Analisis Penerapan Algoritma String Matching dalam Situs Web Monkeytype

Afnan Edsa Ramadhan - 13521011
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): (13521011@std.stei.itb.ac.id)

***Abstrak*—Situs web Monkeytype adalah sebuah platform latihan ketikan daring yang populer digunakan untuk meningkatkan keterampilan mengetik. Algoritma *string matching* menjadi komponen kunci dalam menyediakan latihan ketikan yang efektif pada situs Monkeytype. *String matching*, atau pencocokan string, adalah algoritma yang digunakan untuk mencari *pattern* yang terdapat dalam suatu teks. *String matching* akan dilakukan setiap pengguna mengetikkan sebuah karakter terhadap kata yang ada pada layar. Makalah ini akan menganalisis penggunaan algoritma *string matching* dalam situs Monkeytype dan penerapannya secara spesifik di platform tersebut.**

Kata kunci—String matching; Monkeytype; Knuth-Morris-Pratt (KMP); Boyer-Moore(BM); brute force; typing-test

I. PENDAHULUAN

Dalam era perkembangan teknologi yang semakin cepat dan maju ini, keterampilan mengetik menjadi semakin penting dalam kehidupan kita sehari-hari. Mengetik dengan cepat dan akurat akan berdampak pada efisiensi komunikasi kita saat berinteraksi di dunia maya, mengetik dengan cepat dan akurat juga akan berdampak pada meningkatnya produktivitas kita secara keseluruhan, terutama bagi mereka yang kesehariannya berkutat dengan laptop atau PC.

Untuk membantu orang-orang yang ingin mengetik dengan cepat tersedia bermacam-macam website yang dirancang khusus untuk meningkatkan keterampilan mengetik para penggunanya. Monkeytype merupakan salah satu platform daring untuk meningkatkan kemampuan mengetik pengguna. Platform ini menawarkan latihan mengetik dengan tingkat kesulitan yang dapat disesuaikan oleh pengguna, sehingga memungkinkan pengguna untuk mengembangkan keterampilan mereka sesuai dengan kemampuan dan tingkat kenyamanan pengguna. Melalui latihan yang konsisten, pengguna dapat menambah kecepatan, meningkatkan akurasi, serta mengurangi jumlah kesalahan yang dilakukan saat mengetik.

Dalam pengembangan website Monkeytype penggunaan algoritma *string matching* memegang peran penting dalam memberikan pengalaman yang baik kepada pengguna. Penggunaan algoritma *string matching* digunakan pada fitur utama website ini yaitu tes mengetik.

Gambar 1.1 Tampilan antarmuka website Monkeytype

Pada makalah ini penulis akan berfokus mengenai analisis mengenai algoritma *string matching* seperti apa yang digunakan pada situs web Monkeytype dan bagaimana penerapannya.

II. LANDASAN TEORI

A. *Algortima String Matching*

Algoritma *string matching* adalah sebuah algoritma untuk mencari kecocokan suatu sub-pattern, pola, atau kata tertentu dalam sebuah teks atau kalimat yang dimiliki. Tujuan utama dari algoritma ini adalah untuk menemukan keberadaan sebuah sub-pattern atau pola dalam sebuah teks yang lebih besar. Definisi dari *string matching* pada umumnya sebagai berikut:

1. T : teks, yaitu string denga panjang n karakter
2. P : pattern, yaitu string dengan panjang m karakter (m<n) yang akan diperiksa di dalam teks.

Berikut adalah contoh dari *string matcing* dengan teks T dan *pattern* P

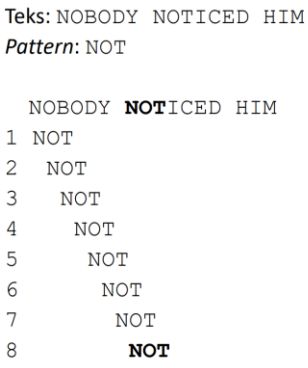
T : Saya berkuliah di teknik **informatika** ITB

P : **informatika**

Algoritma *string matching* ini dapat diaplikasikan ke berbagai bidang sebagai contoh, fitur pencarian pada teks editor atau pdf *reader*, mesin pencarian kata seperti google, analisis citra seperti pembacaan sidik jari, bioinformatik seperti pencocokan DNA, dan yang akan dibahas pada makalah ini yaitu situs web Monkeytype.

B. *Algoritma Brute Force*

Algoritma *brute force* dapat digunakan untuk memecahkan persoalan *string matching*. Penerapan algoritma *brute force* pada persoalan *string matching* dilakukan dengan cara memeriksa kecocokan karakter pada *pattern* dengan teks yang tersedia satu persatu secara urut dari kiri ke kanan. Apabila karakter yang sedang diperiksa cocok, maka akan pemeriksaan bergeser sebanyak satu karakter terhadap masing-masing *pattern* dan teks. Jika karakter tidak sama, maka indeks teks yang diperiksa bergeser satu karakter, dan pemeriksaan diulang kembali.



Gambar 2.1 Contoh impelementasi algoritma brute force pada string matching

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Meskipun algoritma *brute force* ini cukup sederhana, namun algoritma ini memiliki kompleksitas waktu yang relatif tinggi dengan rata-rata kasus sebesar O(n+m). sedangkan untuk kasus terbaik memiliki kompleksitas O(n), ini terjadi apabila karakter pertama *pattern* P tidak pernah sama dengan karakter teks T yang dicocokkan.

C. *Algoritma Knuth-Moris-Pratt (KMP)*

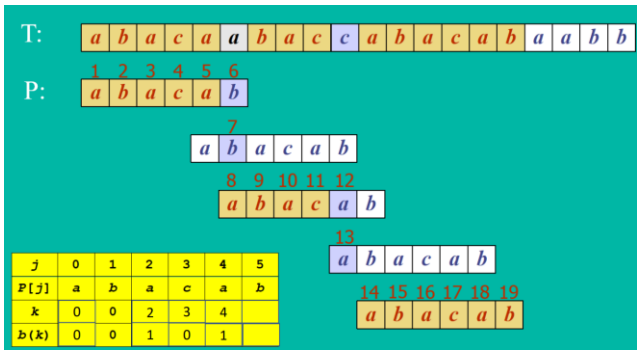
Algortima Knuth-Morris-Pratt adalah sebuah algoritma *string matching* yang diciptakan Donald E. Knuth, Vaughan Pratt, dan James H. Morris secara terpisah pada tahun 1970. Mereka semua lalu mempublikasikan algoritma ini pada tahun 1977. Nama algoritma ini terinspirasi dari ketiga penemunya.

Algoritma KMP ini memiliki cara kerja yang mirip dengan algoritma *string matching* dengan metode *brute force* dengan pencarian *pattern* dari kiri ke kanan, namun dengan perbedaan ide utama pada algoritma KMP adalah menghidari pengulangan iterasi dari teks uji, denga cara pergeseran yang lebih efisien.

Untuk menggunakan algoritma KMP dibutuhkan sebuah fungsi pembantu yang disebut sebagai *border function*, sebuah *border function* didefinisikan sebagai ukuran prefix terbesar dari P[0..k] yang juga merupakan suffix dari P[1..k]. Misal pada sebuah *pattern* ”abacab” maka prefiksnya adalah ”a”, ”ab”, ”aba”, ”abaca”, ”abacab” sedangkan suffiksnya adalah ”b”, ”ab”, ”cab”, ”acab”, ”bacab”. Jika akan dicari nilai pinggiran untuk k = 5, pada suffiks dan prefiks ditemukan kesamaan pada ”ba”. Maka fungsi b(5) akan mengembalikan nilai panjang ”ba”, yaitu 2.

Cara kerja dari algoritma KMP saat mencocokkan *pattern* adalah sebagai berikut.

1. *Pattern* disejajarkan pada awal teks.
2. Telusuri *pattern* dari kiri sampai kanan dengan membandingkan karakter pada pattern dengan karakter dengan urutan bersesuaian pada teks.
3. Setelah melakukan perbandingan, jika semua karakter pada *pattern* cocok dengan karakter pada teks maka pencarian berhasil.
4. Jika saat melakukan perbandingan ditemukan karakter yang tidak cocok atau *mismatch*, maka hitung *border function* untuk menentukan berapa pergeseran yang harus dilakukan. Ulangi kembali ke langkah 2.



Gambar 2.2 Contoh Implementasi algoritma KMP

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Kompleksitas algortima KMP adalah O(m+n) dengan rincian, menghitung *border function* dengan kompleksitas O(m) dan proses pencarian *pattern* di dalam teks dengan kompleksitas O(n) sehingga menghasilkan kompleksitas akhir O(m+n). Algoritma KMP relatif lebih cepat dibandingkan dengan algoritma *brute force*. Namun, jika *border function* mengembalikan nilai yang kecil atau bahkan 0, maka kompleksitasnya akan sama dengan algoritma *brute force*.

D. *Algoritma Boyer-Moore*

Algoritma Boyer-Moore dikembangkan oleh Robert S. Boyer dan J Strother Moore pada tahun 1977. Tidak seperti algoritma brute force dan KMP, algoritma Boyer-Moore melakukan perbandingan string yang bergerak dari kanan ke kiri.

Algoritma Boyer-Moore merupakan algoritma pattern matching yang memanfaatkan dua teknik yang unik, yaitu:

1. *The looking-glass technique*

   Teknik ini merupakan teknik yang mencari *pattern* P pada teks T dengan bergerak mundur dari kanan ke kiri, dimulai dari indeks terakhir.

2. *The character-jump technique*

   Teknik ini berarti indeks akan lompat dengan cara yang berbeda, Apabila *mismatch* terjadi pada T[i] == x, dan

karakter pada *pattern* P[j] tidak sama dengan T[i], maka dilakukan penggeseran karakter.

Terdapat tiga kemungkinan kasus dalam penggeseran karakter pada teknik *character-jump*, yaitu

1. Kasus pertama

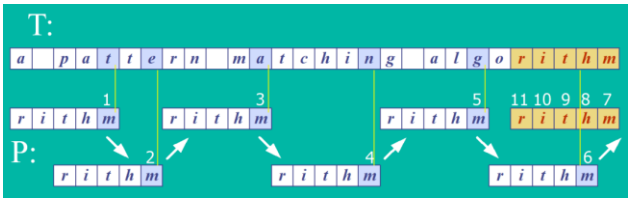
   Jika *pattern* P memiliki karakter x di sebelah kiri dari indeks j, maka geser P ke kanan sedemikian rupa sehingga P sejajar dengan indeks i, yaitu indeks dimana x terdekat berada.

2. Kasus kedua

   Jika *pattern* P memiliki karakter x, tetapi bukan di sebelah kiri dari indeks j, maka geser P ke kanan sebanyak 1 karakter menuju T[i+1].

3. Kasus ketiga

   Jika karakter x bukan merupakan substring dari *pattern* P, maka geser pattern P sedemikian rupa sehingga pattern P[0] sejajar dengan T[i+1]



Gambar 2.3 Contoh Implementasi algoritma Boyer-Moore

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Untuk mempermudah pergeseran dari *character-jump*, maka algoritma BM akan menggunakan fungsi *last occurrence*. Fungsi *last occurrence* L(x) didefinisikan sebagai indeks terbesar dari P dimana P[i] == x. Apabila indeks tidak ditemukan, maka nilai L(x) adalah -1. Contoh dari penggunaan fungsi last occurrence pada pola “abacab” dan ruang lingkup karakter {a, b, c, d}:

x	***a***	***b***	***c***	***d***
L(x)	4	5	3	-1

Gambar 2.4 Contoh penggunaan fungsi last occurance

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Algoritma Boyer-Moore merupakan algoritma yang cepat apabila ruang lingkup karakter dari pola dan teks besar akan tetapi lambat apabila ruang lingkup karakternya kecil.

E. Algoritma Jarak Levenshtein

Algoritma Jarak Levenshtein, juga dikenal sebagai *Edit Distance*, digunakan untuk mengukur perbedaan atau kesamaan antara dua rangkaian karakter. Algoritma ini dinamai dari matematikawan Soviet, Vladimir Levenshtein, yang mengusulkan metode ini pada tahun 1965. Tujuan utama dari algoritma Jarak Levenshtein adalah menghitung jumlah operasi penyisipan, penghapusan, dan penggantian karakter yang diperlukan untuk mengubah satu string menjadi string lainnya.

Algoritma Jarak Levenshtein menggunakan pendekatan pemrograman dinamis untuk menghitung jarak atau perbedaan antara dua string. Dalam langkah awal, algoritma membuat sebuah matriks yang merepresentasikan perbandingan antara karakter-karakter dari kedua string. Setiap elemen matriks menunjukkan jarak Levenshtein antara dua substring yang bersesuaian. Dengan menggunakan aturan rekursif, matriks ini diisi dengan nilai-nilai yang memungkinkan perhitungan jarak Levenshtein yang akurat.

Secara matematis, Levenshtein Distance antara dua string a dan b dengan panjang |a| dan |b| dapat dijabarkan sebagai berikut

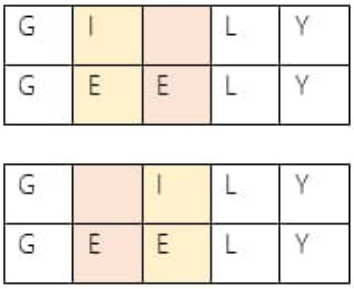
$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Gambar 2.5 Persamaan matematis algoritma jarak levensthein

Sumber : https://www.cuelogic.com/blog/the-levenshtein-algorithm

Di mana 1(ai≠bi) adalah fungsi indikator yang bernilai 0 saat ai≠bi dan bernilai 1 jika sebaliknya, dan leva, b(i,j) adalah jarak antara i karakter pertama dari a dan j karakter pertama dari b.

Sebagai contoh, Jarak Levenshtein antara ”GILY” dan ”GEELY” adalah dua, karena dibutuhkan dua perubahan untuk mengubah salah satu kata menjadi kata lain, dan tidak ada cara lain untuk melakukannya dengan kurang dari dua perubahan.



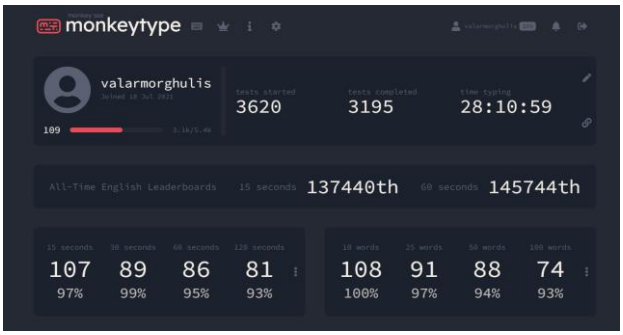
Gambar 2.6 Contoh Implemetasi algoritma jarak Levenshtein

Sumber : https://www.cuelogic.com/blog/the-levenshtein-algorithm

F. Situs Web Monkeytype

Situs Web Monkeytype adalah sebuah platform latihan ketikan daring yang populer digunakan untuk meningkatkan keterampilan mengetik. Dirancang dengan antarmuka pengguna yang intuitif, Monkeytype memberikan pengalaman belajar yang menyenangkan dan efektif. Dalam latihan ketikan, pengguna diberikan teks yang harus diketikkan dengan kecepatan dan ketepatan yang maksimal. Monkeytype menawarkan variasi yang memungkinkan pengguna untuk menyesuaikan sesi latihan sesuai kebutuhan mereka. Selain itu, Monkeytype juga menyediakan statistik dan pemantauan

kemajuan pengguna, memungkinkan mereka melihat perkembangan keterampilan mengetik mereka seiring waktu.



Gambar 2.7 Tampilan Halaman Statistik Monkeytype

Fungsi utama Monkeytype adalah membantu pengguna mengembangkan kecepatan dan ketepatan mengetik mereka. Dengan latihan reguler menggunakan Monkeytype, pengguna dapat memperbaiki posisi jari, mengoptimalkan koordinasi tangan dan mata, serta mengasah kebiasaan mengetik yang benar. Fitur-fitur adaptif dan variasi teks dalam latihan memastikan variasi dan tantangan yang berkelanjutan untuk meningkatkan keahlian mengetik. Monkeytype juga memberikan umpan balik secara *real-time* dan statistik kemajuan pengguna, memungkinkan mereka melacak perkembangan mereka dari waktu ke waktu. Dengan demikian, Monkeytype menjadi alat yang efektif dan terjangkau bagi siapa pun yang ingin meningkatkan kecepatan dan ketepatan mengetik mereka dalam lingkungan belajar yang interaktif dan menyenangkan.

Selain fungsi utama sebagai alat latihan ketikan, Monkeytype juga menjadi tempat komunitas yang aktif bagi para pengguna. Pengguna Monkeytype dapat berinteraksi melalui forum, saling memberikan dukungan, berbagi tips, dan memperoleh inspirasi dari sesama anggota komunitas. Hal ini dapat menciptakan lingkungan belajar yang akan memotivasi pengguna untuk terus meningkatkan keterampilan mengetik mereka.

Dalam website Monkeytype, algoritma *string matching* menjadi komponen kunci dalam menyediakan latihan ketikan yang efektif. Algoritma ini digunakan untuk membandingkan input pengguna dengan teks yang harus diketikkan secara efisien dan akurat. Hal ini memungkinkan Monkeytype untuk memberikan umpan balik yang tepat waktu dan akurat kepada pengguna, serta melacak kecepatan dan ketepatan mereka secara efisien. Dengan menggunakan algoritma *string matching* ini, Monkeytype memastikan latihan ketikan yang menguji kemampuan pengguna secara optimal, membantu mereka memperbaiki keterampilan mengetik dengan lebih efektif.

III. PEMBAHASAN

Pada bagian ini akan dijelaskan bagaimana penggunaan algoritma *string matching* untuk mencocokkan input dari pengguna dengan teks yang harus diketikkan. Penjelasan tentang proses situs web Monkeytype melakukan pengetesan mengetik dan bagaimana algoritma string matching digunakan dalam situs web Monkeytype adalah sebagai berikut:

A. Penerimaan Input Pengguna

Monkeytype menerima input pengguna berupa teks yang harus diketikkan. Pengguna diberikan kata-kata atau kalimat yang harus diketikkan dengan cepat dan akurat pada layar. Teks tersebut mungkin terdiri dari satu kata, beberapa kata, atau kalimat lengkap. Monkeytype akan secara acak membangkitkan kata acak sesuai bahasa dan sesuai dengan mode yang sudah dipilih oleh pengguna sebelumnya, lalu program akan menampilkan kata-kata tersebut ke layar. Pengguna diharapkan mengetikkan setiap karakter dengan benar, mulai dari huruf, angka, hingga tanda baca sesuai dengan yang ada pada layar. Setiap karakter yang diinput oleh pengguna akan langsung diproses pada poin kedua.

B. Pencocokkan Karakter

Bagian ini merupakan salah satu fitur utama yang dimiliki oleh website Monkeytype sekaligus menjadi bahasan utama pada makalah ini. Pada Monkeytype, pencocokkan karakter dilakukan dengan menggunakan algoritma *string matching*. Algoritma ini membandingkan setiap karakter input pengguna dengan karakter-karakter yang seharusnya ada pada posisi yang sama dalam teks yang harus diketikkan.

Dikarenakan tidak terdapat dokumentasi yang secara eksplisit menyebutkan algoritma *string matching* mana yang digunakan dalam situs web Monkeytype. Penulis malakukan analisis mandiri berdasarkan pengalaman pribadi. Berdasarkan hasil analisis tersebut, penulis berasumsi bahwa algoritma string matching yang digunakan bisa jadi adalah algoritma KMP atau algoritma *brute force*. Hal ini didasarkan pada pengamatan bahwa perbandingan karakter-karakter pada MonkeyType dilakukan secara *real-time*, tanpa menunggu seluruh kata selesai diketik. Oleh karena itu, algoritma yang sesuai akan melakukan pencocokan karakter dari kiri ke kanan, bukan dari akhir karakter kata atau dari kanan ke kiri seperti yang dilakukan oleh algoritma Boyer-Moore. Hal lain yang mendasari asumsi ini adalah algoritma Boyer-Moore yang kurang efektif untuk melakukan perbandingan jika ruang lingkupnya kecil dalam hal ini adalah perkata.

Berdasarkan analisis sebelumnya setelah dikerucutkan lagi, penulis berasumsi bahwa Monkeytype menggunakan algoritma *brute force* untuk melakukan proses *string matching*. Asumsi ini didasarkan pada fakta bahwa dalam situs Monkeytype, tidak ada kebutuhan untuk menggeser indeks pencarian karena proses pencocokkan harus dimulai dari karakter pertama dalam teks yang harus diketikkan. Dengan demikian, algoritma *brute force* yang sederhana dapat digunakan tanpa perlu menghitung *border function* seperti yang dibutuhkan oleh algoritma KMP. Penggunaan algoritma *brute force* memungkinkan pencocokkan karakter dilakukan secara efisien tanpa memakan bannyak waktu. Kurang lebih algoritma yang digunakan pada website Monkeytype akan seperti gambar 3.1.

```
function bruteForce(text, pattern : string)
-> boolean
  Kamus Lokal :
  n,m,i : integer

  Algoritma :
  n <- text.length
  m <- pattern.length
  if(m>n) then
    -> false
  end if
  for(i<-0; i<m; i++) do
    if(text[i] ≠ pattern[i]) then
      -> false
    end if
  end for
  -> true
```

Gambar 3.1 Pseudocode function brute force yang digunakan dalam website Monkeytype

Saat pengguna mengetik, setiap karakter yang diinput langsung dibandingkan dengan karakter yang seharusnya muncul pada posisi yang tepat dalam teks. Algoritma *string matching* melakukan pencocokkan karakter secara berurutan, memeriksa kesamaan antara karakter input pengguna dan karakter yang seharusnya ada.

Gambar 3.2 Tampilan jika seluruh kata diketikan dengan benar

Jika karakter yang diinput oleh pengguna sesuai dengan karakter yang seharusnya ada pada posisi tersebut, Monkeytype melanjutkan ke karakter berikutnya dan pengguna dapat melanjutkan mengetik. Namun, jika terdapat kesalahan dalam pencocokkan karakter, Monkeytype memberikan umpan balik visual yang menunjukkan kesalahan tersebut. Ini bisa berupa perubahan warna atau penanda pada karakter yang salah.

Gambar 3.3 Tampilan jika pengguna mengetikan kata yang berlebih

Gambar 3.4 Tampilan jika pengguna mengetikan kata yang salah

Proses penggunaan string matching dalam website Monkeytype kurang lebih seperti pada tabel dibawah ini, setiap baris adalah setiap pengguna mengetikkan suatu karakter (ilustrasi berdasarkan gambar 3.3)

Teks	***Pattern* / Kata yang diketikan oleh pengguna**	**Status**
wajar	w	Diterima
wajar	wa	Diterima
wajar	waj	Diterima
wajar	waja	Diterima
wajar	wajar	Diterima
wajar	wajarr	Ditolak

Tabel 3.1 Ilustrasi proses string matching pada website Monkeytype

C. Pelacakan Kecepatan dan Keakuratan Penguna

Monkeytype secara teliti memantau dan mencatat kecepatan serta keakuratan pengguna saat mengetik. Platform ini memberikan *feedback* kepada pengguna berupa informasi yang berharga mengenai statistik dan perkembangan mereka dalam keterampilan mengetik.

Pertama, Monkeytype melacak kecepatan pengguna dengan menghitung jumlah karakter yang diketik dalam waktu tertentu, biasanya dalam satuan menit. Kecepatan ini dinyatakan dalam jumlah kata per menit (KPM) atau *word per minute* (WPM). Pengguna dapat melihat seberapa cepat mereka mengetik dan membandingkannya dengan target kecepatan tertentu untuk meningkatkan keterampilan mengetik mereka.

Selain itu, Monkeytype juga melacak keakuratan pengguna. Ini dihitung dengan membandingkan jumlah karakter yang diketik dengan benar dengan jumlah total karakter yang harus diketik. Keakuratan ini dinyatakan dalam persentase, di mana pengguna dapat melihat berapa persen karakter yang diketik dengan benar. Dengan mengetahui tingkat keakuratan mereka, pengguna dapat mengidentifikasi area yang perlu diperbaiki dan meningkatkan ketepatan mengetik mereka secara keseluruhan.

Dengan melacak kecepatan dan keakuratan pengguna, Monkeytype memberikan pengguna gambaran yang lengkap tentang kemampuan mengetik pengguna. Ini membantu pengguna mengenali kekuatan dan kelemahan mereka, serta

memberi motivasi untuk terus berlatih dan meningkatkan keterampilan mengetik.

Gambar 3.5 Tampilan pengguna yang telah menyelesaikan latihan

IV. KESIMPULAN

Monkeytype adalah situs web yang digunakan untuk tes mengetik dengan cepat dan akurat. Algoritma string matching digunakan dalam Monkeytype untuk mencocokkan input pengguna dengan teks yang harus diketikkan. Proses ini dilakukan secara real-time, di mana setiap karakter input pengguna dibandingkan dengan karakter yang seharusnya ada pada posisi yang sama dalam teks yang harus diketikkan. Meskipun tidak ada dokumentasi yang menyebutkan algoritma yang digunakan secara eksplisit, hasil analisis menunjukkan bahwa Monkeytype mungkin menggunakan algoritma *brute force* untuk pencocokkan karakter. Kecepatan dan keakuratan pengguna secara teliti dipantau oleh Monkeytype. Informasi tentang kecepatan dan keakuratan ini memberikan pengguna gambaran yang lengkap tentang kemampuan mengetik mereka, membantu mereka mengenali kekuatan dan kelemahan, serta memberi motivasi untuk terus meningkatkan keterampilan mengetik secara keseluruhan.

Secara keseluruhan, penerapan algoritma *string matching* dalam Monkeytype memberikan manfaat yang nyata dalam meningkatkan akurasi, efisiensi, dan pengalaman pengguna dalam latihan ketikan. Dengan penggunaan algoritma ini, Monkeytype dapat menjadi alat yang efektif dalam membantu pengguna mengembangkan keterampilan mengetik mereka, meningkatkan kecepatan, ketepatan, dan produktivitas mereka secara keseluruhan.

V. UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya makalah ini dapat diselesaikan penulis dengan tepat waktu. Penulis juga ingin berterima kasih kepada dosen pengampu mata kuliah IF2211 Strategi Algoritma K-03 yaitu Bapak Ir. Rila Mandala, M.Eng., Ph.D. atas ilmu yang telah disampaikan dan atas bimbingannya selama satu semester. Tidak lupa juga penulis ingin berterima kasih sebanyak-banyaknya kepada teman-teman IF K-03 yang telah membantu penulis selama satu semester kebelakang saat penulis memiliki kesulitan. Semoga makalah ini dapat memberikan manfaat baik secara langsung maupun tidak langsung bagi para pembaca.

REFERENCES

[1] Munir, Rinaldi. 2021. Pencocokan string (string matching). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses pada 18 Mei 2023.

[2] Khumaidi Ali. 2020. Comparison of Knuth Morris Pratt and Boyer Moore algorithms for a web-based dictionary of computer terms. https://media.neliti.com/media/publications/466118-comparison-of-knuth-morris-pratt-and-boy-3222920a.pdf. Diakses pada 19 Mei 2023

[3] Cuelogic Insights. (2021, June 19). The Levenshtein Algoritm. https://www.cuelogic.com/blog/the-levenshtein-algorithm. Diakses pada 19 Mei 2023.

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, 22 Mei 2023

Afnan Edsa Ramadhan
13521011