Penerapan Algoritma *String Matching* dalam Pencarian Kata Asing

Muhammad Rifko Favian - 13521075
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521075@std.stei.itb.ac.id

***Abstract*—Dalam dokumentasi seperti karya ilmiah, penulisan untuk kata asing, seperti kata berbahasa inggris, berbahasa yunani dan bahasa lainya dibuat dengan memiringkan kata tersebut. Untuk menemukan kata asing tersebut, perlu sebuah cara atau algoritma agar dapat dilakukan otomatis atau tidak dengan mencari satu kata tersebut. Algoritma itu merupakan algoritma *string matching*. Algoritma ini bertugas mencari suatu *pattern* atau pola di dalam suatu teks. Penelitian ini akan memperlihatkan bagaimana algoritma *string matching* dapat digunakan untuk mencari kata asing yang berada di dalam suatu teks**

***Keywords—string matching*; knuth-morris-pratt; boyer-moore ; asing; kata asing; pencarian**

I. Pendahuluan

Bahasa Indonesia merupakan bahasa nasional dan resmi negara Indonesia yang dipakai sehari-hari oleh penduduknya di samping pemakaian bahasa daerah. Penggunaan bahasa Indonesia dalam konteks formal harus mengikuti aturan Panduan Umum Ejaan Bahasa Indonesia (PUEBI). Dalam aturan tersebut, ada penggunaan huruf yang salah satunya ialah untuk penulisan huruf miring. Salah satu aturan untuk menulis huruf miring ialah apabila ada penuliskan kata atau ungkapan dalam bahasa daerah atau bahasa asing.

Namun, banyak dari kita yang tidak mengikuti atau lalai dari aturan tersebut. Terlebih, apabila kita sudah terlanjur menulis banyak hal di dalam suatu teks, maka kemungkinan kita harus cari satu per satu kata yang merupakan kata asing untuk diubah menjadi huruf miring. Maka dari itu, penulis melakukan percobaan untuk membuat sebuah algoritma agar kata asing tersebut dapat ditemukan seara otomatis.

Salah satu cara pencarian kata asing ini adalah dengan menggunakan algoritma *string matching*. Pada makalah ini, akan dijelaskan dasar dari algoritma tersebut dan bagaimana cara kerjanya. Kemudian akan disediakan sampel pengujian dalam penggunaannya dalam mencari kata asing di dalam suatu kalimat atau teks, sekaligus waktu eksekusi yang dilakukan dalam pencarian ini oleh algoritma tersebut.

II. Landasan Teori

A. *String Matching*

1. Definisi *String Matching*

   String Matching atau pencocokan *string* adalah sebuah algoritma yang digunakan untuk mencari kemunculan suatu *string* atau yang disebut *pattern* dalam sebuah teks. *Pattern* adalah *string* dengan panjang m karakter dan teks adalah (*long*) *string* yang panjangnya n karakter (asumsi m <<< n) yang akan dicari di dalam teks.

   Beberapa penerapan string matching yang sering digunakan ialah untuk pencarian di dalam *editor* teks, *web search engine* (seperti Google), analisis citra, dan *Bionformatics*. Terdapat beberapa algoritma string matching, antara lain Brute Force, Knuth-Morris-Pratt (KMP), Boyer-Moore (BM). Penjelasan mengenai ketiga algorima tersebut akan dijelaskan di bagian-bagian setelah ini.

2. Algoritma Brute Force

   Salah satu algortima proses *string matching* adalah algoritma brute force. Proses kerja dari algoritma brute force ini adalah dengan Periksa setiap posisi dalam teks untuk melihat apakah *pattern* dimulai pada posisi itu.

   Analisis beberapa jenis kasus yang ada yaitu:

   1. Worst case: Jumlah perbandingan = m(n-m+1) = O(mn). Contoh:

      Teks: aaaaaaaaaaaaaaaaaaaaaaaaah

      Pattern: aaah

   2. Best case: Kompleksitasnya O(n). Terjadi bila karakter pertama pattern tidak pernah sama dengan karakter teks yang dicocokkan. Contoh:

      Teks: abcdefghijklmnopqrstuvwxyzzz

      Pattern: zzz

   3. Average case: teks pada umumnya memakan O(m+n), di mana cukup cepat. Contoh:

      Teks: iniadalahcontohabcdstringmatching

      Pattern: abcd

Brute force akan lebih cepat apabila alfabet pada teks banyak (contoh: A..Z , a..z , 1..9 , dll) dan akan menjadi lebih lambat apabila alfabetnya kecil (contoh: binary, 0 dan 1)

3. Algoritma Knuth-Morris-Pratt (KMP)

   Algoritma proses *string matching* yang lainnya ialah algoritma KMP. Algoritma KMP mencari pattern di dalam teks dari pattern bagian kiri ke bagian kanan, seperti Brute Force. Namun, pergeseran di dalam teksnya lebih teroptimasi dibanding Brute Force.

   Algoritma KMP ini adalah algoritma yang dibuat oleh James H. Morris yang kemudian ditemukan kembali oleh Donald Knuth tidak lama setelahnya. Morris dan Vaughan Pratt juga mempublikasikan laporan tentang algoritma ini pada 1970. Pada tahun 1977, Knuth, Morris, dan Pratt mempublikasikan algoritma ini bersama-sama.

   Bedanya KMP dengan Brute Force ini adalah dengan adanya penambahan fungsi pembantu yang dinamakan *border function* atau *failure function* {b(k)}. b(k) didefinisikan sebagai ukuran prefix terbesar dari P[0..k] yang juga merupakan suffix dari P[1..k] di mana P adalah pattern-nya. Jadi awalnya KMP melakukan preprocessing pattern untuk mencari kesamaan prefiks pada pattern dengan pattern itu sendiri. b(k) sendiri direpresantkan sebagai array

   Kompleksitas pada *border function* adalah O(m), sedangkan kompleksitas pada pencocokan string adalah O(n). Maka dari itu, kompleksitas algoritma KMP adalah O(m+n).

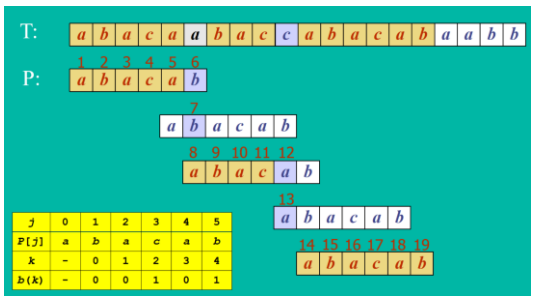

Fig. 1. Ilustrasi Cara Kerja *border function* sebagai fungsi pembantu Algoritma KMP (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf)

   Algoritma ini lebih cepat apabila dibandingkan dengan algoritma brute force, karena algoritma ini mengeliminasi beberapa perbandingan yang kemungkinan sia-sia. Selain itu, algoritma ini juga tidak pernah memundurkan indeks, sehingga cocok untuk memproses file yang diberikan dalam bentuk stream. Namun, KMP akan lebih buruk apabila ukuran alfabet bertambah, karena apabila ada ketidakcocokan pada awal karakter, algoritma ini akan menjadi lebih lambat.

4. Algoritma Boyer-Moore (BM)

   Algoritma BM dikembangkan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1977. Algoritma ini juga merupakan algoritma proses *string matching* yang cukup terkenal dan cukup efektif.

   Algoritma BM ini didasarkan oleh dua teknik:

   1. *The looking-glass technique*
      Mencari *pattern* di dalam teks dari pattern bagian kanan ke bagian kiri (terbalik dengan KMP)

   2. *The character-jump technique*
      Terdiri dari tiga kasus yang terjadi ketika ketidakcocokan ditemukan di karakter pada teks indeks ke-i yang mempunyai karakter x (T[i] = x) dan karakter pada pola P indeks ke-j (P[j]) tidak sama dengan T[i].

      Kasus pertama terjadi ketika terdapat x pada P. Yang dilakukan adalah menggeser P ke kanan sedemikian sehingga dengan kemunculan terakhir dari x pada P sejajar dengan T[i].

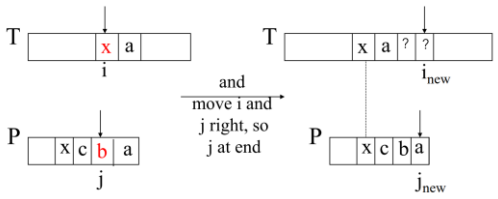

Fig. 2. Ilustrasi Kasus Pertama dari *character-jump technique* (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf)

      Kasus kedua terjadi ketika terdapat x pada P, tetapi pergeseran P ke kanan sehingga seperti kasus pertama tidak memungkinkan. Solusinya adalah menggseser P satu karakter ke kanan sehingga sejajar dengan T[i+1].

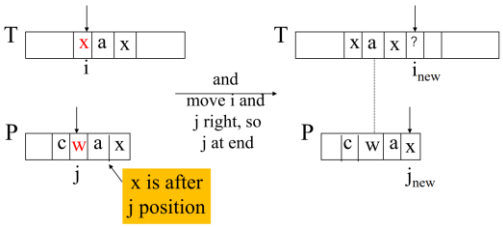

Fig. 3. Ilustrasi Kasus Kedua dari *character-jump technique* (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf)

      Kasus ketiga terjadi ketika kasus pertama dan kasus kedua tidak terjadi. Solusinya adalah menggeser P sehingga P[0] sejajar dengan T[i+1].

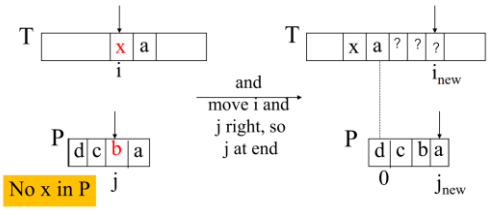

Fig. 4. Ilustrasi Kasus Ketiga dari *character-jump technique* (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf)

Algoritma ini juga terdapat fungsi pembantu yaitu *Last Occurrence Function* yang memproses pattern untuk mencari indeks (dimulai dari 0) kemunculan dari masing-masing alfabet yang ada di dalamnya menjadi L() yang direpresentasikan sebagai array, di mana jika tidak ada suatu alfabet di dalam pattern tersebut, maka indeks yang keluar adalah -1.

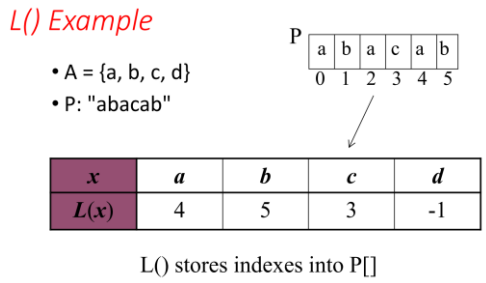

Fig. 5. Ilustrasi Proses Pembuatan L() dalam *Last Occurrence Function* (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf)

Kompleksitas terburuk dari algoritma BM adalah O(nm + A) di mana A adalah semua alfabet yang ada di dalam teks. Namun, BM akan lebih baik ketika alfabetnya besar, sehingga lebih cocok untuk matching pada teks seperti bahasa Inggris.

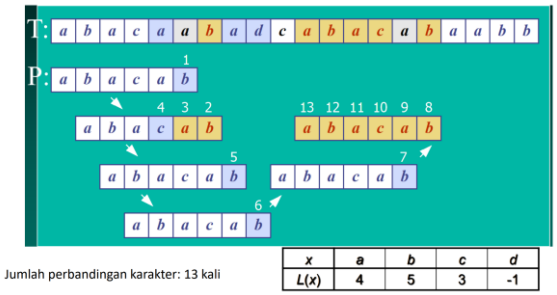

Fig. 6. Ilustrasi Cara Kerja Algoritma BM (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf)

B. *Kata Asing*

Kata asing mengacu pada kata-kata atau frasa yang berasal dari bahasa yang berbeda dengan bahasa yang sedang digunakan atau dibahas. Ketika kita berbicara tentang bahasa asing, ini mengimplikasikan bahwa bahasa tersebut bukanlah bahasa ibu atau bahasa dominan yang digunakan dalam lingkungan tertentu. Contoh seperti saya yang biasa berbicara dan menulis dengan bahasa Indonesia, maka kata-kata yang bukan merupakan bahasa Indonesia, seperti bahasa Inggris, bisa saya sebut kata asing.

Kata asing ini berbeda dengan kata serapan. Kata serapan adalah kata yang diserap dari bahasa lain berdasarkan kaidah bahasa penerima. Jadi kata serapan sudah diintegrasikan ke dalam suatu bahasa dan diterima pemakaiannya secara umum.

Contoh kata asing ialah makanan "sushi" yang merupakan kata dari bahasa Jepang. Sementara contoh kata serapan yaitu "televisi" yang merupakan serapan dari dua bahasa yaitu bahasa Yunani (tele) dan Latin (visio).

III. Penerapan Pencarian Kata Asing

Pada penerapan pencarian kata asing ini akan menggunakan algoritma string matching KMP dan BM dalam mencari kata asing dalam suatu teks, kemudian akan mengeluarkan kata asing apa saja yang ada di dalam teks tersebut (jika ada).

A. *Implementasi Algoritma KMP dan BM*

Penulis mengimplementasi algoritma KMP dan BM ini dalam suatu program dengan menggunakan bahasa Python. Di sini kedua proses string matching sudah diubah menjadi tidak case sensitive.

1. Algoritma KMP

```python
def kmpMatch(text, pattern):
    n = len(text)
    m = len(pattern)
    fail = computeFail(pattern)
    i = 0
    j = 0
    while i < n:
        if text[i].lower() == pattern[j].lower():
            if j == m - 1:
                return i - m + 1
            i += 1
            j += 1
        elif j > 0:
            j = fail[j - 1]
        else:
            i += 1
    return -1

def computeFail(pattern):
    m = len(pattern)
    fail = [0] * m
    j = 0
    i = 1
    while i < m:
        if pattern[j].lower() == pattern[i].lower():
            fail[i] = j + 1
            i += 1
            j += 1
        elif j > 0:
            j = fail[j - 1]
        else:
            fail[i] = 0
            i += 1
    return fail
```

Fig. 7. Implementasi Algoritma KMP (Sumber: Dokumentasi Pribadi)

Implementasi algoritma KMP terbagi menjadi dua fungsi, yaitu fungsi kmpMatch dan fungsi computeFail.

Fungsi computeFail ialah fungsi yang kita ketahui sebagai *border function* atau *failure function*-nya. Fungsi ini mengembalikan sebuah array fail atau yang sudah dijelaskan sebelumnya yaitu disebut b(k). Sementara fungsi kmpMatch ialah fungsi algoritma KMP utamanya, yaitu sebagai string matching. Apabila ditemukan pattern di dalam teks, maka akan dikembalikan indeks ke berapa pattern itu ditemukan. Namun apabila tidak ditemukan pattern di dalam teks, maka akan mengembalikan angka -1.

2. Algoritma BM

```python
def bmMatch(text, pattern):
    last = buildLast(pattern.lower())
    n = len(text)
    m = len(pattern)
    i = m - 1
    if i > n - 1:
        return -1
    j = m - 1
    while True:
        if pattern[j].lower() == text[i].lower():
            if j == 0:
                return i
            else:
                i -= 1
                j -= 1
        else:
            lo = last[ord(text[i].lower())]
            i = i + m - min(j, 1 + lo)
            j = m - 1
        if i > n - 1:
            break
    return -1

def buildLast(pattern):
    last = [-1] * 128
    for i in range(len(pattern)):
        last[ord(pattern[i].lower())] = i
    return last
```

Fig. 8. Implementasi Algoritma BM (Sumber: Dokumentasi Pribadi)

Implementasi algoritma BM terbagi menjadi dua fungsi, yaitu fungsi bmMatch dan fungsi buildLast. Fungsi buildLast ialah fungsi yang kita ketahui sebagai *last occurrence function*-nya. Fungsi ini mengembalikan sebuah array last atau yang sudah dijelaskan sebelumnya yaitu L(). Sementara fungsi bmMatch sebagai algoritma string matching utamanya. Sama seperti pada algoritma KMP tadi, fungsi ini juga mengembalikan indeks di mana pattern tersebut ditemukan jika ada, namun apabila tidak ada yang cocok, maka akan mengembalikan angka -1.

B. *Mekanisme Pengujian*

Pada pengujian, penulis akan mempersiapkan beberapa (tepatnya 120) kata asing yang biasa digunakan di dalam suatu dokumen untuk dijadikan pattern yang akan dicek di dalam suatu teks. Kata-kata ini saya buat dengan bantuan ChatGPT. Untuk mekanisme yang dilakukan ialah sebagai berikut:

- Kata yang diambil akan dijadikan array yang akan digunakan sebagai pattern string matching.
- Pertama program akan meminta untuk memasukkan algoritma apa yang akan digunakan antara KMP atau BM
- Kemudian program akan meminta user untuk menulis suatu teks dengan bahasa Indonesia sebagai bahasa utamanya.
- Teks akan dicek apakah ada kata yang merupakan kata asing, jika ditemukan, maka akan ditambahkan ke sebuah list yang berisi kata asing tersebut

words_alpha.txt

```
Abstract
Acknowledgment
Analysis
Appendix
Conclusion
Definition
Discussion
Document
Evidence
Example
Figure
Graph
Heading
Hypothesis
Illustrate
Introduction
Method
Note
Observation
Paragraph
Proof
Purpose
Reference
Result
Section
Summary
```

Fig. 9. Beberapa *Pattern* Bahasa Asing yang Dibuat (Sumber: Dokumentasi Pribadi)

```python
print("Masukkan teks:")
text = input()

kata_asing = []
for i in range(len(english_words)):
    if algoritma == "1":
        result = stringMatching.kmpMatch(text, english_words[i])
    elif algoritma == "2":
        result = stringMatching.bmMatch(text, english_words[i])
    if result != -1 and english_words[i] not in kata_asing:
        kata_asing.append(english_words[i])
```

Fig. 10. Potongan Kode yang Dibuat pada Tahap *String Matching* (Sumber: Dokumentasi Pribadi)

C. *Hasil Pengujian*

Pengujian akan dilakukan dengan penulis menulis sebuah teks/kalimat dalam bahasa Indonesia dengan

diselipkan beberapa kata asing yang biasa digunakan. Kemudian program akan mengeluarkan list kata asing yang ada di dalam teks tersebut. Jika tidak ada kata asing, program hanya akan mengeluarkan pernyataan bahwa tidak ada kata asing di dalam teks tersebut. Program juga akan mengeluarkan berapa lama waktu eksekusi yang dilakukan dalam proses *string matching* tersebut. Jadi penulis akan menguji menggunakan kedua algoritma sekaligus sebagai perbandingan waktu eksekusi antar algoritma.

1. Pengujian Teks Pertama
   Teks Pertama yang digunakan adalah sebagai berikut:

> Halo, Apa kabar semuanya? Kali ini saya akan mencoba beberapa test case untuk mengetest kata asing apa saja yang ada di suatu teks.

Pertama, program akan di-*test* menggunakan algoritma KMP. Hasilnya ialah sebagai berikut:

```
Kata asing yang ditemukan:
Test, Case
Waktu eksekusi: 3.006458282470703 ms
```

Fig. 11. Keluaran program pada pengujian teks pertama menggunakan algoritma KMP (Sumber: Dokumentasi Pribadi)

Kemudian program akan di-*test* menggunakan algoritma BM. Hasilnya ialah sebagai berikut:

```
Kata asing yang ditemukan:
Test, Case
Waktu eksekusi: 1.9919872283935547 ms
```

Fig. 12. Keluaran program pada pengujian teks pertama menggunakan algoritma BM (Sumber: Dokumentasi Pribadi)

Pada pengujian pertama ini, dapat dilihat bahwa algoritma BM memiliki waktu eksekusi yang lebih cepat dibanding dengan algoritma KMP.

2. Pengujian Teks Kedua
   Teks Kedua yang digunakan adalah sebagai berikut:

> Pada sebuah dokumentasi mengenai penelitian ini, terdapat berbagai elemen yang perlu diperhatikan. Abstract atau ringkasan merupakan bagian awal yang memberikan gambaran singkat tentang isi dokumen tersebut.

Hasil pengujian program menggunakan algoritma KMP ialah sebagai berikut:

```
Kata asing yang ditemukan:
Abstract
Waktu eksekusi: 4.004001617431641 ms
```

Fig. 13. Keluaran program pada pengujian teks kedua menggunakan algoritma KMP (Sumber: Dokumentasi Pribadi)

Kemudian hasil pengujian program menggunakan algoritma BM:

```
Kata asing yang ditemukan:
Abstract
Waktu eksekusi: 2.9990673065185547 ms
```

Fig. 14. Keluaran program pada pengujian teks kedua menggunakan algoritma BM (Sumber: Dokumentasi Pribadi)

Sama seperti pengujian pertama, pada pengujian kedua, waktu eksekusi algoritma BM lebih cepat dibanding dengan waktu eksekusi algoritma KMP.

3. Pengujian Teks Ketiga
   Teks Ketiga yang digunakan ialah sebagai berikut:

> Framework yang digunakan dalam pengembangan aplikasi ini memungkinkan designer dengan mudah mengimplementasikan design yang menarik.

Hasil pengujian program menggunakan algoritma KMP ialah:

```
Kata asing yang ditemukan:
Framework, Design
Waktu eksekusi: 2.954721450805664 ms
```

Fig. 15. Keluaran program pada pengujian teks ketiga menggunakan algoritma KMP (Sumber: Dokumentasi Pribadi)

Kemudian untuk hasil pengujian menggunakan algoritma BM sebagai berikut:

```
Kata asing yang ditemukan:
Framework, Design
Waktu eksekusi: 2.035379409790039 ms
```

Fig. 16. Keluaran program pada pengujian teks ketiga menggunakan algoritma BM (Sumber: Dokumentasi Pribadi)

Sama seperti pengujian sebelum-sebelumnya, di pengujian ketiga ini, waktu eksekusi algoritma BM juga lebih cepat dibanding dengan waktu eksekusi algoritma KMP.

4. Waktu Eksekusi Hasil Pengujian
   Agar mudah dilihat, berikut merupakan tabel waktu eksekusi dari masing-masing pengujian dari masing-masing algoritma:

TABLE I. TABEL WAKTU EKSEKUSI HASIL PENGUJIAN

Pengujian	**Waktu Eksekusi (*milliseconds*)**	
	KMP	**BM**
1	3.006458282470703	1.9919872283935547
2	4.004001617431641	2.9990673065185547
3	2.954721450805664	2.035379409790039

IV. Kesimpulan dan Saran

Berdasarkan hasil pengujian sebelumnya, dapat dilihat bahwa hasil waktu eksekusi algoritma BM lebih cepat dibanding dengan waktu eksekusi algoritma KMP. Hal ini dikarenakan algoritma BM lebih cocok digunakan pada alfabet yang besar seperti teks bahasa Indonesia atau bahasa Inggris, sementara algoritma KMP lebih cocok digunakan pada alfabet yang kecil, misalnya pada biner. Jadi dalam pengujian ini, penggunaan algoritma BM lebih baik dibanding dengan algoritma KMP.

Meskipun begitu, kedua algoritma berhasil mengeluarkan hasil yang sesuai ekspektasi penulis, karena keduanya telah menemukan kata asing yang tepat di dalam teks tersebut. Namun, karena waktu dan pengetahuan yang kurang, program yang penulis bikin ini masih jauh dari sempurna, karena kata asing yang dipersiapkan masih sangat sedikit untuk menjadi program yang sangat baik. Harapannya, salah satu dari pengembangan program ini adalah apabila kita menulis suatu dokumen yang terdapat kata asingnya, kata tersebut dapat dideteksi dan otomatis diubah *font* nya menjadi *italic* sebagaimana aturan penulisan huruf miring yang sebenarnya.

Ucapan Terima Kasih

Di akhir penghujung makalah ini, pertama penulis ingin berterima kasih kepada Tuhan yang Maha Esa atas anugerah dan rahmat-Nya, penulis dapat menyelesaikan makalah ini dengan baik. Penulis juga berterima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T., selaku dosen kelas K1 karena sudah membimbing penulis dengan sangat baik selama perkuliahan Strategi Algoritma berjalan. Terakhir penulis juga berterima kasih kepada seluruh tim dosen mata kuliah Strategi Algoritma yang telah berperan dalam memberi wadah pembelajaran kepada penulis dan mahasiswa lainnya.

Referensi

[1] Munir, Rinaldi. “Pencocokan String (String/Pattern Matching)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses pada 21 Mei 2023

[2] I Knuth, Donald; Morris, James H.; Pratt, Vaughan (1977). "Fast pattern matching in strings". SIAM Journal on Computing. 6 (2): 323–350.

[3] Boyer, Robert S.; Moore, J Strother (October 1977). "A Fast String Searching Algorithm". Comm. ACM. New York: Association for Computing Machinery. 20 (10): 762–772.

[4] Yasa. 2021. “Apa Itu Kata Serapan? Berikut Jenis, Penulisan, dan Contoh Kata Serapan”. https://xerpihan.id/blog/1246/apa-itu-kata-serapan-berikut-jenis-penulisan-dan-contoh-kata-serapan/. Diakses pada 21 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

Muhammad Rifko Favian, 13521075