Penerapan Algoritma Pencocokan String dalam Sistem Pendeteksian Spam pada *Chat Messenger*

Ghazi Akmal Fauzan - 13521058[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521058@mahasiswa.itb.ac.id

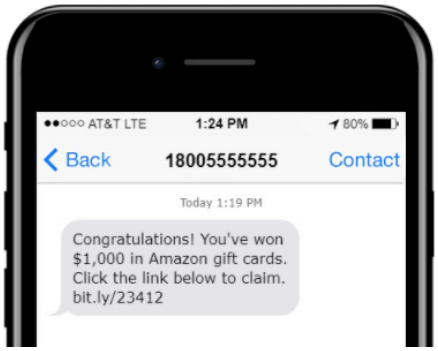
***Abstrak*—*Chat messenger* telah menjadi salah satu alat komunikasi yang paling penting dan populer di era digital saat ini. Dalam beberapa tahun terakhir, perkembangan teknologi telah memungkinkan komunikasi melalui pesan digital menjadi lebih cepat, efisien, dan mudah digunakan. Namun, dampak negatif dari kemajuan ini adalah kemunculan pesan spam yang terus meningkat, mengganggu pengguna dengan pesan-pesan yang tidak diinginkan dan seringkali mengandung konten yang tidak pantas atau penipuan. Pengiriman pesan spam tidak hanya menjadi gangguan bagi pengguna, tetapi juga dapat mengakibatkan kerugian finansial, kehilangan data pribadi, dan penyebaran *malware*. Untuk melawan ancaman ini, sistem pendeteksian spam pada *chat messenger* menjadi semakin penting. Oleh karena itu, makalah ini bertujuan untuk menerapkan algoritma pencocokan string dalam pengembangan sistem pendeteksian spam pada *chat messenger*. Algoritma pencocokan string yang akan digunakan adalah algoritma KMP (Knuth-Morris-Pratt) dan BM (Boyer-Moore).**

***Kata Kunci*—*Chat messenger*, Sosial media, *Spam*, Pencocokan *string*, KMP, BM, Pendeteksian, Penerapan.**

I. PENDAHULUAN

Dalam era digital saat ini, *chat messenger* seperti LINE, Whatsapp, Telegram, dll telah menjadi salah satu alat komunikasi yang paling penting dan populer. Namun, kemunculan pesan spam telah menjadi masalah yang merugikan pengguna dengan membanjiri *chat messenger* dengan pesan yang tidak diinginkan. Pesan spam tidak hanya mengganggu pengguna, tetapi juga dapat menyebabkan kerugian finansial, kehilangan data pribadi, dan penyebaran malware. Oleh karena itu, sistem pendeteksian *spam* pada *chat messenger* menjadi semakin penting.

Pada makalah ini, penulis bertujuan untuk menerapkan algoritma pencocokan string dalam pengembangan sistem pendeteksian spam pada *chat messenger*. Algoritma pencocokan string yang akan digunakan adalah algoritma KMP (Knuth-Morris-Pratt) dan BM (Boyer-Moore), yang dikenal efisien dalam mencari pola teks dalam sebuah teks. Algoritma KMP didasarkan pada konsep pemrosesan prapemrosesan, sementara algoritma BM menggunakan pendekatan pemrosesan mundur dan heuristik.



Gambar 1.1 Pesan Spam
Sumber: Google Image

Penerapan algoritma KMP dan BM dalam sistem pendeteksian spam pada *chat messenger* diharapkan dapat meningkatkan kecepatan dan keakuratan dalam mengidentifikasi pola spam yang telah dikenali. Dengan menggunakan kedua algoritma ini, sistem dapat dengan cepat dan efisien membandingkan pola teks dalam pesan yang masuk dengan daftar pola spam yang telah diketahui sebelumnya. Hal ini akan membantu dalam mengklasifikasikan pesan sebagai spam atau bukan dengan tingkat keakuratan yang tinggi.

Manfaat dari makalah ini adalah memberikan solusi yang efisien dalam mengatasi masalah pesa spam. Dengan mengimplementasikan algoritma KMP dan BM dalam sistem pendeteksian spam, pengguna *chat messenger* dapat mengurangi jumlah pesan spam yang masuk, meningkatkan efisiensi dalam mengelola *chat messenger*, dan melindungi diri dari ancaman yang terkait dengan pesan spam.

Makalah ini terdiri dari beberapa bagian. Pertama, terdapat pendahuluan yang memberikan gambaran tentang permasalahan pesan spam dan pentingnya pengembangan sistem pendeteksian yang efektif. Bagian selanjutnya adalah tinjauan literatur yang mencakup topik-topik seperti pesan spam, pendeteksian *spam*, serta algoritma pencocokan *string* KMP dan BM yang relevan. Selanjutnya, hasil dan analisis makalah terkait implementasi algoritma disajikan untuk memberikan pemahaman yang lebih mendalam. Makalah ini diakhiri dengan kesimpulan yang mencakup temuan utama dan saran / rekomendasi untuk penelitian selanjutnya dalam bidang ini serta ucapan terima kasih.

Dengan makalah ini, penulis berharap dapat memberikan kontribusi dalam menghadapi pesan spam dan meningkatkan pengalaman pengguna dalam menggunakan *chat messenger*. Penggunaan algoritma KMP dan BM dalam sistem pendeteksian spam pada *chat messenger* diharapkan dapat meningkatkan performa sistem dan memberikan solusi yang lebih handal dalam menghadapi pesan spam.

II. Landasan Teori

A. Chat Messenger

Chat messenger adalah perangkat lunak yang menyediakan fasilitas pengiriman pesan singkat secara langsung antara dua orang atau lebih melalui teks yang diketik. Pengguna dapat mengirim dan menerima pesan teks dengan cepat melalui perangkat elektronik, seperti *gadget* atau komputer, yang terhubung ke jaringan, seperti Internet. *Chat messenger* telah menjadi alternatif populer bagi komunikasi konvensional seperti surat atau *email*. Dibandingkan dengan *email*, *chat messenger* menawarkan keunggulan dalam hal kecepatan dan komunikasi *real-time*. Pesan yang dikirim melalui *chat messenger* dapat tiba dengan cepat dan pengguna dapat melihat respons secara langsung, memungkinkan interaksi yang lebih instan dan efisien. Dengan kemampuan pengiriman pesan secara *real-time*, *chat messenger* telah mengubah cara orang berkomunikasi dan memberikan solusi yang lebih cepat dan interaktif dalam berkomunikasi jarak jauh.

Gambar 2.1 Chat Messenger
Sumber: Google Image

B. String

String adalah suatu urutan karakter yang terdiri dari huruf, angka, tanda baca, dan karakter valid lainnya. Biasanya string diimplementasikan sebagai *array of char* yang menyimpan serangkaian elemen *char* dengan enkoding karakter tertentu. Umumnya string terdapat di dalam dua tanda kutip ganda (*double quote*) dan diakhiri dengan *null terminator* atau karakter '\0'. Selain itu, terdapat beberapa istilah yang terkait dengan string, antara lain:

1. *Prefix*: *substring* dari *string* S yang dimulai dari huruf pertama sampai huruf ke-k, dengan k adalah index apapun diantara 0 sampai m-1 dengan m adalah panjang *string*.
2. *Suffix*: *substring* dari string S yang dimulai dari huruf ke-k sampai huruf terakhir, dengan k adalah index apapun diantara 0 sampai m-1 dengan m adalah panjang *string*.

C. Algoritma String Matching

Pencocokan *string*, juga dikenal sebagai *string matching*, merupakan sebuah algoritma yang digunakan untuk mencari keberadaan suatu *string*, yang disebut sebagai pola (*pattern*), dalam sebuah teks. Pola ini merupakan serangkaian karakter yang diharapkan muncul dalam teks tersebut, sehingga panjang pola harus lebih pendek dibandingkan panjang teks. Algoritma pencocokan string akan memeriksa apakah pola tersebut terdapat dalam teks yang diberikan. Algoritma ini memiliki berbagai penerapan, seperti pendeteksi plagiarisme, forensik digital, pemeriksaan ejaan, analisis citra, dan lain sebagainya.

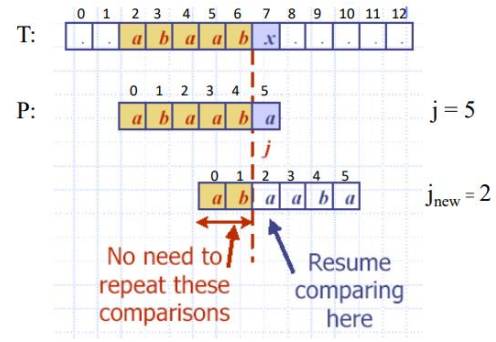
Terdapat beberapa algoritma yang dapat digunakan dalam implementasi pencocokan string, di mana setiap algoritma memiliki pendekatan dan alur berpikir berbeda. Beberapa algoritma pencocokan string yang terkenal meliputi:

1. Algoritma KMP (Knuth-Morris-Pratt): Algoritma ini menggunakan prapemrosesan pola untuk menghindari pengulangan yang tidak perlu dalam pencarian pola pada teks.
2. Algoritma BM (Boyer-Moore): Algoritma ini menggunakan pendekatan pemrosesan mundur dan heuristik untuk mempercepat proses pencarian pola dalam teks.

Pada bagian selanjutnya, penjelasan lebih rinci mengenai kedua tersebut akan dijabarkan.

D. Algoritma KMP (Knuth-Morris-Pratt)

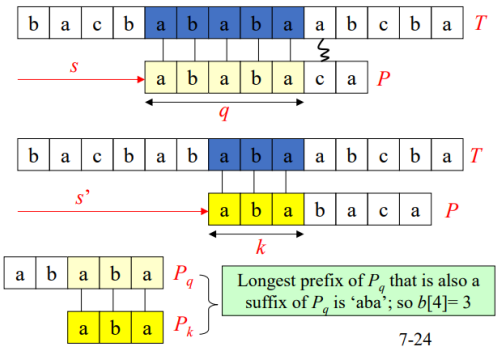
Algoritma Knuth-Morris-Pratt adalah salah satu algoritma pencarian *string* yang dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, tetapi keduanya mempublikasikannya secara bersamaan pada tahun 1977.



Gambar 2.2 KMP Tidak Mengulangi Perbandingan
Sumber: [1]

Algoritma ini melakukan pencocokan *string* dari kiri ke kanan (seperti algoritma Brute Force). Tetapi, algoritma ini melakukan pergeseran yang lebih cerdas dibandingkan dengan algoritma Brute Force. Hal tersebut dilakukan untuk mengurangi perbandingan yang sia-sia.

Pada algoritma KMP, terdapat fungsi pembantu yang disebut sebagai *border function* b(k). *Border function* b(k) didefinisikan dengan ukuran *prefix* terbesar dari P[0..k] yang juga merupakan *suffix* dari P[1..k]. Fungsi ini dibutuhkan untuk mengetahui jumlah pergeseran maksimal yang dapat dilakukan agar tidak ada perbandingan yang sia-sia. *Border function* ini seringkali diterapkan sebagai sebuah *array*.



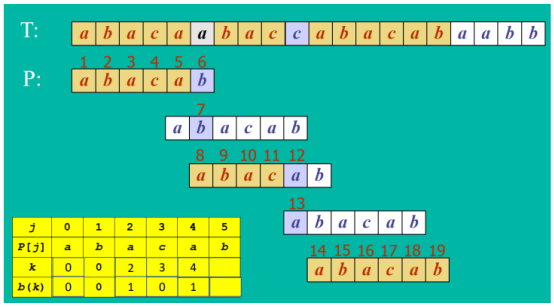
Gambar 2.3 Pencocokan String dengan Algoritma KMP
Sumber: [1]

Misalkan *pattern* disimpan di dalam variabel P. Langkah pertama yang dilakukan algoritma ini adalah dengan membuat larik yang berisikan nilai *border function* b(k) untuk setiap kemungkinan posisi ketidakcocokan yang terjadi di P[].

Dengan j adalah ketidakcocokan yang terjadi di P[] dan k adalah posisi sebelum ketidakcocokan (k = j - 1). *Border function* b(k) didefinisikan sebagai *prefix* terpanjang dari P[0..k] yang juga merupakan *suffix* dari P[1..k]. Berikut ini adalah contoh dari border function pada pattern "abacab".

j	0	1	2	3	4	5
P[j]	a	b	a	c	a	b
k	0	0	2	3	4	
b(k)	0	0	1	0	1	

Dengan menggunakan border function tersebut, algoritma ini memodifikasi pergeseran pada algoritma Brute Force. Misalnya jika ketidakcocokan karakter terjadi pada indeks ke-j, maka ubah nilai j menjadi b(k) dengan k = j – 1.



Gambar 2.4 Contoh Penerapan Algoritma KMP
Sumber: [1]

Dengan panjang karakter dari *pattern* dilambangkan sebagai m dan panjang karakter dari teks dilambangkan sebagai n maka kompleksitas waktu algoritma KMP adalah:

- Menghitung fungsi pinggiran: O(m).
- Pencarian *string*: O(n).
- Kompleksitas waktu algoritma KMP: O(m+n).

Algoritma KMP memiliki kompleksitas waktu yang sangat cepat bila dibandingkan algoritma Brute Force. Algoritma KMP memiliki keuntungan tidak perlu bergerak mundur dalam input teks, sehingga membuat algoritma ini bagus untuk pemrosesan file sangat besar. Namun, algoritma KMP memiliki kekurangan juga, yaitu algoritma ini tidak berfungsi dengan baik apabila ukuran alfabet / teks meningkat, sehingga lebih banyak kemungkinan ketidakcocokan dan ketidaksesuaian cenderung terjadi di awal pola, tetapi algoritma KMP lebih cepat ketika ketidak sesuaian terjadi kemudian.

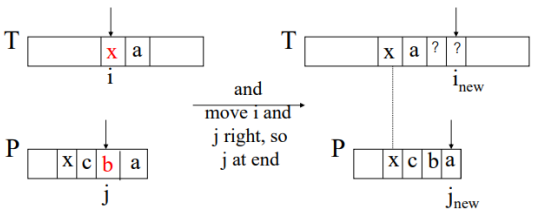
E. Algoritma BM (Boyer-Moore)

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977. Algoritma ini mulai mencocokkan karakter dari sebelah kanan pattern.

Misalkan *pattern* P akan dicocokkan dengan teks T. Algoritma Boyer-Moore melakukan pencocokan string menggunakan dua teknik, yaitu:

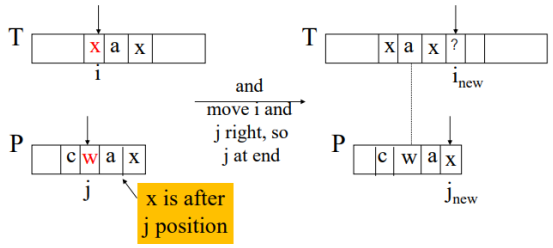
1. Teknik *looking-glass*
   Mencari P di T dengan bergerak ke belakang melewati P, dimulai dari huruf terakhir P.
2. Teknik *character-jump*
   Dilakukan ketika karakter yang dicocokkan tidak sama. Misalkan huruf yang tidak sama pada T adalah x. Terdapat 3 kasus:

   Kasus 1:
   Jika P mengandung x, cobalah geser P ke kanan untuk menyesuaikan posisi x di P dengan T[i].



Gambar 2.5 Kasus 1 Algoritma BM
Sumber: [1]

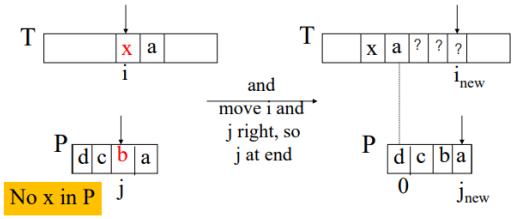
Kasus 2:
Jika P mengandung x, namun penggeseran ke kemunculan terakhir tidak mungkin, maka geser P sebesar 1 karakter ke T[i+1].



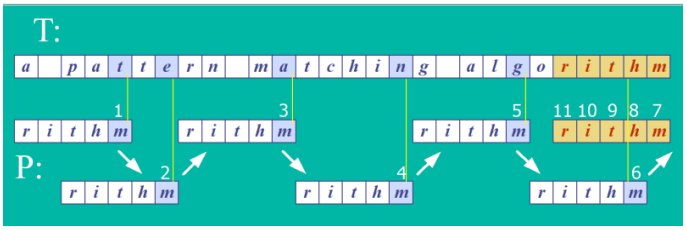
Gambar 2.6 Kasus 2 Algoritma BM
Sumber: [1]

Kasus 3:

Jika kasus 1 dan kasus 2 tidak memenuhi, geser P sehingga menyesuaikan P[0] dengan T[i+1].



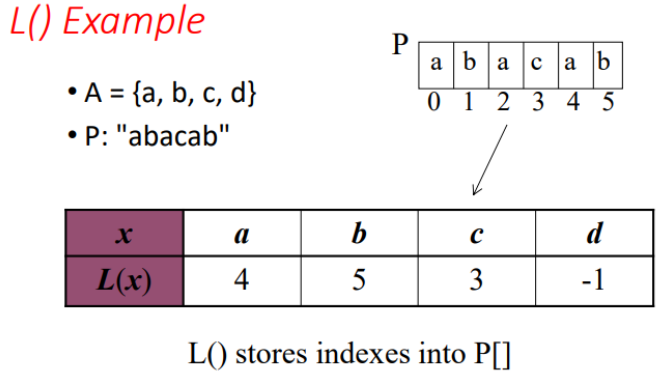
Gambar 2.7 Kasus 3 Algoritma BM
Sumber: [1]



Gambar 2.8 Contoh 1 Penerapan Algoritma BM
Sumber: [1]

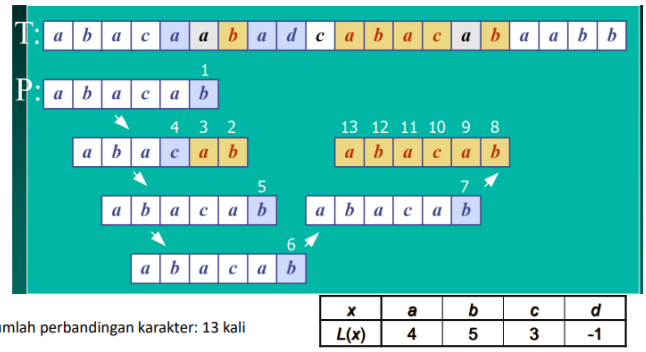
Algoritma Boyer-Moore melakukan *preprocess pattern* P dan alfabet A untuk membentuk fungsi *last occurrence* L(). Fungsi ini memetakan semua huruf dalam alfabet A menjadi *integer*.

Fungsi L(x), x adalah sebuah huruf dalam A, didefinisikan sebagai indeks terbesar sehingga P[i] == x atau -1 jika tidak terdapat index yang memenuhi.



Gambar 2.9 Contoh Fungsi Last Occurrence
Sumber: [1]

Di dalam algoritma Boyer-Moore, fungsi *last occurrence* dihitung saat *pattern* P telah dibaca. Biasanya fungsi *last occurrence* ini disimpan dalam bentuk array.



Gambar 2.10 Contoh 2 Penerapan Algoritma BM
Sumber: [1]

Dengan panjang karakter dari *pattern* dilambangkan sebagai m dan panjang karakter dari teks dilambangkan sebagai n maka kompleksitas waktu *running time* terburuk algoritma BM adalah O(nm + A).

Algoritma BM lebih cepat dari algoritma Brute Force apabila alfabet (A) lebih besar. Namun, algoritma BM lambat apabila alfabet yang dicocokkan kecil. Algoritma BM juga secara signifikan lebih cepat untuk mencari teks berbahasa Inggris, namun lambat apabila mencari teks *binary*.

F. Spam

Spam adalah penggunaan perangkat elektronik untuk mengirim pesan secara masif tanpa diinginkan oleh penerimanya. Ini mencakup spam surat elektronik, spam pesan instan, dan spam telepon genggam. Spam memiliki biaya operasional rendah dan dapat mengganggu pengguna internet serta menyebabkan ketidaknyamanan. Istilah "spamming" digunakan untuk tindakan spam, sedangkan "spammers" adalah orang yang menciptakan spam elektronik.

III. APLIKASI PENCOCOKAN STRING

Program yang dibuat akan mengimplementasikan algoritma Knuth-Morris-Pratt dan Boyer-Moore dan ditulis dalam bahasa pemrograman *python*. Kedua algoritma akan digunakan dan diberikan hasil ujinya untuk menentukan apakah pesan input merupakan spam atau tidak. Sebagai catatan tambahan, kata-kata yang dijadikan *pattern* dan *threshold* penentuan apakah pesan tersebut spam atau tidak telah ditentukan oleh penulis. Hal ini akan dijelaskan dalam bagian berikutnya.

A. Alur Program

1. Menentukan *database* kata-kata pemicu spam

   Pertama-tama, penulis menentukan terlebih dahulu kata-kata yang paling sering digunakan pada pesan spam. Kata-kata ini akan digunakan sebagai *pattern* untuk pencocokan *string*. Dapat dilihat pada tabel berikut.

Buy	Prize
Limited	Offer

Discount	Promotion
Cash	Winner
Exclusive	Guaranteed
Free	Click
Amazing	Opportunity
Money	Urgent
Check	Save
Investment	Profits
Online	Reward
Jackpot	Work
Income	Earn

Tabel 3.1 Tabel Kata Spam

2. Menentukan *threshold* pencocokan string

   Threshold yang digunakan dalam program adalah sebesar 0.5. Hal ini berarti bahwa apabila terdapat lebih dari setengah kata dari pesan input yang *match* dengan *database* kata-kata spam, maka pesan input tersebut terindikasi spam.

3. Input pesan yang ingin diuji

   Pengguna diminta untuk memasukkan pesan yang akan diuji apakah terindikasi spam atau tidak.

4. Pilih algoritma yang ingin digunakan

   Pengguna diminta untuk memilih algoritma yang akan digunakan dalam pencocokan string, yaitu Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM).

5. Hitung total kata dalam pesan input

   Pesan input dihitung jumlah kata-katanya untuk keperluan analisis rasio kecocokan.

6. Ubah pesan input ke huruf kecil

   Pesan input diubah menjadi huruf kecil agar pencocokan tidak bersifat *case sensitive*. Hal ini dilakukan agar kata-kata dalam *database* spam juga diubah menjadi huruf kecil saat dibandingkan.

7. Lakukan pencocokan string menggunakan algoritma yang dipilih

   Proses pencocokan string dilakukan dengan menggunakan algoritma KMP atau BM tergantung pada pilihan pengguna. Algoritma ini digunakan untuk mencocokkan pesan input dengan setiap kata dalam *database* spam.

8. Hitung rasio kecocokan

   Jumlah kata yang cocok dalam pesan input dibandingkan dengan total kata dalam pesan untuk mendapatkan rasio kecocokan. Rasio kecocokan ini akan menjadi faktor dalam menentukan apakah pesan terindikasi sebagai spam atau tidak.

9. Bandingkan rasio kecocokan dengan *threshold*

   Rasio kecocokan dibandingkan dengan *threshold* yang telah ditentukan sebelumnya. Jika rasio kecocokan melebihi *threshold* yang telah ditentukan, maka pesan akan terindikasi sebagai spam.

10. Tampilkan hasil pencocokan, rasio pencocokan, dan status spam atau bukan

    Hasil dari analisis pencocokan ditampilkan kepada pengguna, termasuk jumlah kata serupa, rasio kecocokan, serta status apakah pesar terindikasi sebagai spam atau tidak.

11. Tampilkan waktu eksekusi dalam satuan milidetik

    Waktu eksekusi algoritma pencocokan string ditampilkan kepada pengguna dalam satuan milidetik untuk memberikan gambaran tentang kinerja algoritma tersebut.

B. Implementasi Algoritma String Matching

1. Algoritma Knuth-Morris-Pratt (KMP)

```python
# Fungsi untuk membuat tabel lompatan KMP
def generate_kmp_table(pattern):
    table = [0] * len(pattern)
    prefix_length = 0

    for i in range(1, len(pattern)):
        while prefix_length > 0 and pattern[i] != pattern[prefix_length]:
            prefix_length = table[prefix_length - 1]

        if pattern[i] == pattern[prefix_length]:
            prefix_length += 1

        table[i] = prefix_length

    return table

# Fungsi untuk mencocokkan string menggunakan algoritma KMP
def kmp_search(text, pattern):
    start_time = time.time()
    m = len(pattern)
    n = len(text)
    table = generate_kmp_table(pattern)
    matches = []
    comparisons = 0  # Total komparasi huruf

    i, j = 0, 0

    while i < n:
        # Menambahkan 1 pada setiap komparasi huruf yang dilakukan
        comparisons += 1

        if pattern[j] == text[i]:
            i += 1
            j += 1

            if j == m:
                matches.append(i - j)
                j = table[j - 1]
        else:
            if j != 0:
                j = table[j - 1]
            else:
                i += 1

    execution_time = time.time() - start_time
    return matches, execution_time, comparisons
```

2. Algoritma Boyer-Moore (BM)

```python
# Fungsi untuk membuat tabel lompatan BM
def generate_bm_table(pattern):
    table = {}

    for i in range(len(pattern) - 1):
        table[pattern[i]] = len(pattern) - 1 - i

    return table

# Fungsi untuk mencocokkan string menggunakan algoritma BM
def bm_search(text, pattern):
    start_time = time.time()
    m = len(pattern)
    n = len(text)
    table = generate_bm_table(pattern)
    matches = []
    comparisons = 0  # Total komparasi huruf

    i = m - 1

    while i < n:
        # Menambahkan 1 pada setiap komparasi huruf yang dilakukan
        comparisons += 1

        j = m - 1
        k = i

        while j >= 0 and text[k] == pattern[j]:
            # Menambahkan 1 pada setiap komparasi huruf yang dilakukan
            comparisons += 1
            j -= 1
            k -= 1

        if j == -1:
            matches.append(k + 1)
            i += m
        else:
            if text[k] in table:
                i += max(table[text[k]], m - j)
            else:
                i += m

    execution_time = time.time() - start_time
    return matches, execution_time, comparisons
```

C. Pengujian dan Analisis

Pada makalah ini, penulis melakukan 6 kali pengujian dengan 6 pesan input berbeda, 3 diantaranya merupakan pesan spam dan 3 lainnya merupakan bukan pesan spam. Untuk setiap pesan input akan diuji menggunakan Algoritma KMP dan BM untuk memastikan hasil yang didapatkan sekaligus menguji kebenaran masing-masing algoritma yang diterapkan pada program.

1. Pengujian 1
   "Click to check our exclusive promotion. Claim your cash reward now!" (spam).

   KMP:

   ```
   Masukkan pesan: Click to check our exclusive promotion. Claim your cash reward now!
   Masukkan algoritma: KMP

   Total kalkulasi KMP: 1817
   Total kata serupa: 6
   Rasio kecocokan: 0.5454545454545454
   Pesan ini terindikasi sebagai spam.
   Waktu eksekusi: 0.9999275207519531 ms
   ```

   BM:

   ```
   Masukkan pesan: Click to check our exclusive promotion. Claim your cash reward now!
   Masukkan algoritma: BM

   Total kalkulasi BM: 397
   Total kata serupa: 6
   Rasio kecocokan: 0.5454545454545454
   Pesan ini terindikasi sebagai spam.
   Waktu eksekusi: 0.0 ms
   ```

2. Pengujian 2
   "Limited time offer! Get an exclusive discount. Click here for amazing deals!" (spam).

   KMP:

   ```
   Masukkan pesan: Limited time offer! Get an exclusive discount. Click here for amazing deals!
   Masukkan algoritma: KMP

   Total kalkulasi KMP: 2048
   Total kata serupa: 6
   Rasio kecocokan: 0.5
   Pesan ini terindikasi sebagai spam.
   Waktu eksekusi: 0.9937286376953125 ms
   ```

   BM:

   ```
   Masukkan pesan: Limited time offer! Get an exclusive discount. Click here for amazing deals!
   Masukkan algoritma: BM

   Total kalkulasi BM: 450
   Total kata serupa: 6
   Rasio kecocokan: 0.5
   Pesan ini terindikasi sebagai spam.
   Waktu eksekusi: 0.0 ms
   ```

3. Pengujian 3
   "Check our online investment program and start earning guaranteed profits. Act now!" (spam).

   KMP:

   ```
   Masukkan pesan: Check our online investment program and start earning guaranteed profits. Act now!
   Masukkan algoritma: KMP

   Total kalkulasi KMP: 2207
   Total kata serupa: 6
   Rasio kecocokan: 0.5
   Pesan ini terindikasi sebagai spam.
   Waktu eksekusi: 1.0015964508885664 ms
   ```

   BM:

   ```
   Masukkan pesan: Check our online investment program and start earning guaranteed profits. Act now!
   Masukkan algoritma: BM

   Total kalkulasi BM: 497
   Total kata serupa: 6
   Rasio kecocokan: 0.5
   Pesan ini terindikasi sebagai spam.
   Waktu eksekusi: 0.0 ms
   ```

4. Pengujian 4
   "Hey, how are you doing? Just wanted to check in and see how you're doing. Let's catch up soon!" (bukan spam).

   KMP:

   ```
   Masukkan pesan: Hey, how are you doing? Just wanted to check in and see how you're doing. Let's catch up soon!
   Masukkan algoritma: KMP

   Total kalkulasi KMP: 2534
   Total kata serupa: 1
   Rasio kecocokan: 0.052631578947368421
   Pesan ini bukan spam.
   Waktu eksekusi: 1.5063285827636719 ms
   ```

   BM:

   ```
   Masukkan pesan: Hey, how are you doing? Just wanted to check in and see how you're doing. Let's catch up soon!
   Masukkan algoritma: BM

   Total kalkulasi BM: 586
   Total kata serupa: 1
   Rasio kecocokan: 0.052631578947368421
   Pesan ini bukan spam.
   Waktu eksekusi: 0.0 ms
   ```

5. Pengujian 5
   "Hi, I'm interested in your online work opportunity. Can you tell me more about it?" (bukan spam).

   KMP:

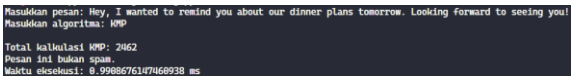
   ```
   Masukkan pesan: Hi, I'm interested in your online work opportunity. Can you tell me more about it?
   Masukkan algoritma: KMP

   Total kalkulasi KMP: 2211
   Total kata serupa: 3
   Rasio kecocokan: 0.2
   Pesan ini bukan spam.
   Waktu eksekusi: 0.9486675262451172 ms
   ```

   BM:

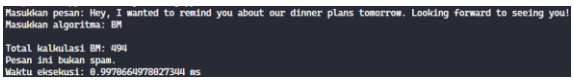
   ```
   Masukkan pesan: Hi, I'm interested in your online work opportunity. Can you tell me more about it?
   Masukkan algoritma: BM

   Total kalkulasi BM: 460
   Total kata serupa: 3
   Rasio kecocokan: 0.2
   Pesan ini bukan spam.
   Waktu eksekusi: 0.0 ms
   ```

6. Pengujian 6
   "Hey, I wanted to remind you about our dinner plans tomorrow. Looking forward to seeing you!" (bukan spam).

   KMP:



   BM:



Berikut adalah tabel ringkasan hasil pengujian.

Pengujian	KMP	BM
1	Kalkulasi: 1817 Serupa: 6 Rasio: 0.54 Waktu: 0.99ms **Spam**	Kalkulasi: 397 Serupa: 6 Rasio: 0.54 Waktu: <0.01ms **Spam**
2	Kalkulasi: 2048 Serupa: 6 Rasio: 0.5 Waktu: 0.99ms **Spam**	Kalkulasi: 450 Serupa: 6 Rasio: 0.5 Waktu: <0.01ms **Spam**
3	Kalkulasi: 2207 Serupa: 6 Rasio: 0.5 Waktu: 1ms **Spam**	Kalkulasi: 497 Serupa: 6 Rasio: 0.5 Waktu: <0.01ms **Spam**
4	Kalkulasi: 2534 Serupa: 1 Rasio: 0.05 Waktu: 1.5ms **Bukan spam**	Kalkulasi: 506 Serupa: 1 Rasio: 0.05 Waktu: <0.01ms **Bukan spam**
5	Kalkulasi: 2211 Serupa: 3 Rasio: 0.2 Waktu: 0.99ms **Bukan spam**	Kalkulasi: 460 Serupa: 3 Rasio: 0.2 Waktu: <0.01ms **Bukan spam**
6	Kalkulasi: 2462 Serupa: 0 Rasio: 0.0 Waktu: 1ms **Bukan spam**	Kalkulasi: 494 Serupa: 0 Rasio: 0.0 Waktu: <0.01ms **Bukan spam**

Tabel 3.2 Tabel Ringkasan Pengujian

Berdasarkan hasil pengujian yang telah dilakukan, kedua algoritma sudah benar dalam mendeteksi pesan spam, dapat dilihat dari hasil penentuan pesan spam yang sama (dicetak tebal). Dari segi kalkulasi, algoritma BM memiliki jumlah kalkulasi pencocokan huruf yang lebih sedikit dibandingkan algoritma KMP. Hal ini juga akan berdampak kepada waktu eksekusi program yang lebih cepat. Dari segi pendeteksian kata yang serupa dan rasio kata spam yang dihasilkan, kedua algoritma dapat menghasilkan hasil yang sama.

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Dalam makalah ini, penulis telah menerapkan algoritma pencocokan string, yaitu algoritma KMP (Knuth-Morris-Pratt) dan BM (Boyer-Moore), dalam pengembangan sistem pendeteksian spam pada *chat messenger*. Implementasi algoritma ini bertujuan untuk meningkatkan kecepatan dan keakuratan dalam mengidentifikasi pola spam yang telah dikenali. Dengan menggunakan kedua algoritma ini, sistem dapat dengan cepat dan efisien membandingkan pola teks dalam pesan chat dengan daftar pola spam yang telah diketahui sebelumnya. Makalah ini memberikan solusi yang efisien dalam mengatasi masalah spam pada *chat messenger*, membantu pengguna dalam mengurangi jumlah pesan spam yang masuk, meningkatkan efisiensi dalam berkomunikasi, dan melindungi pengguna dari risiko yang terkait dengan spam.

B. Saran

Dalam pengembangan selanjutnya, terdapat beberapa saran yang dapat diperhatikan:

1. Penelitian lebih lanjut mengenai algoritma pencocokan string: Meskipun telah diterapkan algoritma KMP dan BM dalam makalah ini, masih terdapat banyak algoritma pencocokan string lainnya yang dapat dieksplorasi. Penelitian lebih lanjut dapat melibatkan algoritma-algoritma tersebut untuk membandingkan kinerja dan efektivitasnya dalam sistem pendeteksian spam.
2. Pengujian lebih lanjut dengan dataset yang lebih luas: Untuk menguji kehandalan dan kinerja sistem secara menyeluruh, dianjurkan untuk melibatkan dataset yang lebih luas dan beragam. Penggunaan dataset yang representatif akan memberikan hasil yang lebih akurat dan dapat membantu dalam mengevaluasi kemampuan sistem dalam mengenali pola spam dengan tingkat keberhasilan yang tinggi.
3. Perbaikan antarmuka pengguna: Pengembangan sistem pendeteksian spam pada *chat messenger* juga dapat melibatkan perbaikan pada antarmuka pengguna. Antarmuka yang intuitif dan mudah digunakan akan memberikan pengalaman pengguna yang lebih baik dan memudahkan pengguna dalam mengelola pesan chat.

Dengan mengimplementasikan saran-saran tersebut, diharapkan sistem pendeteksian spam pada *chat messenger* dapat terus berkembang dan memberikan solusi yang lebih efektif dalam mengatasi masalah spam serta meningkatkan pengalaman pengguna dalam berkomunikasi secara aman dan efisien.

V. UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan Yang Maha Esa karena atas berkat dan rahmatnya, sehingga penulis dapat menyelesaikan makalah dengan tepat waktu. Harapan penulis adalah semoga ilmu ini dapat bermanfaat bagi pembaca.

Ucapan terima kasih kepada dosen pengampu mata kuliah IF2211 Strategi Algoritma K02 2022/2023, Dr. Nur Ulfa Maulidevi, S.T., M.Sc., atas segala ilmu dan pedomen yang telah diberikan sehingga makalah ini dapat diselesaikan dengan baik.

REFERENSI

[1] Munir, Rinaldi. 2021. "Pencocokan String (*String/Pattern Matching*)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. 22 Mei 2023.

[2] Wahyupjl. 2018. "Aplikasi Messenger". https://eventkampus.com/blog/detail/1408/aplikasi-messenger. 22 Mei 2023.

[3] Busbee, Kenneth Leroy & Dave Braunscheweig. "String Data Type". https://press.rebus.community/programmingfundamentals/chapter/string-data-type/. 22 Mei 2023.

[4] Knuth, Donald; Morris, James H.; Pratt, Vaughan. 1977. "Fast pattern matching in strings". SIAM Journal on Computing. 22 Mei 2023.

[5] Boyer, Robert S.; Moore, J Strother. 1977. "A Fast String Searching Algorithm". Comm. ACM. New York: Association for Computing Machinery. 22 Mei 2023.

[6] Abdi, Husnul. 2021. "Apa itu Spam? Kenali Pengertian, Bentuk, dan Penjelasannya". https://www.liputan6.com/hot/read/4624766/apa-itu-spam-kenali-pengertian-bentuk-dan-penjelasannya. 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

Ghazi Akmal Fauzan 13521058