APLIKASI PERMAINAN SUSUN KATA UNTUK PEMBELAJARAN BAHASA INGGRIS DENGAN ALGORITMA *KNUTH–MORRIS–PRATT* DAN *FISHER-YATES SHUFFLE*

[1]Suryo Hadi Sampurno, [2]A'la Syauqi, M.Kom.
[12]Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
email: suryaduasatu@gmail.com

Abstrak -Proses pembelajaran bahasa Inggris tentu perlu mempelajari tentang tata bahasa Inggris (*grammar*). Dengan definisi, rumus, dan contohnya yang cukup rumit, aplikasi komputer interaktif (*game*) yang memuat konten edukasi bahasa Inggris ini hadir sebagai hiburan yang mendidik. Penggunaan *game* susun kata sebagai sarana edukasi dikarenakan sifatnya yang interaktif sekaligus menghibur. Pemain diminta aktif untuk merangkai kata menjadi kalimat yang sesuai dengan tata bahasa tertentu. Permainan dibangun dengan mengimplementasikan metode *Knuth–Morris–Pratt* pada proses koreksi susunan kata dan metode *Fisher–Yates shuffle* pada proses membangkitkan permutasi susunan soal. Hasil uji coba metode *Knuth–Morris–Pratt* didapatkan hasil akurasi sebesar 100% dengan konsumsi waktu rata-rata adalah $4{,}133403 \times 10^{-4}$ detik menunjukkan proses koreksi susunan kata pada permainan berjalan efektif dari segi akurasi dan konsumsi waktu. Hasil uji coba metode *Fisher–Yates shuffle* didapatkan frekuensi permutasi dengan nilai rata-rata *range* sebesar 1,61% menunjukkan permutasi acak dari suatu himpunan terhingga terdistribusi secara hampir merata sehingga susunan soal yang ditampilkan permainan seimbang dan lebih bervariasi.

Kata Kunci: Bahasa Inggris, aplikasi komputer interaktif, *Knuth–Morris–Pratt*, *Fisher–Yates shuffle*

I. PENDAHULUAN

Bahasa Inggris saat ini merupakan bahasa yang paling umum digunakan di seluruh dunia. Bahasa ini telah digunakan secara luas pada bidang pengetahuan, pendidikan, perdagangan, politik, dan teknologi. Bahasa Inggris saat ini menjadi bahasa utama dan secara tidak resmi dianggap sebagai *lingua franca* di berbagai belahan dunia bagi yang menggunakannya sebagai bahasa ibu/utama, bahasa kedua, dan bahasa asing (Wardhaugh, 2010). Berdasarkan fakta tersebut mempelajari bahasa Inggris menjadi penting dan sebagai suatu kebutuhan. Melalui penguasaan bahasa Inggris, masyarakat diharapkan mampu berkomunikasi di kancah internasional serta dapat menguasai ilmu pengetahuan dan teknologi yang pada umumnya ditulis dalam bahasa Inggris.

Pada proses belajar bahasa Inggris tentu pelajar perlu mempelajari tentang tata bahasa Inggris (*grammar*). Pemahaman tentang tata bahasa dalam belajar bahasa Inggris sangat penting. Karena penggunaan bahasa yang benar harus mengikuti aturan tata bahasanya sehingga pelajar diharapkan memiliki keterampilan dan menjadi terlatih dalam membaca, menulis, berbicara, memahami, dan membuat serta menyusun kata dan kalimat secara benar. Namun tata bahasa Inggris semisal *grammatical tense* dengan segala definisi, rumus, dan contohnya seringkali dianggap sebagai kesulitan dalam belajar bahasa Inggris. Karena pada praktiknya pelajar seringkali menghafal banyak rumus disertai pembahasannya yang cukup rumit.

Untuk membantu pelajar mempelajari tata bahasa Inggris diluar pendidikan formal yang telah dilakukan maka pada penelitian ini dibuat sebuah Aplikasi komputer interaktif yang berbentuk *game.* Penggunaan *game* sebagai sarana edukasi dikarenakan sifatnya yang interaktif sekaligus menghibur. Dalam permainan, pembelajaran dilakukan lewat praktek

atau *learning by doing*. Sehingga konten edukasi tersampaikan kepada pemain dengan cara menyenangkan, serta tidak membosankan.

Aplikasi komputer interaktif ini dibuat dengan tujuan untuk memberikan suatu nilai edukasi kepada pemain. Aplikasi ini akan meminta pemain untuk aktif merangkai kata-kata menjadi kalimat yang sesuai dengan tata bahasa (*grammar*) tertentu. Sehingga secara tidak langsung akan melatih kemampuan membaca (*reading skill*) dan juga kemampuan menulis (*writing skill*) pemain.Diharapkan dengan adanya media alternatif pembelajaranberupa aplikasi komputer interaktif ini dapat berkontribusi terhadap peningkatan atau menumbuhkan minat pelajar dalam mempelajari tata bahasa Inggris.

Aplikasi berupa *game* tentang tata bahasa Inggris ini dibangun dengan meng-implementasikan metode *Knuth–Morris–Pratt* pada proses koreksi susunan kata dan metode *Fisher–Yates shuffle* pada proses membangkitkan susunan soal.Untuk menilai keberhasilan implementasi maka akan diukur akurasi dan konsumsi waktu terhadap implementasi metode *Knuth–Morris–Pratt* sebagai proses koreksi susunan kata dan diukur permutasi yang dihasilkan terhadap implementasi metode *Fisher–Yates shuffle* pada proses membangkitkan susunan soalpada aplikasi komputer interaktif.

II. TINJAUAN PUSTAKA

A. *Knuth-Morris-Pratt*

Algoritma *Knuth-Morris-Pratt*dipilih untuk diterapkan pada pembuatan aplikasi komputer interaktif ini dibandingkan dengan algoritma *string matching* lain karenamemberikan hasil yang efektif dari segi waktu komputasi(Soleh, 2010) dan (Lestari & Djaya, 2011) serta lebih cepat dibandingkan dengan algoritma *Brute Force*(Hadiati, 2007),(Pandiselvam.P et al., 2014) dan (Mulyana, 2014).

Knuth-Morris-Pratt(KMP) merupakan metode *exact string matching* yang menerapkan arah pencarian dari kiri ke kanan. metode *Knuth-Morris-Pratt* melakukan pencarian kata P terhadap teks T dengan menggunakan pengamatan bahwa ketika terjadi ketidakcocokan, kata itu sendiri memberikan informasi untuk menentukan dimana perbandingan berikutnya bisa dimulai, sehingga melewati pencocokan ulang karakter sebelumnya. Dengan demikian metode *KMP* melakukan pergeseran lebih jauh sesuai dengan informasi ketidakcocokan yang disimpan sehingga waktu pencarian dapat dikurangi secara signifikan. Perhatikan posisi karakter terjadinya ketidakcocokan untuk pertama pada teks berikut:

Contoh 1:

```
T =????????????????
P =aaabaaaxyz
  ^
```

Dapat dipastikan bahwa karakter sebelumnya *(prefix)* sama dengan *pattern* P

```
T =????aaab????????????
P =aaabaaaxyz
  ^
```

Kemudian gunakan informasi *prefix* ini untuk mencocokkan *string*. Maka dapat dipastikan jika *pattern* P tidak akan ditemukan jika pencocokan dimulai pada posisi berikut:

```
T =????aaab????????????
P =aaab??????
  ^
T =????aaab????????????
P =aaab??????
  ^
T =????aaab????????????
P =aaab??????
  ^
```

Kemungkinan pertama ditemukan kecocokan adalah jika pencocokan dimulai pada posisi berikut:

```
T =????aaab????????????
P =     --->aaab??????
  ^(4 char slide)
```

Berdasarkan contoh 1 dapat diambil kesimpulan bahwa proses pencocokan dapat digeser sebanyak 4 karakter tanpa melakukan pencocokan yang tidak diperlukan dan melewatkan satupun pola kecocokan *pattern* P.

Metode *Knuth-Morris-Pratt* memberikan hasil yang efektif dari segi waktu komputasi karena proses pencocokan tidak selalu dilakukan dengan menggeser ke kanan 1 karakter. Untuk melakukan pencarian yang efektif metode ini memiliki dua poin utama yaitu *prefix function* dan *KMP matcher*:

- *Prefix fuction* menyimpan informasi tentang kecocokan *pattern* dan informasi ini digunakan untuk menghindari pergeseran yang tidak diperlukan *pattern*. Berikut ini adalah *pseudocode prefix function* (Cormen et al., 2009):

```
COMPUTE-PREFIX-FUNCTION
(P)
1 m = P.length
2 let π[1..m] be a new
array
3 π[1] = 0
4 k = 0
5 for q = 2 to m
6       while k > 0 and
P[k+1] ≠ P[q]
7           k = π[k]
8     if P[k+1] == P[q]
9           k = k+1
10    π[q] = k
11 returnπ
```

- *KMP matcher* menggunakan *prefix function, pattern* dan teks sebagai *input* untuk menemukan *pattern* pada teks dan mengembalikan jumlah pergeseran setelah ditemukan pola pertama. Berikut ini adalah *Pseudocode KMP matcher* (Cormen et al., 2009):

```
KMP-MATCHER (T,P)
1 n = T.length
2 m = P.length
3 π = COMPUTE-PREFIX-
FUNCTION (P)
4 q = 0       //number of
characters matched
5 for i = 1 to n //scan
the text from left to
right
6     while q>0 and
P[q+1]≠ T[i]
7         q = π[q]// next
character does not match
8     if P[q+1] == T[i]
9         q = q + 1 //
next character matches
10    if q == m    // is
all of P matched?
11        print "Pattern
occurs with shift" i - m
12        q = π[q]  //look
for the next match
```

B. Fisher-Yates shuffle

Fisher-Yates shuffle atau juga disebut sebagai *Knuth shuffle* adalah sebuah metode untuk menghasilkan sebuah permutasi acak dari suatu himpunan terhingga. Metode *Fisher-Yates shuffle* memungkinkan setiap permutasi yang dihasilkan memiliki kemungkinan yang sama *(unbiased).* Metode ini memiliki *time complexcity* O(n) (Black, 2005). Berikut ini adalah *pseudocode*:

```
1 To shuffle an array a of
n elements (indices 0..n-
1):
2  for i from n − 1 downto
1 do
3        j ← random integer
with 0 ≤ j ≤ i
4        exchange a[j] and
a[i]
```

Berikut ini adalah contoh pengerjaan untuk permutasi angka 1-8:

Range	Roll	Scratch	Result
		1 2 3 4 5 6 7 8	

Pertama kita pilih angka acak 1-8, kita pilih 6 kemudian kita menukar angka urutan keenam yaitu angka 6 dengan angka urutan kedelapan yaitu angka 8

Range	Roll	Scratch	Result
1-8	6	1 2 3 4 5 **8** 7	**6**

Angka acak selanjutnya dari 1-7, kita pilih 2. Kita tukar angka urutan kedua dan ketujuh

Range	Roll	Scratch	Result
1-7	2	1 **7** 3 4 5 8	**2** 6

Angka acak berikutnya dari 1-6, 1-5 dan seterusnya. Sehingga dengan mengulangi langkah-langkah seperti diatas akan didapatkan hasil sebagai berikut:

Tabel 1. Pengerjaan metode *Fisher–Yates shuffle*

Range	Roll	Scratch	Result
1-6	6	1 7 3 4 5	**8** 2 6
1-5	1	**5** 7 3 4	**1** 8 2 6

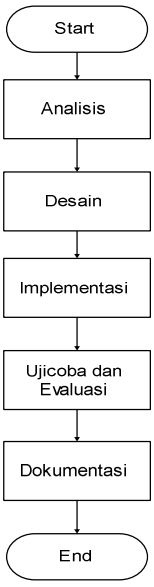
1-4	3	5 7 **4**	**3** 1 8 2 6
1-3	3	5 7	**4** 3 1 8 2 6
1-2	1	**7**	**5** 4 3 1 8 2 6

Permutasi yang dihasilkan adalah 7 5 4 3 1 8 2 6.

III. METODE PENELITIAN

A. *Prosedur Penelitian*

Tahap awal dilakukan Analisis meliputi analisis literatur, analisis permainan, analisis *input*, analisis proses, dan analisis *output*. Kemudian tahap Desain meliputi desain *use case diagram,* desain *activity diagram,* desain *class diagram* dan desain *interface*. Dilanjutkan tahap Implementasi meliputi implementasi *interface,* implementasi metode *Knuth–Morris–Pratt* dan implementasimetode *Fisher–Yates shuffle*. Selanjutnya tahap Uji coba dan Evaluasi meliputi uji coba metode *Fisher-Yates shuffle* dan uji coba metode *Knuth-Morris-Pratt*, kemudian evaluasi berdasarkan hasil uji coba. Tahap terakhir adalah Dokumentasiyang merupakan kegiatan penulisan laporan sebagai dokumentasi dari keseluruhan tahapan yang telah dilaksanakan.



Gambar 1.*Flowchart* prosedur penelitian

B. *Analisis Permainan*

Konsep dari permainan ini adalah pemain diberikan sebuah *field* permainan berupa kotak-kotak sebanyak 54 (6 x 9) kotak dan memuat *subject, verb, noun, pronoun* dan lain sebagainya. Pemain akan diminta untuk menyusun kata-kata/membuat kalimat bahasa Inggris berdasarkan soal yang muncul. Pemain melakukan klik pada kotak yang ada, apabila kalimat yang dibuat benar maka pemain akan mendapatkan skor.Berdasarkan konsep permainan tersebut kemudian diimplementasikan metode *Knuth–Morris–Pratt* pada proses koreksi susunan kata dengan membandingkan kalimat jawaban pemain terhadap kunci jawaban yang tersimpan dalam *database.*

C. *Analisis Input*

Database soal dan jawaban yang digunakan merupakan kalimat-kalimat yang mempunyai struktur tata bahasa Inggris. Tata bahasa Inggris yang digunakan pada permainan ini antara lain: *Simple Present, Simple Present Continuous, Simple Past,* dan *Simple Future (will).Database* soal dan jawaban berupa dokumen teks (.txt) mempunyai format: **kalimat-bahasa-inggris;arti-kalimat**.Diantara kalimat bahasa Inggris dan arti kalimat dipisahkan oleh tanda *semicolon (;)*. Antar kata pada kalimat bahasa Inggris dan arti kalimat dipisahkan oleh *dash (-)*.

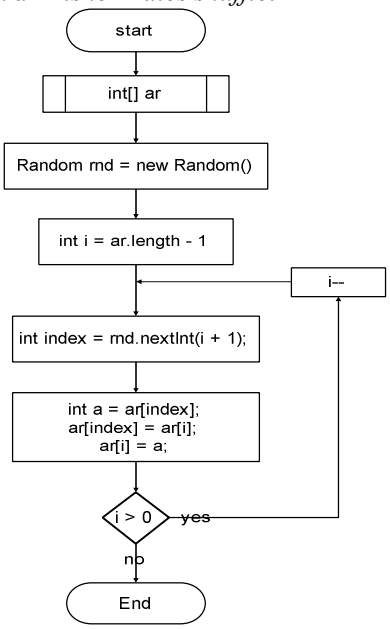
D. *Analisis proses permutasi*

Permutasi adalah proses mengubah letak urutan dari suatu grup/himpunan dengan memperhatikan urutan. Apabila urutan diperhatikan dan setiap objek yang tersedia hanya bisa dipilih atau dipakai sekali maka jumlah permutasi yang ada dapat dihitung dengan rumus:

$$P(n,r) = \frac{n!}{(n-r)!}$$

di mana *n* adalah jumlah objek yang dapat dipilih, *r* adalah jumlah yang harus dipilih dan *!* adalah simbol faktorial. Proses permutasi pada aplikasi digunakan untuk menentukan soal-soal yang akan ditampilkan pada *field* permainan sehingga soal yang ditampilkan lebih bervariasi dan seimbang. Metode yang

diimplementasikan pada proses ini adalah *Fisher-Yates shuffle*.



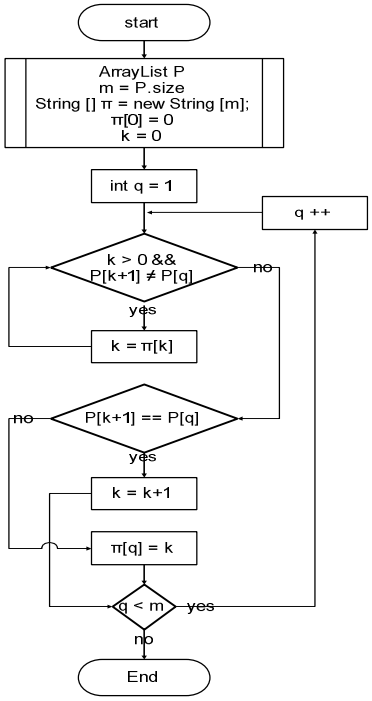
Gambar 2.*Flowchart* proses permutasi metode *Fisher-Yates Shuffle*

E. Analisis proses koreksi

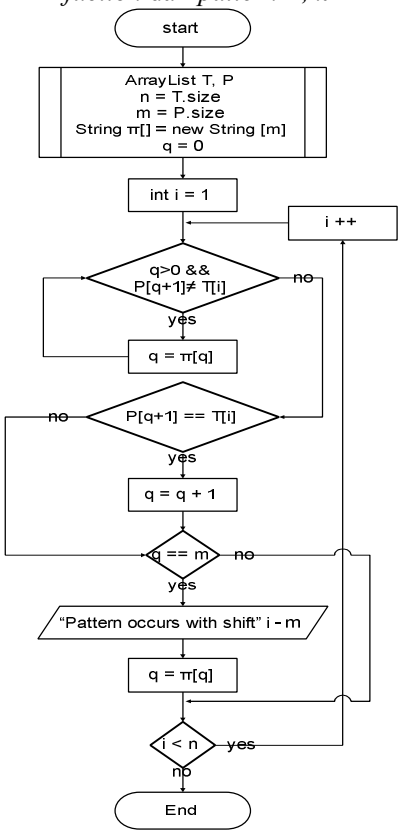
Proses koreksi susunan kalimat menggunakan metode *Knuth–Morris–Pratt*melibatkan dua parameter yaitu *pattern* P (kalimat) yang akan dicari dan *text* T yang merupakan *database*serta melakukan dua proses yaitu fase *preprosessing* dan fase pencarian.

Pada fase *preprosessing* metode *KMP* membuat *temporarry array**Prefix fuction* dari *pattern*`P, π. π` merupakan sebuah *array* berukuran (`P.length`) dimana untuk setiap `π[i]`terhadap `P, π[i]` didefinisikan sebagai 'panjang terpanjang dari *proper suffix*`P[i]`' yang cocok dengan '*properprefix* dari `P[i]`'.

Pada fase pencarian kita mengkomparasikan karakter pertama dari teks `T T[i]`dengan karakter pertama dari *pattern* `P P[i]`.Jika ditemukan kesamaan maka nilai `i`dan`q` akan dinaikkan. Jika ditemukan ketidaksamaan lalu kita lihat pada `Π`untuk menggeser posisi dari `P` sebanyak nilai yang didasarkan pada perhitungan *Prefix fuction* dari *pattern*`P, Π`.

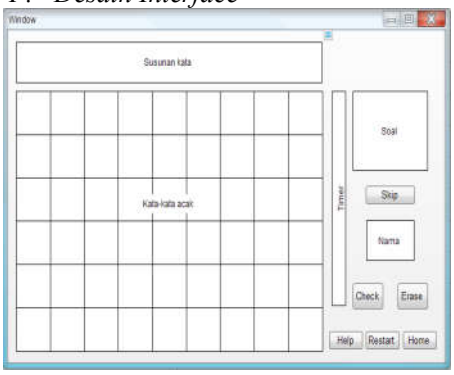


Gambar 3.*Flowchart*Fase *preprosessing:* menghitung *Prefix fuction* dari *pattern P, π*



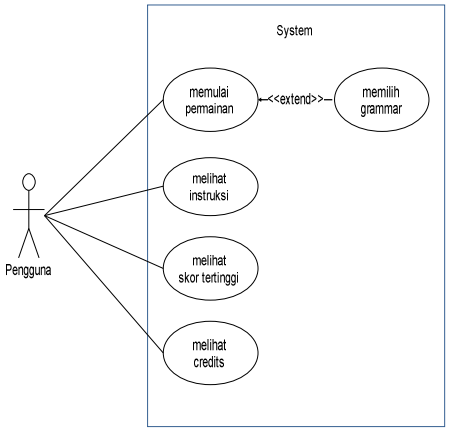
Gambar 4.*Flowchart*Fase pencarian

F. *Desain Interface*



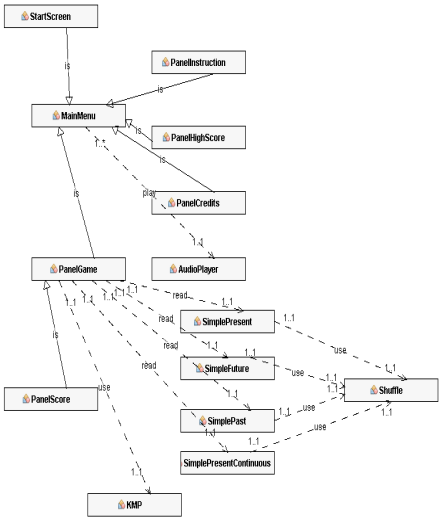
Gambar 5. Desain *frame*permainan

G. *Desain Use Case Diagram*



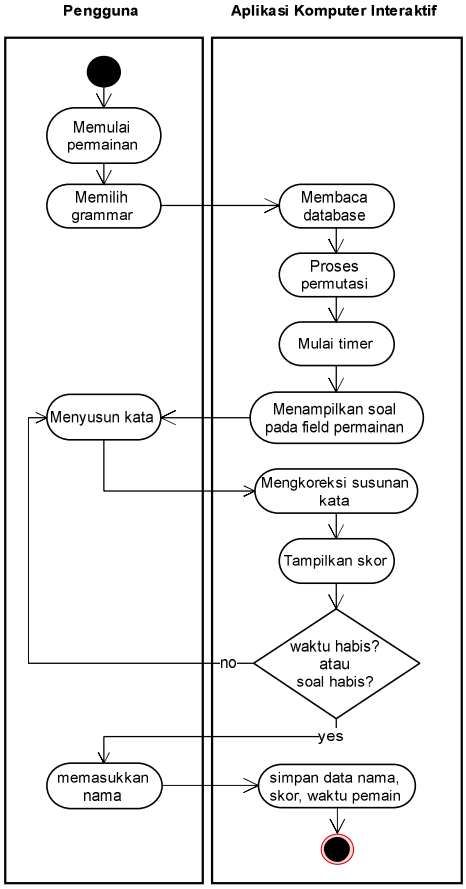
Gambar 6. *Use Case Diagram* untuk interaksi antara aktor dan sistem

H. *Desain Class Diagram*



Gambar 7. *Class Diagram*aplikasi

I. *Desain Activity Diagram*



Gambar 8.*Activity diagram* aplikasi

IV. HASIL DAN PEMBAHASAN

A. *Implementasi Interface permainan*

Gambar 9. *Frame* Permainan

B. *Uji Coba Metode Fisher-Yates shuffle*

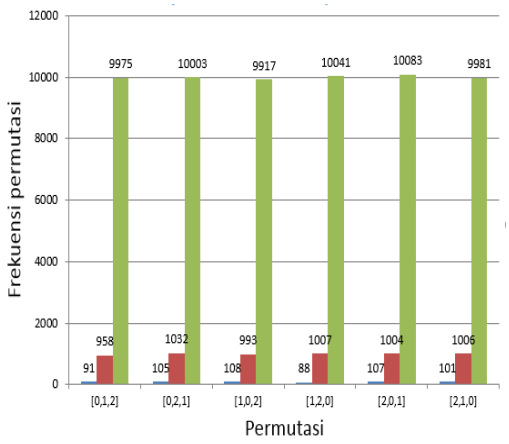
Frequency analysis, dilakukan untuk mengetahui bahwa metode *Fisher-Yates shuffle* menghasilkan sebuah permutasi acak dari suatu himpunan terhingga (*array*) yang

memiliki kemungkinan yang sama *(unbiased)*.Menggunakan *array*`a[] = {0, 1, 2}` sebagai *array* uji coba, kemudian dilakukan perulangan permutasi sebanyak 600, 6.000 dan 60.000 kali.Hasil uji coba permutasi metode*Fisher-Yates shuffle* akan dibandingkan dengan hasil permutasi yang didapatkan dengan menggunakan fungsi yang terdapat pada *java* untuk melakukan permutasi acakyaitu `Collections.shuffle(Array)`.

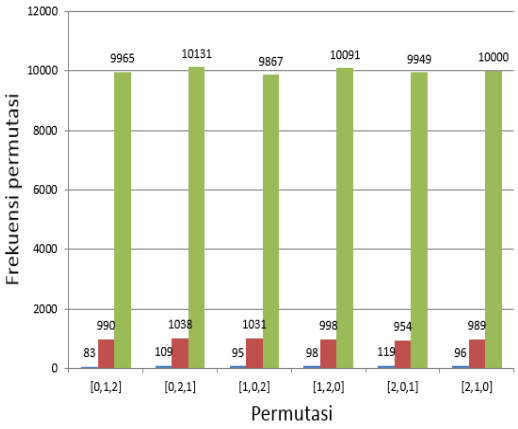
Pada percobaan metode*Fisher-Yates shuffle*(**Gambar 10**) dengan jumlah perulangan 600 kali didapat frekuensi permutasi dengan *range* sebesar 20 (nilai frekuensi *max* – nilai frekuensi *min*) atau 3.33% $\left(\frac{\text{nilairange}}{\text{jumlah perulangan}}\right)$, 6.000 perulangan sebesar 74 (1.23%) dan 60.000 perulangan 166 (0.28%) dengan nilai rata-rata *range* sebesar 1,61%. Sedangkan pada percobaan permutasi `Collections.shuffle()` (**Gambar 11**) dengan jumlah perulangan 600 kali didapat frekuensi permutasi dengan *range* sebesar 36 (6.00%), 6.000 perulangan sebesar 84 (1.40%) dan 60.000 perulangan 264 (0.44%) dengan nilai rata-rata *range* sebesar 2,61%. Frekuensi permutasi yang dihasilkan metode *Fisher-Yates shuffle* memiliki nilai rata-rata *range* lebih kecil (1,00%) jika dibandingkan dengan nilai rata-rata *range* fungsi `Collections.shuffle()` yang terdapat pada *java*.

C. *Uji coba Knuth–Morris–Pratt*

Pencarian, proses ini dilakukan sebagai simulasi koreksi susunan kalimat dengan membandingkan antara jawaban pemain terhadap kunci jawaban yang tersimpan dalam *database*. *Database* soal permainan berupa file teks akan dibaca dan ditampung pada struktur data *ArraList* sebagai teks `T`. Uji coba dilakukan terhadap *database* untuk soal *Simple Present, Simple Present Continuous, Simple Past, Simple Future (will)* dan dilakukan sebanyak 50 kali untuk masing-masing tipe soal dengan kondisi *pattern*(kalimat) `P` ditentukan secara acak. Uji coba digunakan untuk mengukur akurasi dan konsumsi waktu metode.



Gambar 10. Grafik frekuensi permutasi metode *Fisher-Yates shuffle*

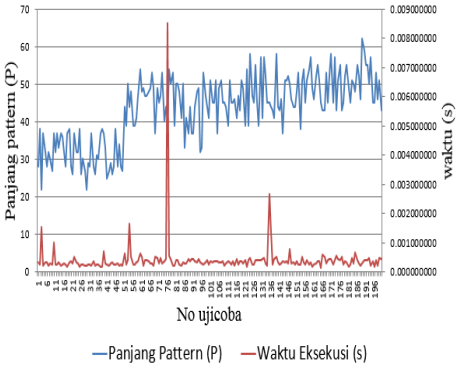


Gambar 11. Grafikfrekuensi permutasi `Collections.shuffle()`

Hasil uji coba akan didapatkan tabel dengan data berupa:

No	Panjang Pattern (P)	Panjang Teks (T)	Jumlah Pergeseran (i0)	Waktu Eksekusi (s)

Dari hasil 200 kali percobaan pencarian didapatkan grafik konsumsi waktu pencarian dapat dilihat pada gambar 12.



Gambar 12. Grafikkonsumsi waktu pencarian metode *Knuth–Morris–Pratt*

Panjang *pattern* P didapatkan dengan menghitung jumlah karakter pada *pattern* yang terpilih secara acak dengan teks T yang tidak berubah. Akurasi pada tabel uji coba diperoleh dengan melihat nilai kembalian *methodKMP_matcher(T,P)* sebagai`i0`(kolom Jumlah Pergeseran (i0)).Apabila nilai kembalian bernilai `-1`, artinya *pattern*`P` tidak ditemukan pada teks `T`. Apabila *pattern*`P` ditemukan pada teks `T` maka nilai kembalian `i0` bernilai bilangan cacah (0, 1, 2, ...). Nilai `i0` menunjukkan total banyaknya pergeseran terhadap teks `T` ketika *pattern*`P` ditemukan. Waktu eksekusi diperoleh dengan menghitung nilai selisih antara `endTime - startTime`pada kode uji coba`.`Rata-rata waktu eksekusi pada tabel hasil uji coba diperoleh dengan rumus sebagai berikut:

$$\text{Rata-rata waktu eksekusi} = \frac{\text{Total waktu}}{\text{Jumlah percobaan}}$$

Berdasarkan hasil uji coba yang telah dilakukan didapatkan hasil akurasi sebesar 100%. Akurasi sebesar 100% dapat diperoleh karena metode *Knuth–Morris–Pratt* melakukan pencocokan string secara tepat *(Exact String Matching)*. Metode *string matching* ini mencari *string* pada teks yang sama persis dengan *string* masukan. Kemudian didapatkan konsumsi waktu pencarian yang dibutuhkan aplikasi rata-rata adalah 0,0004133403 ($4{,}133403 \times 10^{-4}$) detik.

V. PENUTUP

A. *Kesimpulan*

Berdasarkan hasil penelitian Aplikasi Permainan Susun Kata Untuk Pembelajaran Bahasa Inggris dengan Algoritma *Knuth–Morris–Pratt* dan *Fisher-Yates Shuffle* didapatkan kesimpulan sebagai berikut:

1. Hasil uji coba metode *Knuth–Morris–Pratt* menunjukkan bahwa implementasi metode *Knuth–Morris–Pratt* sebagai proses koreksi susunan kata pada aplikasi komputer interaktif bersifat efektif. Berdasarkan hasil uji coba yang telah dilakukan didapatkan hasil akurasi sebesar 100%. Kemudian didapatkan konsumsi waktu pencarian yang dibutuhkan aplikasi rata-rata adalah 0,0004133403 ($4{,}133403 \times 10^{-4}$) detik. Nilai akurasi dan konsumsi waktu yang didapatkan menunjukkan bahwa implementasi metode ini dapat berjalan maksimal dari segi akurasi dan kecepatan (konsumsi waktu < 0,5 detik).
2. Hasil uji coba metode *Fisher–Yates shuffle*menunjukkan bahwa implementasi metode *Fisher–Yates shuffle* pada proses membangkitkan susunan soalpada aplikasi komputer interaktif bersifat efektif. Frekuensi permutasi yang dihasilkan metode *Fisher-Yates shuffle* memiliki nilai rata-rata *range* sebesar 1,61%. Nilai tersebut lebih kecil (1,00%) jika dibandingkan dengan nilai rata-rata *range* fungsi `Collections.shuffle()` yang terdapat pada *java* yaitu sebesar 2,61%. Nilai tersebut membuktikan implementasi metode ini menghasilkan permutasi acak dari suatu himpunan terhingga dengan distribusi secara hampir merata pada sejumlah perulangan permutasi yang dilakukan. Hal tersebut menunjukkan bahwa susunan soal yang ditampilkan oleh aplikasi seimbang. Tidak ada susunan soal yang lebih mendominasi kemunculannya dari susunan soal

lain sehingga soal yang ditampilkan permainan lebih bervariasi.

B. Saran

Pada Penelitian Aplikasi Permainan Susun Kata Untuk Pembelajaran Bahasa Inggris dengan Algoritma *Knuth–Morris–Pratt* dan *Fisher-Yates Shuffle* menggunakan materi edukasi tata bahasa Inggris dalam bentuk kalimat positif dari *Simple Present, Simple Present Continuous, Simple Past, dan Simple Future (will)*. Sehingga masih memungkinkan untuk dilakukan pengembangan lebih lanjut dengan materi edukasi 16 *tenses*yang lebih lengkap, atau menambah materi lain seperti *Vocabulary*, *Spelling*, dan lain-lain.

VI. DAFTAR PUSTAKA

[1] Andriasnyah, 2014. *Perancangan Aplikasi Game Edukasi Menggunakan Metode Linear Congruent Method (LCM)*. Medan: Pelita Informatika Budi Darma, Volume : VI, Nomor: 1 ISSN : 2301-9425.

[2] Azhari & Fatta, H.A., 2014. *Pembuatan Game Design Document 'Vocab Mania' (Menyusun Gambar Dengan Kosa Kata Bahasa Inggris) Berbasis Android*. Yogyakarta: Sekolah Tinggi Manajemen Informatika dan Komputer AMIKOM.

[3] Black, P.E., 2005. *Fisher-Yates shuffle*. [Online] Available at: HYPERLINK "http://xlinux.nist.gov/dads/HTML/fisherYatesShuffle.html" http://xlinux.nist.gov/dads/HTML/fisherYatesShuffle.html [Accessed 5 April 2015].

[4] Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C., 2009. *Introduction to Algorithms*. 3rd ed. Massachusetts: The MIT Press.

[5] Frihardianto, Y., 2012. *Penerapan Algoritma Generate And Test Pada Permainan Scrabble*. Yogyakarta: Undergraduate thesis, Duta Wacana Christian University. Retrieved from http://sinta.ukdw.ac.id.

[6] Hadiati, D., 2007. *Penerapan Algoritma String Matching pada Permainan "Word Search Puzzle"*. Bandung: Makalah IF2251 Strategi Algoritmik.

[7] KEMENDIKNAS, 2011. *Standar Kompetensi Lulusan*. Direktorat Pembinaan Kursus & Pelatihan.

[8] Lestari, S. & Djaya, A., 2011. *Aplikasi Search Engine Menggunakan Algoritma Knuth-Morris-Pratt (KMP)*. Surabaya: Prosiding Seminar Nasional Manajemen Teknologi XIII.

[9] Masya, F. & Elvina, 2010. *Pengembangan Aplikasi Permainan Scrabble Dua Bahasa Menggunakan Java*. Jakarta: Teknik Informatika, Universitas Mercu Buana.

[10] Muhammad, D.A.b., 2003. *Tafsir Ibnu Katsir Jilid 5*. Bogor: Pustaka Imam asy-Syafi'i.

[11] Mulyana, F., 2014. *Penerapan Algoritma Knuth-Morris-Pratt Pada Game Puzzle Untuk Mencari Kecocokan Pola Warna*. Bandung: Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia.

[12] Pandiselvam.P, Marimuthu.T & Lawrance.R, 2014. *A Comparative Study on String Matching Algorithm of Biological Sequences*. Sivakasi: Department of Computer Applications, Ayya Nadar Janaki Ammal College, India.

[13] Rahmawati, S.R., 2007. *Panduan Lengkap Menguasai English Grammar*. Jakarta: PT Kawan Pustaka.

[14] Soleh, M.Y., 2010. *Implementasi Algoritma KMP dan Boyer-Moore dalam Aplikasi Search*

Engine Sederhana. Bandung: Makalah IF3051 Strategi Algoritma.

[15] Wardhaugh, R., 2010. *An Introduction to Sociolinguistics*. 6th ed. Oxford: Blackwell.