Implementasi Algoritma String Matching dan Algoritma Greedy dalam Penentuan Rekomendasi Pencarian Media Sosial Tiktok

Muhamad Salman Hakim Alfarisi - 13521010
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail) : 13521010@std.stei.itb.ac.id

***Abstract*—Makalah ini membahas implementasi Algoritma String Matching dan Algoritma Greedy dalam konteks penentuan rekomendasi pencarian di platform media sosial TikTok. Algoritma String Matching, termasuk Brute Force, Knuth-Morris-Pratt, dan Boyer-Moore, digunakan untuk mencari substring dalam teks untuk menentukan kata kunci yang relevan. Selanjutnya, Algoritma Greedy digunakan untuk memilih kata-kata teratas berdasarkan jumlah kemunculannya. Implementasi ini bertujuan untuk membandingkan akurasi dan kecepatan ketiga algoritma dalam pencarian rekomendasi pada TikTok.**

Keywords—tiktok, string matching, greedy

I. Pendahuluan

Dewasa ini, media sosial telah menjadi bagian yang penting dalam kehidupan sehari-hari kita. Media sosial memungkinkan kita melakukan berbagai aktivitas seperti mengirim pesan, melakukan panggilan biasa, melakukan panggilan video, membagikan aktivitas, dan sebagainya. Banyak sekali platform media sosial yang menyediakan hal-hal tersebut. Salah satu platform media sosial yang sangat populer akhir-akhir ini adalah Tiktok. Tiktok adalah platform berbagi video pendek antar pengguna. Pengguna Tiktok dapat membuat konten video pendek lalu mengunggahnya maupun hanya sekedar menonton, berkomentar, dan menyukai video pengguna lain dalam platform tersebut.

Tiktok menjadi populer karena dapat menyajikan video pendek dengan efek yang beragam, kualitas video yang mumpuni namun tetap ringan diakses, serta dapat menyesuaikan konten video berdasarkan preferensi masing-masing pengguna. Selain itu, hal lain yang menjadi daya tarik utama tiktok adalah relevansi rekomendasi pencarian dalam suatu kolom komentar pada masing-masing konten video. Tiktok menyajikan rekomendasi pencarian yang sesuai bahasan, baik pada video maupun pada kolom komentar. Rekomendasi pencarian pada kolom komentar Tiktok berbentuk kata kunci yang terdiri dari 2 hingga 3 kata.

Dalam menentukan kesesuaian rekomendasi pencarian pada masing-masing video Tiktok diperlukan suatu algoritma yang efektif. Salah satu algoritma yang dapat diimplementasikan adalah algoritma string matching. Algoritma ini dapat digunakan untuk membandingkan dua buah string. String yang dibandingkan dapat berasal dari video dan kolom komentar.

Pada penelitian ini saya ingin melakukan implementasi algoritma string matching, yaitu Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Brute Force, ditambah dengan Greedy dalam menentukan rekomendasi pencarian di Tiktok.

II. Teori Dasar

A. Tiktok

Tiktok adalah platform media sosial yang memungkinkan pengguna untuk membuat, membagikan, dan menonton video pendek. Tiktok merupakan produk unggulan yang diluncurkan pada Mei 2017 oleh ByteDance Ltd., dengan tujuan utama platform adalah video berdurasi pendek. Pada umumnya durasi video pada Tiktok berada di rentang 15 hingga 60 detik. Durasi video yang pendek ini memungkinkan untuk menjadi daya tarik tersendiri bagi pengguna, karena dengan durasi yang cukup pendek tersebut pengguna dapat memperoleh informasi dan hiburan.

Pada Tiktok, terdapat dua jenis pengguna, yaitu konten kreator dan pengguna biasa. Konten kreator adalah pengguna yang aktif membuat video lalu mengunggahnya ke dalam komunitas Tiktok. Konten kreator dapat menggunakan berbagai *tools* dan efek visual yang disediakan Tiktok untuk mendukung kualitas serta daya tarik video. Pada umumnya, konten kreator memiliki jumlah pengikut yang banyak. Sedangkan pengguna biasa adalah pengguna yang pasif mengunggah video ke dalam komunitas. Mereka cenderung hanya menonton video yang diunggah konten kreator dan berinteraksi dengan memberikan *like*, *comment*, dan *share* video yang mereka sukai.

Masing-masing platform media sosial memiliki fitur utama yang diandalkan. Fitur utama pada Tiktok yang digemari pengguna adalah fitur “For You Page” atau biasa disebut FYP. Fitur FYP pada Tiktok memungkinkan pengguna dapat memperoleh video-video yang sudah di personalisasi.

Personalisasi ini didasarkan pada preferensi pengguna yang melakukan *like*, *comment*, dan *share* video tertentu.

Fig. 1. Kolom komentar Tiktok. *(terdapat kata kunci rekomendasi pencarian "harga macbook m1")*

Pada masing-masing video Tiktok terdapat sebuah kolom komentar dimana pengguna dapat berinteraksi dengan konten kreator maupun pengguna lain. Di kolom komentar ini seringkali terdapat sebuah kata kunci 2 sampai 3 kata yang merekomendasikan ke dalam sebuah pencarian. Ini merupakan fitur yang muncul ketika konten video dengan kolom komentar memiliki topik yang sama. Hal ini dapat membantu pengguna dalam menjelajahi lebih lanjut tentang topik yang dibahas pada video serta kolom komentar.

B. *String*

String adalah urutan karakter, yang terdiri dari huruf, angka, tanda baca, dan simbol-simbol lain. Secara umum, string merupakan sebuah array karakter yang menyimpan urutan elemen karakter dengan pengkodean tertentu. String biasanya dikelilingi oleh tanda kutip ganda (“”). Pada string terdapat dua macam istilah antara lain:

1. Prefix (Awalan), yaitu substring dari string S yang dimulai dari karakter pertama hingga karakter ke-i, dimana i adalah indeks antara 0 dan panjang string-1.

2. Suffix (Akhiran), yaitu yaitu substring dari string S yang dimulai dari karakter ke-i hingga karakter terakhir, dimana i adalah indeks antara 0 dan panjang string-1.

C. *Algoritma String Matching*

Algoritma pencocokan string atau Algoritma String Matching adalah suatu algoritma yang digunakan untuk mencari kemunculan suatu pola atau string yang disebut pattern dalam sebuah teks. Pattern ini adalah sebuah pola yang diharapkan muncul dalam teks tersebut, dengan catatan panjang pattern harus lebih pendek daripada atau sama dengan panjang teks yang dicocokkan.

Algoritma String Matching seringkali diterapkan di berbagai bidang. Dalam bidang pengembangan perangkat lunak, algoritma string matching digunakan untuk pencocokan string atau pola tertentu dalam teks atau kode program. Hal ini dapat difungsikan untuk pengeditan teks, analisis teks, dan verifikasi sintaks.

Selain itu, pada bidang pencarian informasi dan mesin pencari, algoritma string matching digunakan untuk mencari dan mencocokkan string kueri dengan dokumen atau halaman web yang relevan. Hal ini memungkinkan pengindeksan dan pencarian yang efisien di dalam koleksi besar teks.

D. *Jenis-jenis Algoritma String Matching*

Terdapat beberapa jenis algoritma string matching yang umum digunakan, di antaranya:

1. Algoritma Brute-Force String Matching
2. Algoritma Knuth-Morris-Pratt (KMP)
3. Algoritma Boyer-Moore (BM)

E. *Algoritma Brute-Force String Matching*

Algoritma Brute-Force String Matching adalah algoritma pencocokan string sederhana dengan mencocokkan setiap kemungkinan pola secara berurutan dengan string input (*pattern*).

Langkah kerja algoritma brute force sebagai berikut.

1. Pada awalnya, *pattern* dan teks disejajarkan pada posisi awal di dalam string.
2. Dengan menelusuri dari kiri ke kanan pada pola, periksa setiap karakter pada pola dengan karakter yang sesuai di dalam teks hingga terjadi salah satu dari dua kondisi berikut:
    - Semua karakter yang dibandingkan cocok atau sama (pencarian berhasil).
    - Ditemukan ketidakcocokan karakter (pencarian belum berhasil).
3. Jika pola belum ditemukan kecocokan dan teks belum mencapai akhir string, maka geser pola satu karakter ke kanan dan ulangi langkah 2 untuk mendapatkan kemunculan pola yang sesuai di dalam teks.

Pencocokan string dengan menggunakan algoritma brute force memiliki kompleksitas yang bergantung pada *case* yang terjadi. Analisis kompleksitas brute force sebagai berikut.

1. Worst Case

   Worst case terjadi ketika setiap kali pencocokan pattern, semua karakter di pattern dibandingkan dengan karakter di text. Sehingga menghasilkan total jumlah perbandingan karakter = m(n – m + 1) = O(mn). Contoh :

   Text : aaaaaaaaaaaaaab
   pattern : aab

2. Best Case

   Best Case terjadi ketika karakter pertama pattern tidak pernah sama dengan karakter text yang dicocokkan. Kompleksitas best case adalah O(n).

   Text : aaaaaaaaaaaaazzz
   pattern : zzz

3. Average Case

   Average Case terjadi ketika pencarian pada teks normal. Kompleksitas average case adalah O(m + n).

   Text: Pada bulan Januari, hujan hampir turun
   Pattern: hujan

F. *Algoritma Knuth-Morris-Pratt (KMP)*

Algoritma KMP adalah algoritma yang melakukan pencarian pola secara berurutan dari kiri ke kanan, mirip dengan algoritma brute force. Namun, yang membedakan KMP adalah cara cerdasnya dalam menggeser pola ketika terjadi ketidakcocokan.

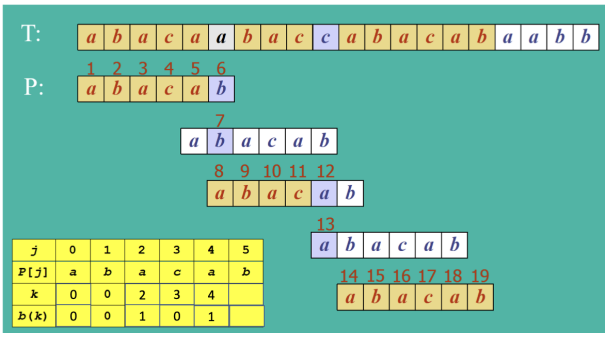


Fig. 2. Contoh penggunaan KMP

Proses kerja algoritma KMP dimulai dengan melakukan pra-pemrosesan pada pola yang akan dicari. Selama tahap ini, algoritma KMP mencari kecocokan awalan pola dengan bagian lain dari pola itu sendiri. Fungsi pinggiran atau fungsi kegagalan (failure function) digunakan untuk menghitung ukuran awalan terpanjang dari pola yang juga merupakan akhiran dari pola tersebut. Fungsi pinggiran b(k) didefinisikan sebagai ukuran awalan terpanjang dari P[0..k] yang juga merupakan akhiran dari P[1..k]. Dengan menggunakan fungsi pinggiran, algoritma KMP dapat menghindari pengujian yang tidak perlu, sehingga meningkatkan efisiensi pencarian.

Kompleksitas waktu algoritma KMP terdiri dari dua bagian utama. Pertama, adalah menghitung fungsi pinggiran, yang memakan waktu O(m) di mana m adalah panjang pola. Proses ini dilakukan sebelum pencarian pola dan hanya perlu dilakukan sekali. Kemudian, pencarian string dilakukan dengan kompleksitas O(n) di mana n adalah panjang teks. Algoritma KMP secara keseluruhan memiliki kompleksitas waktu O(m+n), yang sangat cepat dibandingkan dengan algoritma brute force yang memiliki kompleksitas waktu O(m*n). Oleh karena itu, algoritma KMP lebih efisien dalam mencari pola dalam teks yang panjang dan kompleks.

G. *Algoritma Boyer-Moore (BM)*

Algoritma BM adalah algoritma yang menggunakan pendekatan heuristik untuk mempercepat pencarian pola dalam string. Algoritma ini memanfaatkan informasi dari pola dan karakter terakhir yang tidak cocok dalam string input untuk melompati langkah-langkah pencocokan yang tidak perlu.

Algoritma pencocokan Boyer-Moore didasarkan pada dua teknik utama:

1. Teknik "looking-glass":

Teknik ini mengarahkan pencarian pola P dalam teks T dengan bergerak mundur melalui P, dimulai dari akhir pola. Dengan demikian, pencocokan pola dimulai dari akhir pola dan secara bertahap mundur ke awal.

2. Teknik "character-jump":

Ketika terjadi ketidakcocokan pada T[i] == x, di mana karakter pada posisi i dalam teks tidak sama dengan karakter x dalam pola pada posisi j, algoritma Boyer-Moore menggunakan teknik ini untuk menentukan jumlah geseran yang optimal sebelum mencocokkan kembali pola dengan teks. Algoritma ini mencari kecocokan terakhir dari karakter x dalam pola P sebelum posisi j, dan melompat ke posisi tersebut untuk melanjutkan pencocokan.

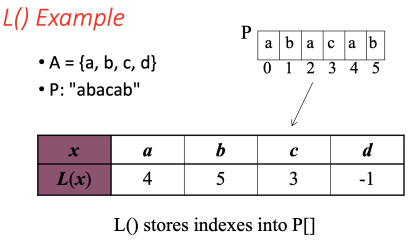


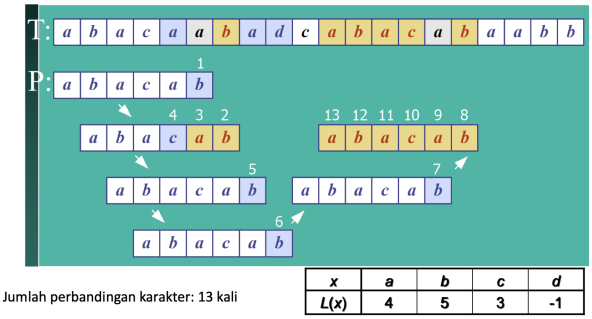
Fig. 3. Contoh penggunaan Last Occurrence

Selain itu, algoritma Boyer-Moore melakukan pra-pemrosesan pada pola P dan alfabet A untuk membangun fungsi "last occurrence" (L). Fungsi L() memetakan semua

huruf dalam alfabet A ke bilangan bulat. L(x) didefinisikan sebagai:

- Indeks terbesar i di mana P[i] == x.
- (-1) jika tidak ada pada pola.

Dalam implementasi algoritma Boyer-Moore, fungsi L() dihitung ketika pola P dibaca. Biasanya, L() disimpan sebagai array yang berisi informasi tentang indeks terakhir dari setiap huruf dalam pola. Dengan adanya tabel ini, pencocokan pola dapat dilakukan secara efisien.



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

Fig. 4. Contoh penggunaan BM

Kompleksitas waktu terburuk algoritma Boyer-Moore adalah O(nm + A), di mana n adalah panjang teks, m adalah panjang pola, dan A adalah ukuran alfabet. Namun, Boyer-Moore menjadi sangat cepat ketika ukuran alfabet (A) cukup besar, sementara menjadi lebih lambat ketika alfabet kecil.

H. Algoritma Greedy

Algoritma greedy (greedy algorithm) adalah metode yang populer dan sederhana untuk memecahkan persoalan optimasi. Persoalan optimasi adalah jenis persoalan yang mencari solusi optimal. Dalam algoritma greedy, tujuan utamanya adalah mencapai solusi optimal dengan memilih langkah terbaik pada setiap langkah, berharap bahwa langkah-langkah tersebut secara akumulatif akan menghasilkan solusi terbaik secara keseluruhan.

Ada dua jenis utama dari persoalan optimasi dalam konteks algoritma greedy, yaitu:

1. Persoalan Maksimasi (Maximization): Dalam persoalan maksimasi, tujuan adalah mencari solusi yang memberikan nilai maksimum dari suatu fungsi objektif.
2. Persoalan Minimasi (Minimization): Dalam persoalan minimasi, tujuan adalah mencari solusi yang memberikan nilai minimum dari suatu fungsi objektif.

III. Implementasi Algoritma

Pada implementasi algoritma string matching dan algoritma greedy dalam penentuan rekomendasi pencarian media sosial Tiktok digunakan ketiga algoritma string matching yaitu brute force, knuth morris pratt, dan boyer-moore. Ketiga algoritma tersebut akan dibandingkan efisiensinya berdasarkan run time. Selain itu digunakan algoritma greedy maksimasi dalam pemilihan kata kunci rekomendasi pencarian.

A. Mekanisme Implementasi Pengujian

Implementasi pengujian dilakukan dengan menguji sebuah sampel yang berisikan kumpulan komentar dan sebuah subtitle video. Pengujian dilakukan dengan sebuah program menggunakan bahasa Python. Sistem kerja program adalah sebagai berikut.

1. Mengolah string subtitle menjadi lowercase dan menghilangkan tanda baca.
2. Mengolah string array komentar menjadi lowercase dan menghilangkan tanda baca.
3. Melakukan pencocokan string menggunakan algoritma brute force, knuth morris pratt, dan boyer-moore.
4. Mendapatkan kata paling sering muncul dengan algoritma greedy maksimasi.

Sampel subtitle yang digunakan :

```
"Hai semuanya, kali ini aku mau unboxing Macbook
Pro M1, Macbook Pro M1 ini adalah laptop terbaru
dari Apple, Macbook Pro M1 ini memiliki performa
yang sangat baik karena menggunakan chip M1,
Cocok banget buat kalian yang suka editing
video, editing foto, dan lain-lain, Macbook Pro
M1 ini juga memiliki desain yang sangat premium,
Macbook Pro M1 ini juga memiliki keyboard yang
sangat nyaman, Macbook Pro M1 ini juga memiliki
layar yang sangat bagus, Macbook Pro M1 ini juga
memiliki speaker yang sangat bagus, Macbook Pro
M1 ini juga memiliki baterai yang sangat awet,
Macbook Pro M1 ini juga memiliki touchpad yang
sangat nyaman, Macbook Pro M1 ini juga memiliki
port yang sangat lengkap, Macbook Pro M1 ini
juga memiliki kamera yang sangat bagus, Macbook
Pro M1 ini juga memiliki microphone yang sangat
bagus, Macbook Pro M1 ini juga memiliki webcam
yang sangat bagus, Macbook Pro M1 ini juga
memiliki fitur yang sangat lengkap, Macbook Pro
M1 ini juga memiliki harga yang sangat mahal,
Macbook Pro M1 ini juga memiliki kualitas yang
sangat baik, Macbook Pro M1 ini juga memiliki
kualitas yang sangat bagus, Gimana menurut
kalian? Apakah Macbook Pro M1 ini worth it?"
```

Sampel komentar yang digunakan :

```
comments = [
    "Macbook Pro M1",
    "Macbook emang keren",
    "Macbook Pro M1 bagus banget",
    "Macbook Pro M1 keren banget",
    "Macbook Pro M1 keren",
    "Macbook Pro M1 bagus",
    "Macbook Pro M1 bagus banget",
    "Pengen banget Macbook Pro M1",
    "Mah beliin aku Macbook Pro M1",
    "Macbook Pro M1 mahal banget",
    "Macbook Pro M1 kurang worth it",
]
```

B. Implementasi Algoritma String Matching

Berikut adalah kode implementasi yang digunakan.

- Algoritma Brute Force

```
def bruteForce(text, pattern):
    lenPattern = len(pattern)
    lenText = len(text)
    if lenPattern > lenText:
        return 0
    elif lenPattern == lenText:
        if text == pattern:
            return 1
        return 0
    i = 0
    count = 0  # Counter for substring
occurrences
    for i in range(0, lenText - lenPattern + 1):
        j = 0
        while j < lenPattern and text[i + j] ==
pattern[j]:
            j += 1
        if j == lenPattern:
            count += 1
    return count
```

- Algoritma KMP

```
def kmpTab(pattern, table):
    position, candidate = 2, 0
    table[0], table[1] = -1, 0
    while position < len(pattern):
        if pattern[position - 1] ==
pattern[candidate]:
            candidate += 1
            table[position] = candidate
            position += 1
        elif candidate > 0:
            candidate = table[candidate]
        else:
            table[position] = 0
            position += 1

def knuthMorrisPratt(text, pattern):
    m, i = 0, 0
    table = [0] * len(pattern)
    kmpTab(pattern, table)
    count = 0  # Counter for substring
occurrences
    while m + i < len(text):
        if pattern[i] == text[m + i]:
            if i == len(pattern) - 1:
                count += 1
                m += i - table[i]
                i = table[i]
            else:
                i += 1
        else:
            if table[i] > -1:
                m += i - table[i]
                i = table[i]
            else:
                i = 0
                m += 1
    return count
```

- Algoritma BM

```
def bmTab(substr):
    table = [-1] * 256
    for i in range(len(substr)):
        table[ord(substr[i])] = i
    return table

def boyerMoore(text, pattern):
    table = bmTab(pattern)
    lenPattern = len(pattern)
    lenText = len(text)
    if lenPattern > lenText:
        return 0
    elif lenPattern == lenText:
        if text == pattern:
            return 1
        return 0
    i = 0
    count = 0  # Counter for substring
occurrences
    while i <= lenText - lenPattern:
        j = lenPattern - 1
        while j >= 0 and text[i + j] ==
pattern[j]:
            j -= 1
        if j < 0:
            count += 1
            i += lenPattern
        else:
            slide = j - table[ord(text[i + j])]
            if slide < 1:
                slide = 1
            i += slide
    return count
```

C. Algoritma Greedy

```
def find_recommended_words(substr_dict_brute):
    recommended_word = []
    for i in range(3):
        max_key = max(substr_dict_brute,
key=substr_dict_brute.get)
        recommended_word.append(max_key)
        substr_dict_brute.pop(max_key)

    recommended_word_sentence = "
".join(recommended_word)
    print("Recommended word: ",
recommended_word_sentence)
```

IV. Hasil Pengujian dan Pembahasan

A. Hasil Pengujian

Dari implementasi algoritma string matching diperoleh hasil sebagai berikut :

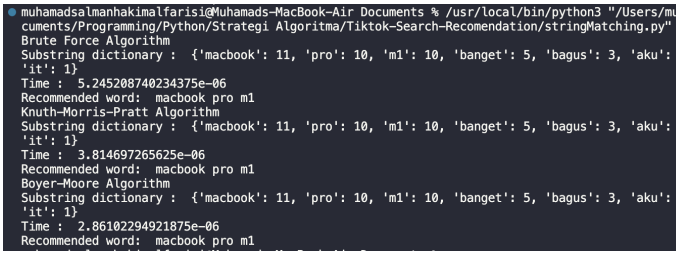


Fig. 5. Hasil Pengujian

Diperoleh data sebagai berikut :

Algoritma	Count Word 1	Count Word 2	Count Word 3	Run Time
Brute Force	11	10	10	5.24e-06
KMP	11	10	10	3.81e-06
BM	11	10	10	2.86e-06

B. Pembahasan

Berdasarkan tabel Hasil Pengujian:

1. Algoritma Brute Force menghitung waktu eksekusi sebesar 5.24e-06 detik dan memberikan rekomendasi kata-kata teratas berdasarkan jumlah kemunculannya. Hasilnya adalah "macbook pro m1" sebagai kata-kata yang paling sering muncul dalam teks.
2. Algoritma KMP menghitung waktu eksekusi sebesar 3.81e-06 detik, yang lebih cepat daripada algoritma Brute Force. Hasil rekomendasi kata-kata teratas dari algoritma KMP adalah "macbook pro m1".
3. Algoritma Boyer-Moore menghitung waktu eksekusi sebesar 2.86e-06 detik, yang lebih cepat dari kedua algoritma sebelumnya. Rekomendasi kata-kata teratas yang diberikan oleh algoritma Boyer-Moore juga sama dengan hasil sebelumnya, yaitu "macbook pro m1".

V. Kesimpulan

Dari hasil implementasi, dapat disimpulkan ketiga algoritma pencarian substring, yaitu Brute Force, Knuth-Morris-Pratt, dan Boyer-Moore, memberikan hasil rekomendasi kata-kata teratas yang sama, yaitu "macbook pro m1". Namun, algoritma Knuth-Morris-Pratt dan Boyer-Moore menunjukkan waktu eksekusi yang lebih cepat daripada algoritma Brute Force. Hal ini menunjukkan bahwa algoritma Knuth-Morris-Pratt dan Boyer-Moore adalah pilihan yang lebih efisien dalam pencocok, terutama ketika menghadapi teks yang lebih besar atau kompleks.

Ucapan Terima Kasih

Puji syukur dan terima kasih kepada Tuhan Yang Maha Esa karena berkat rahmat-Nya, penulis berhasil menyelesaikan makalah berjudul "Implementasi Algoritma String Matching dan Algoritma Greedy dalam Penentuan Rekomendasi Pencarian Media Sosial Tiktok". Penulis juga ingin mengungkapkan rasa terima kasih kepada orang tua, sahabat, dan teman-teman penulis yang selalu memberikan dukungan emosional dan bantuan sehingga makalah ini dapat diselesaikan dengan cepat. Penulis juga ingin mengucapkan terima kasih kepada Ir. Rila Mandala, M.Eng., Ph.D., dosen IF2211 Strategi Algoritma, yang telah memberi penulis pengajaran selama semester 4 ini.

Daftar Pustaka

[1] Munir, Rinaldi. 2021. Algoritma Brute Force (Bagian 1). Algoritma Brute Force (Bagian 1) . Diakses pada 22 Mei 2023.

[2] Munir, Rinaldi. 2021. Algoritma Greedy (Bagian 1). Algoritma Greedy (Bagian 1). Diakses pada 22 Mei 2023.

[3] Munir, Rinaldi. 2021. Pencocokan String (String/Pattern Matching). Pencocokan string (String matching/pattern matching). Diakses pada 22 Mei 2023.

[4] TikTok, "Transparency Report," Tersedia secara daring: https://www.tiktok.com/transparency/id-id/. Diakses pada 22 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

Muhamad Salman Hakim Alfarisi
13521010