Pengecekan Ejaan Berperforma Tinggi pada Editor Teks dengan Strategi *Decrease and Conquer*

Bryan Amirul Husna - 13520146
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520146@std.stei.itb.ac.id

***Abstract*—Pengecekan ejaan adalah salah satu fitur dalam editor teks yang dapat membantu penulis mendeteksi kesalahan penulisan ejaan (*typo*). Diperlukan algoritma yang efisien dalam pencocokkan suatu kata yang sedang diperiksa dengan daftar kata yang terdapat dalam kamus. Pendekatan *decrease-and-conquer* menghasilkan algoritma logaritmik yang dapat mencocokkan keberadaan suatu kata di dalam kamus. Algoritma telah diterapkan dalam suatu program dan berhasil memberikan respons yang benar sesuai kamus masukan. Peningkatan yang dapat dilakukan adalah dengan pengecekan awalan (me-, di-, ter-, dsb.) dan akhiran (-i, -kan, dsb.) karena algoritma dan program hanya memeriksa secara literal yang tertulis di kamus.**

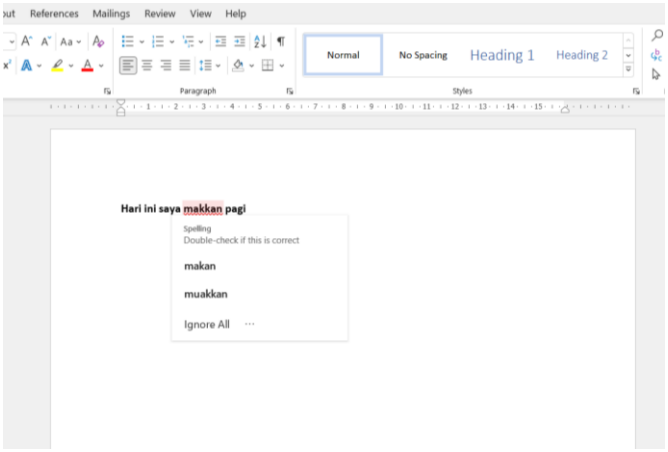
Kata kunci—decrease-and-conquer; ejaan; editor teks

I. PENDAHULUAN

Menurut Kamus Besar Bahasa Indonesia [1], satuan bahasa yang secara relatif berdiri sendiri, mempunyai pola intonasi final dan secara aktual atau potensial terdiri atas klausa disebut kalimat. Klausa merupakan unit gramatikal yang memiliki predikat. Kata merupakan elemen bahasa yang diucapkan atau dituliskan, merupakan perwujudan kesatuan perasaan dan pikiran yang digunakan dalam berbahasa. Ejaan merupakan kaidah penggambaran bunyi-bunyi (kata, kalimat, dan sebagainya) dalam bentuk tertulis (huruf-huruf) dan pemakaian tanda baca.

Pengecekan ejaan (*spell checking*) adalah salah satu fitur dalam aplikasi pemrosesan kata/editor teks yang dapat memeriksa dan menunjukkan letak kesalahan pengetikan pada suatu kata yang dilakukan oleh pengguna. Ukuran kamus kumpulan kata berukuran besar sehingga perlu strategi khusus agar pengecekan kata dalam kamus lebih cepat. Pengecekan yang cepat dan efisien diperlukan agar pengguna tidak terganggu dengan fitur ini ketika sedang mengetik.

Dalam makalah ini, digunakan pendekatan *decrease-and-conquer* untuk merumuskan algoritma yang dapat memvalidasi kata sesuai daftar kamus masukan. Hanya akan dibahas pengecekan ejaan kata, tanpa adanya koreksi terhadap ejaan tersebut. Pengecekan ejaan bertujuan untuk memeriksa kecocokan kata masukan dengan kamus, sedangkan koreksi ejaan memperbaiki kata dengan yang mirip di kamus (pengguna mungkin diberikan beberapa pilihan). Koreksi ejaan tidak dibahas karena makalah berfokus pada pendeteksian ketidakbenaran kata yang diketikkan. Koreksi ejaan memerlukan algoritma yang lebih kompleks dari segi waktu karena memerlukan pengecekan *edit distance* sehingga pengecekan dan koreksi sebaiknya dipisah. Skenario yang mungkin adalah kata-kata diperiksa dahulu dengan cek ejaan, kemudian menggunakan algoritma pengoreksi kata untuk membenarkan kata telah terdeteksi tidak benar sehingga tidak setiap kata perlu diperiksa dengan algoritma koreksi yang mahal.



Gambar 1. Fitur pengecek ejaan dan sugesti perbaikan pada aplikasi pemrosesan kata Microsoft Word
Sumber: dokumen penulis

II. LANDASAN TEORI

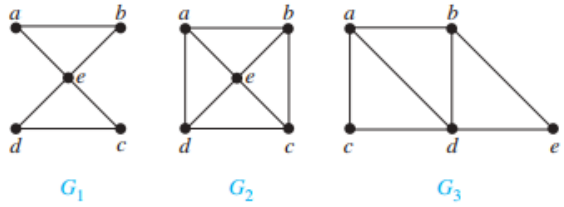
A. *Strategi Decrease-and-Conquer*

Decrease-and-conquer merupakan salah satu pendekatan penyelesaian masalah dengan memanfaatkan hubungan antara solusi dari permasalahan dan solusi dari pecahan masalahnya yang lebih kecil. Teknik ini bekerja dengan mengurangi suatu *instance* persoalan menjadi *instance* yang lebih kecil pada tiap tahapan. Terdapat tiga variasi umum strategi ini, yaitu mengurangi dengan konstanta (*decrease by a constant)*, mengurangi dengan faktor konstan (*decrease by a constant factor*), dan mengurangi dengan perubahan yang tidak tetap (*variable size*) [2].

Berbeda dengan pendekatan *divide-and-conquer*, pada *decrese-and-conquer* tidak terdapat tahap kombinasi subpersoalan. Ini karena hanya satu subpersoalan yang diproses dan diselesaikan. Maka, pendekatan ini hanya terdiri atas dua tahap, yaitu *decrease* (mengecilkan persoalan menjadi beberapa persoalan yang lebih kecil) dan *conquer* (menyelesaikan satu subpersoalan) [3].

B. *Graf*

Graf adalah pasangan himpunan (V, E), ditulis dengan notasi G = (V, E), yang dalam hal ini V adalah himpunan tidak-kosong dari simpul-simpul dan E adalah himpunan sisi-sisi yang menghubungkan sepasang simpul. Tiap sisi mungkin memiliki angka tertentu yang dapat merepresentasikan biaya, jarak, waktu tempuh, dan sebagainya. Graf yang demikian disebut sebagai graf berbobot. Suatu graf mungkin memiliki sisi ganda atau bisa juga memiliki sisi gelang. Berdasarkan keberadaan arah pada sisinya, graf bisa tergolong berarah atau tidak berarah [4].



Gambar 2. Contoh-contoh graf
Sumber: Discrete Mathematics and Its Application [4]

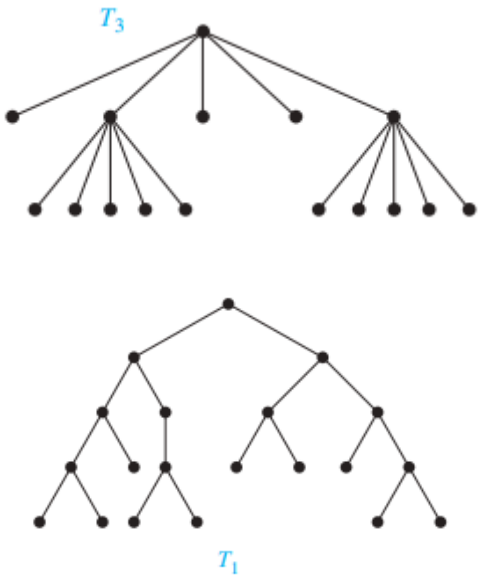
C. *Pohon*

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Sebuah graf tak berarah adalah pohon jika dan hanya jika ada lintasan unik antara dua sebarang simpul. Pohon digunakan sebagai model untuk berbagai bidang seperti psikologi, geologi, kimia, dan ilmu computer. Pohon berakar adalah pohon yang salah satu simpulnya ditetapkan sebagai akar dan tiap sisi terarah keluar dari akar. Sebuah simpul pohon disebut daun jika tidak memiliki anak, sedangkan yang memiliki anak disebut simpul dalam [4].

D. *Pohon Penuh dan Pohon Seimbang*

Suatu pohon disebut *m*-ary penuh jika tiap simpul dalamnya tepat memiliki m buah anak. Suatu pohon dikatakan seimbang jika semua simpul daunnya terletak pada kedalaman h atau h-1. Ini berarti tiap daun akan memiliki lintasan yang panjangnya kira-kira sama. Pohon *m*-ary penuh dengan banyak simpul daunnya adalah *l* akan memiliki *n* buah simpul yang dapat dihitung dengan rumus berikut [4].

$$n = (ml - 1) / (m - 1)$$

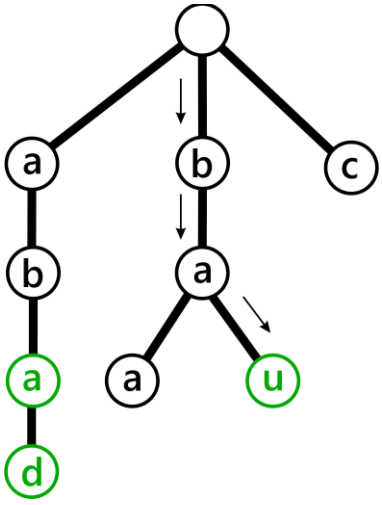


Gambar 3. (*T3*) Pohon 5-ary penuh (m = 5);
(*T1*) Pohon seimbang
Sumber: Discrete Mathematics and Its Application [4]

III. HASIL DAN PEMBAHASAN

A. *Algoritma*

Digunakan struktur data pohon untuk merepresentasikan kamus kata. Tiap simpul memiliki info berupa karakter yang termuat di dalamnya, boolean sebagai indikasi apakah simpul ini merupakan huruf akhir dari kata yang valid, dan kumpulan simpul anak. Kata disusun dan dicek satu huruf per satu huruf dari akar melewati simpul-simpul. Kata yang valid adalah kata yang huruf terakhirnya berada pada suatu simpul dengan boolean penanda kata valid bernilai *true*.



Gambar 4. Contoh *instance* struktur data yang digunakan dan penurunan kata “bau”. Simpul hijau berarti kata yang berakhir pada simpul tersebut adalah kata yang valid.
Sumber: dokumen penulis

Pengecekan ejaan dilakukan dengan memeriksa kata masukan terhadap kamus. Kamus berbentuk simpul masukan yang merupakan akar dari seluruh simpul yang telah dibangun sebelumnya. Kamus bisa dibangun secara manual dalam kode (menambahkan kata satu per satu melalui kode) atau berasal dari pembacaan suatu *file*. Garis besar algoritma yang digunakan dalam pengecekan sebagai berikut:

$$check(word, node) = \begin{cases} false, & jika\ node = null \\ node.isvalidword, & jika\ length(word) = 0 \\ check(substring(word, 1), node.findchild(word[0])), & otherwise \end{cases}$$

```
function checkWord(word : string, node :
Node) → bool
KAMUS
  n : Node
  i : integer
ALGORITMA
  if(node = NIL) then
    → false
  else if(length(word) = 0) then
    → node.isvalidword
  else
    i = 0
    while(node.children[i].info ≠ word[0])
      i++
    → checkWord(word[1:], node.children[i])
```

Gambar 5. *Pseudocode* algoritma *decrease-and-conquer* untuk pengecekan ejaan kata
Sumber: dokumen penulis

Algoritma bekerja dengan cara mengeliminasi kata-kata dalam kamus yang sudah tidak mungkin cocok dengan kata masukan. Pada tiap kedalaman, yang perlu dievaluasi hanyalah simpul anak yang bersesuaian dengan huruf pada kata. Suatu kata memiliki salah ejaan jika tidak terdapat dalam kamus, yaitu jika tidak ada simpul anak dengan huruf yang bersesuaian (node = NIL) atau kata berakhir tidak pada simpul yang merupakan akhiran kata valid, misalnya tidur adalah kata valid, tetapi tidu yang merupakan awalannya tidak valid. Namun, suatu kata tidak perlu berakhir di daun agar menjadi valid, misalnya "maka" yang merupakan awalan dari "makan" adalah valid, sehingga tiap simpul perlu menyimpan informasi apakah kata yang berakhir pada simpul tersebut valid.

Misal m adalah panjang kata yang akan diperiksa dan n adalah banyak kata dalam kamus. Dengan asumsi pohon adalah penuh dan seimbang (semua daunnya terletak pada kedalaman k = $^{26}\log n$ atau dengan kata lain semua kata panjangnya k), n bernilai perpangkatan dari 26 (jumlah alfabet), dan m = (k = $^{26}\log n$), kompleksitas waktunya adalah sebagai berikut.

$$T(m,n) = \begin{cases} a, & jika\ m = 0\ atau\ n = 1 \\ T(m-1, n/26) + 26c, & otherwise \end{cases}$$

Penyelesaiannya adalah

$$\begin{aligned} T(m, n) &= T(m\text{-}1, n/26^1) + 26c \\ &= T(m\text{-}2, n/(26^2)) + 26c + 26c \\ &= T(m\text{-}3, n/(26^3)) + 3{\times}26c \\ &\cdots\cdots \\ &= T(m\text{-}k, n/(26^k)) + k{\times}26c \end{aligned}$$

jika k = $^{26}\log n$, maka

$$\begin{aligned} T(m, n) &= T(0, 1) + {}^{26}\log n \times 26c \\ &= a + {}^{26}\log n \times 26c \end{aligned}$$

Maka, kompleksitas waktu dari algoritma ini adalah **O(m + log(n))**. Pada perhitungan diasumsikan m = k. Jika m < k, algoritma akan berhenti lebih cepat daripada log(n) karena tidak perlu menunggu n terbagi menjadi 1 sehingga yang membatasi adalah m (atau O(m)). Jika m > k, n akan terbagi menjadi 1 terlebih dahulu sebelum m menjadi 1 sehingga yang membatasi adalah log(n) (atau O(log(n))). Algoritma ini termasuk dalam *decrease by constant factor* dengan faktornya adalah 26, tetapi dapat tergolong *decrease by variable size* jika jumlah simpul anak tiap simpul tidak tetap (pohon tidak penuh) dan pohon tidak seimbang sebagaimana implementasi dalam bahasa Java selanjutnya.

Dengan rumus perhitungan jumlah simpul total dari simpul daun *l* pada pohon *m*-ary penuh, jumlah simpul totalnya adalah:

$$\begin{aligned} \text{nsimpul} &= (ml - 1)/(m\text{-}1) \\ &= (26{\times}n - 1) / (26\text{-}1) \\ &= 1{,}04{\times}n - 0{,}04 \end{aligned}$$

Jika tiap simpul memiliki ukuran b byte, memori yang diperlukan adalah (1,04×n – 0,04) × b byte. Untuk menyimpan 456.976 (26^4) buah kata dengan ukuran tiap simpulnya adalah 210 byte (26 ×8 byte pointer 64-bit + 1 byte char + 1 byte Boolean), diperlukan 99.803.550 byte atau sekitar 99 MB memori.

B. Implementasi Program

Algoritma diimplementasikan dalam bahasa Java. Struktur data simpul (class Node) adalah sebagai berikut. Untuk menghemat memori, tiap simpul tidak menyimpan semua 26 simpul anak, tetapi ditambahkan secara dinamis sesuai kebutuhan. Karena tidak menggunakan array yang statis, simpul anak tidak dapat diakses langsung dengan indeks sesuai urutan alfabet. Maka untuk mempercepat pengaksesan, simpul anak disimpan dalam suatu Map dalam implementasi HashMap dengan *key*-nya adalah karakter yang disimpan oleh anak tersebut.

```
/* file: Node.java */
public class Node {
    private Character info;
    private Boolean isValidWord;
    private Map<Character, Node> children;

    ... method-method kelas ...
}
```

Gambar 6. Kelas Node dalam bahasa Jawa
Sumber: dokumen penulis

Implementasi algoritma pencocokkannya dalam bahasa Java adalah sebagai berikut, *method* checkWord berada di dalam kelas Node. Algoritma diimplementasikan dengan iterasi, bukan rekursi sebagaimana yang terdapat pada *pseudocode*.

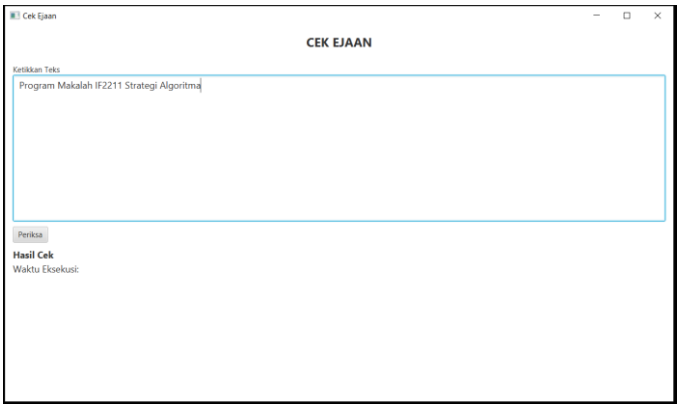
```
/* file: Node.java, class Node */
public Boolean checkWord(String word){
  word = word.trim();
  if(word.isEmpty()) return true;

  boolean isEnded = false;
  boolean isWordExists = false;
  Node current = this;

  while(!isEnded){
    if(current == null){
      isEnded = true;
      isWordExists = false;
    } else if(word.length() == 0){
      isEnded = true;
      isWordExists= current.isValidWord;
     } else {
      current =
     current.children.get(word.charAt(0));
      word = word.substring(1);
     }
  }
  return isWordExists;
}
```

Gambar 7. Implementasi pengecekan ejaan dalam bahasa Jawa
Sumber: dokumen penulis

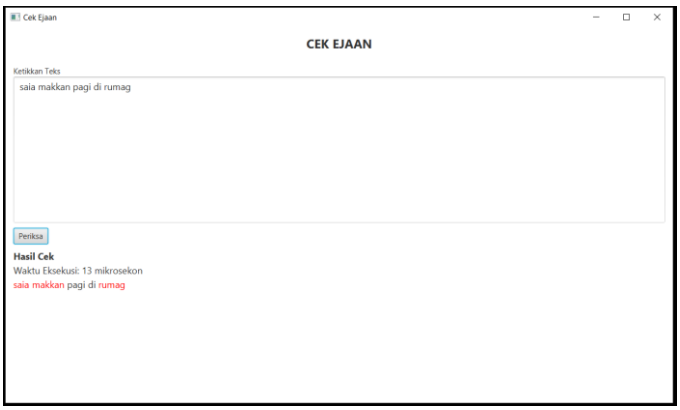
Algoritma kemudian diimplementasikan dalam sebuah program GUI dengan JavaFX. Program berisi area teks yang dapat diketik oleh pengguna, tombol untuk memulai pengecekan, waktu eksekusi pengecekan, dan area teks hasil pengecekan. Teks yang diketikkan di dalam area teks akan di-*split* dengan pemisah spasi dan kemudian tiap kata hasil dari pemecahan akan diperiksa ejaannya. Program masih belum bisa menerima teks yang mengandung tanda baca.



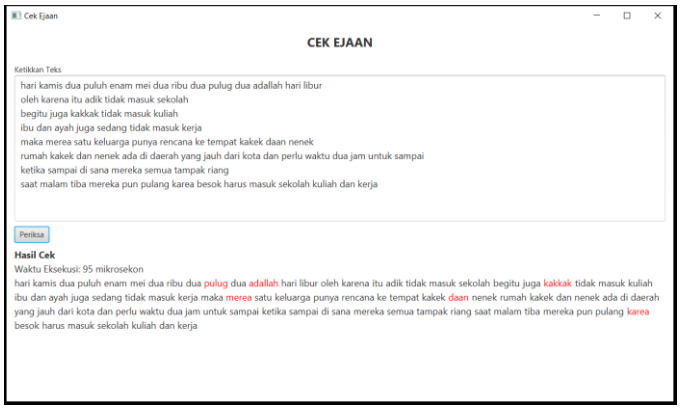
Gambar 8. Program editor teks dengan pemeriksa ejaan
Sumber: dokumen penulis

C. *Pengujian*

Pengujian dilakukan menggunakan daftar kata dasar (tanpa imbuhan) dalam bahasa Indonesia yang terdiri atas 28.526 entri. Data didapatkan dari https://github.com/andrisetiawan/lexicon. Pada area teks hasil cek, kata-kata yang valid akan berwarna hitam, sedangkan yang tidak valid akan berwarna merah. Waktu eksekusi yang tercatat adalah waktu murni untuk mencari suatu kata dalam daftar kamus, tidak termasuk waktu untuk memuat daftar kamus dari *file* dan waktu untuk menampilkan hasil pemeriksaan ke GUI.



Gambar 9. (*atas)* Pengujian tanpa kata salah eja
(bawah) Pengujian beberapa kata salah eja
Sumber: dokumen penulis



Gambar 10. Pengujian teks panjang dengan beberapa kata yang dieja salah
Sumber: dokumen penulis

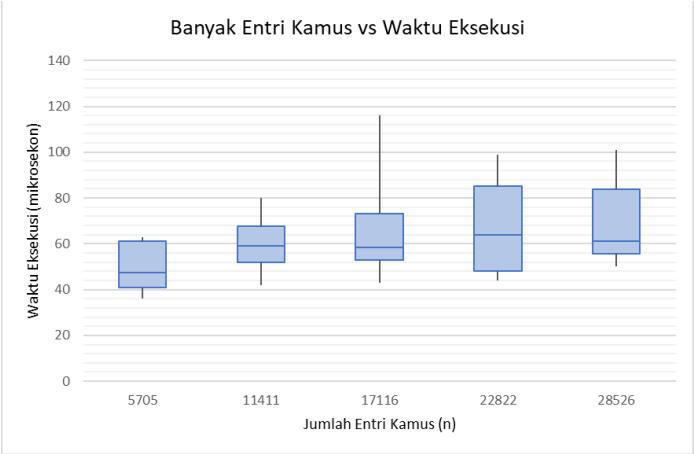
Pada gambar 9 (atas), semua kata dalam kalimat "saya makan pagi di rumah" sudah memenuhi ejaan yang benar. Pada gambar 9 (bawah), kata "saia", "makkan", dan "rumag" tidak sesuai kaidah ejaan dalam kamus sehingga ditandai merah pada hasil pemeriksaan. Pada gambar 10, program dapat memeriksa teks yang cukup panjang dan mendeteksi beberapa kata salah eja, yaitu "pulug", "adallah", "kakkak", merea, "daan", dan "karea".

Pengujian selanjutnya adalah pengujian performa algoritma. Pengujian dilakukan dengan memeriksa sekelompok kata terhadap kamus masukan tertentu. Data kamus masih menggunakan data yang sama dengan bagian pengujian sebelumnya, tetapi urutan baris diacak (*shuffle*) sehingga tidak terjadi bias. *s* buah kata dengan banyak huruf total *l* dipilih dari suatu kumpulan kalimat, diperiksa terhadap kamus berjumlah entri *n*, kemudian waktu pemeriksaan dicatat. Pengetesan dilakukan terhadap sekumpulan kata karena pengecekan terhadap kata individual memerlukan waktu yang hampir tidak dapat dibedakan satu sama lain karena keterbatasan *timer* komputer. Hasilnya terdapat pada tabel dan grafik berikut.

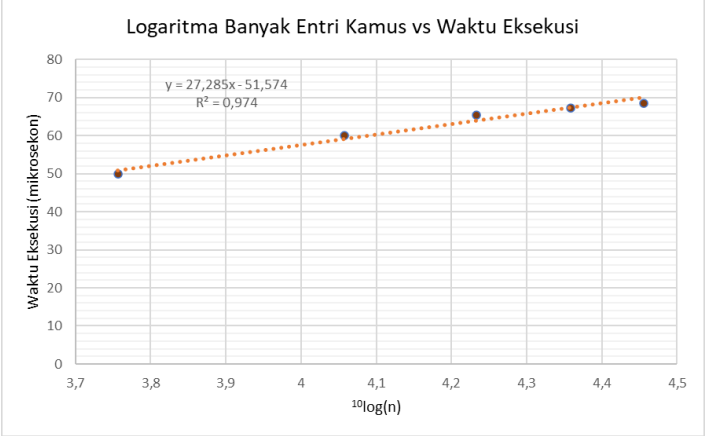
Tabel 1. Tabel hasil 10 kali pengujian untuk pengecekan suatu kalimat dengan jumlah kata = 86 dan total huruf non-*whitespace* = 397 terhadap variasi banyak entri kamus (n)

Sumber: dokumen penulis

Jumlah Entri(n)	5705	11411	17116	22822	28526
AVERAGE	49,9	60,1	65,4	67,3	68,6
STD (S)	10,81	13,48	22,52	21,53	18,26
MIN	36	42	43	44	50
Q1	41	52	52,75	48	55,75
Q2	47,5	59	58,5	64	61
Q3	61	67,5	73,25	85,25	83,75
MAX	63	80	116	99	101



Gambar 11. *Boxplot* yang bersesuaian dengan tabel 1
Sumber: dokumen penulis



Gambar 12. *Plot logaritma* jumlah entri kamus terhadap waktu eksekusi yang bersesuaian dengan tabel 1
Sumber: dokumen penulis

Hasil pengujian dapat dipengaruhi oleh berbagai faktor, terutama spesifikasi komputer yang digunakan. Pengujian juga dapat berbeda-beda pada kondisi uji yang sama karena sumber daya komputer sebenarnya berbeda karena berbagai proses yang berjalan di *background*. Pengujian di atas dilakukan dengan mengusahakan kondisi komputer yang sama untuk tiap kasus pengujian.

Berdasarkan perhitungan kompleksitas, waktu eksekusi akan sebanding dengan penjumlahan m dan log(n). Dilakukan perhitungan regresi linear dengan sumbu y adalah waktu eksekusi dan sumbu x adalah ^{10}log(n). Basis logaritma tidak penting untung mengetahui tren karena hanya akan memberikan perbedaan koefisien dari x pada regresi. Regresi menghasilkan suatu tren yang mendekati garis lurus dengan nilai R^2 sama dengan 0,974. Ini merupakan hasil yang mendekati 1 sehingga dapat disimpulkan kompleksitas algoritma sesuai dengan perhitungan teoritis.

IV. PENUTUP

A. *Kesimpulan*

Pendekatan *decrease-and-conquer* dapat menghasilkan algoritma dengan kompleksitas waktu logaritmik terhadap banyak entri kamus masukan untuk mengecek kesalahan ejaan dalam penulisan. Algoritma dapat memeriksa kata dalam waktu singkat bahkan dalam kamus yang memuat puluhan ribu entri. Struktur data pohon mampu dimanfaatkan untuk menyelesaikan permasalahan pengecekan ejaan dengan efisien.

B. *Saran*

Peningkatan yang dapat dilakukan adalah dengan pengecekan awalan (me-, di-, ter-, dsb.) dan akhiran (-i, -kan, dsb.). Sementara ini, algoritma dan program hanya memeriksa kata sesuai yang tertera di kamus. Bisa saja suatu kata memiliki imbuhan awalan atau akhiran, misalkan kata makan ditambahkan awalan me- sehingga menjadi memakan. Jika kata memakan tidak terdapat di kamus masukan, algoritma akan menandainya sebagai salah ejaan padahal tidak demikian sehingga peningkatan seperti ini dapat ditambahkan ke algoritma. Program juga hanya dapat menerima ketikan kata tanpa tanda baca dan masih *case-sensitive* (makan dan MAKAN berbeda) sehingga dapat ditingkatkan ke depannya.

LINK VIDEO YOUTUBE

https://youtu.be/4zRPqo70ZgA

REPOSITORY LINK AT GITHUB

https://github.com/bryanahusna/Pengecekan-Ejaan

UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan Yang Maha Esa sehingga makalah ini dapat diselesaikan dengan tepat waktu. Terima kasih penulis sampaikan kepada dosen mata kuliah IF2211 Strategi Algoritma yang telah membagikan sebagian dari ilmunya kepada penulis. Tidak lupa penulis berterima kasih kepada orang tua penulis dan teman-teman yang telah mendukung penulisan makalah ini. Akhir kata, semoga makalah ini dapat memberi manfaat kepada pembaca.

REFERENSI

[1] Tim Penyusun Kamus Pusat Bahasa. 2008. *Kamus Bahasa Indonesia*. Jakarta: Pusat Bahasa.

[2] Levitin, Anany. 2012. *Introduction to The Design & Analysis of Algorithms, 3rd Edition.* Harlow: Pearson Education Limited.

[3] Munir, Rinaldi. "Algoritma Decrease and Conquer (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian1.pdf. Diakses 22 Mei 2022.

[4] Rosen, Kenneth H., *Discrete Mathematics and Its Applications, eight edition*. New York: McGraw-Hill, 2019

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 23 Mei 2022

Bryan Amirul Husna
13520146