact String Matching Algorithms. 238.

[5] Community, Ewolf. 2011. Buku Wajib Programmer: Indeks Lengkap Syntax Kumpulan Perintah-perintah Dasar Pemrograman yang Sering Digunakan Visual Basic, Delphi, C++, HTML, JavaScript, PHP, dan SQL. Yogyakarta: MediaKom.

[6] Eka Rinjani, R. (2012). Pemanfaatan Json (Javascript Object Notation) Sebagai Data Interchange Pada Sistem Automatic Testing Dan Web Based Learning D3 Teknik Informatika Uns Tugas.

[7] Kandir, N. (2017). Ringkasan Al-Amtsilah At-Tashrifiyah (1st ed.). Pustaka Syabab.

[8] Langgeni, Baizal & Firdaus. (2010). Clustering Artikel Berita Berbahasa Indonesia Menggunakan Unsupervised Feature Selection, Seminar Nasional Informatika, Yogyakarta.

[9] Megawaty. (2017). Implementasi dan Perbandingan Algoritma Brute Force dengan Algoritma Simon dalam Pembuatan Kamus Istilah Kebidanan. Universitas Sumatera Utara. http://repositori.usu.ac.id/handle/123456789/4660

[10] Parno, dharmayanti, N. (2011). Aplikasi Mobile Kamus Istilah_Ug.Pdf. http://repository.gunadarma.ac.id