Analisis Penerapan Algoritma *Boyer-Moore* dan *Knutt-Morris-Pratt* dalam Pengecekan Plagiarisme Antar Dua Dokumen

Yusuf Ardian Sandi - 13522015
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522015@std.stei.itb.ac.id

***Abstrak*—Plagiarisme adalah tindakan mengambil atau menggunakan karya, ide, atau kata-kata orang lain tanpa memberikan kredit atau pengakuan yang semestinya, dan mengklaimnya sebagai karya asli. Plagiarisme dianggap sebagai pelanggaran etika dan integritas akademik, serta sering kali ilegal. Dalam mengatasi hal tersebut, dibuatlah teknologi untuk mengecek tingkat persentase plagiasi dari dua dokumen. Algoritma pencocokan merupakan suatu hal yang wajib ada pada pengecekan plagiasi ini. Makalah ini akan membahas mengenai pengimplementasian algoritma *string matching* dan menganalisis algoritma mana yang lebih baik digunakan diantara algoritma Boyer-Moore dan Knutt-Moris-Pratt.**

***Kata Kunci*—Plagiarisme, pencocokan, *string matching*, dokumen.**

I. PENDAHULUAN

Pengecekan plagiarisme merupakan salah satu tantangan utama dalam dunia akademik dan profesional. Dengan semakin mudahnya akses terhadap informasi melalui internet, risiko plagiarisme semakin tinggi. Plagiarisme tidak hanya merugikan pihak yang asli, tetapi juga merusak integritas akademik dan kepercayaan publik terhadap hasil karya ilmiah dan tulisan profesional. Oleh karena itu, pengembangan metode yang efektif dan efisien untuk mendeteksi plagiarisme menjadi sangat penting.

Algoritma string matching dapat digunakan untuk mencocokkan kedua dokumen dalam pengecekan plagiarisme. Namun, beberapa algoritma string matching dinilai kurang efektif untuk menyelesaikan masalah ini. Salah satu contohnya adalah algoritma pencocokan string brute force, yang meskipun mudah diimplementasikan, memiliki kompleksitas waktu O(n*m), di mana n adalah panjang teks dan m adalah panjang pola. Kompleksitas waktu yang tinggi ini membuat algoritma brute force tidak praktis untuk dokumen berukuran besar karena memerlukan waktu yang sangat lama untuk melakukan pencocokan.

Selain itu, algoritma Rabin-Karp, yang menggunakan hashing untuk menemukan substring dengan cepat, sering menghadapi masalah kolisi hash yang dapat menurunkan keakuratannya. Meskipun algoritma ini memiliki kompleksitas waktu O(n+m) dalam kasus terbaik, kolisi hash dapat menyebabkan pengecekan ulang yang mengurangi efisiensinya, terutama saat menangani teks yang besar dan kompleks. Oleh karena itu, algoritma *string matching* yang akan digunakan pada pengecekan plagiarisme ini ialaha algoritma Boyer-Moore dan Knuth-Morris-Pratt.

Makalah ini berfokus pada penerapan algoritma Boyer-Moore dan Knuth-Morris-Pratt (KMP) untuk mengecek plagiarisme antar dua dokumen. Kedua algoritma ini dikenal dalam bidang string matching dan memiliki kompleksitas waktu yang lebih baik dibandingkan dengan metode konvensional, sehingga memungkinkan pengecekan yang lebih cepat dan akurat pada dokumen yang besar.

Algoritma Boyer-Moore, yang diperkenalkan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1977, menggunakan teknik pencocokan pola dengan memindai teks dari kanan ke kiri dan menggunakan informasi yang diperoleh untuk melompati bagian-bagian teks yang tidak perlu diperiksa. Algoritma ini sangat efisien untuk pencarian string pada teks yang panjang.

Di sisi lain, algoritma Knuth-Morris-Pratt (KMP), yang dikembangkan oleh Donald Knuth, Vaughan Pratt, dan James H. Morris pada tahun 1977, menggunakan tabel prefiks untuk mempercepat proses pencocokan dengan melompati bagian-bagian teks yang telah diketahui cocok. KMP sangat berguna untuk menghindari pencocokan karakter yang tidak perlu.

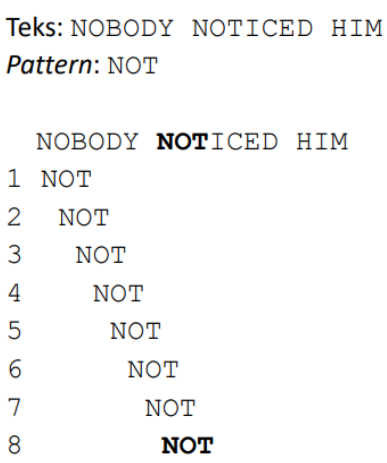
Dalam makalah ini, penulis akan membahas bagaimana kedua algoritma tersebut dapat diterapkan untuk mendeteksi plagiarisme dengan efisien. Penulis akan menjelaskan langkah-langkah detail dari implementasi algoritma Boyer-Moore dan KMP dalam konteks pengecekan plagiarisme, serta mengevaluasi kinerja dan keakuratan dari kedua algoritma tersebut. Dengan menggunakan kedua algoritma ini, diharapkan dapat membantu memerangi praktik plagiarisme dan menjaga integritas karya tulis ilmiah dan profesional.

II. LANDASAN TEORI

A. Algoritma String Matching

Algoritma string matching adalah teknik fundamental dalam ilmu komputer dan rekayasa perangkat lunak yang digunakan untuk menemukan keberadaan suatu pola dalam sebuah teks. Teknik ini esensial dalam berbagai aplikasi, termasuk pencarian teks dalam dokumen, pengecekan plagiarisme, analisis genom, dan kompresi data. Algoritma string matching bekerja dengan cara mencocokkan urutan karakter dari pola (*pattern*) dengan urutan karakter dalam teks (*text*) dan menentukan apakah pola tersebut muncul dalam teks atau tidak.

Algoritma string matching dapat dikategorikan ke dalam beberapa pendekatan, mulai dari yang paling sederhana hingga yang paling kompleks. Metode brute force adalah yang paling sederhana, di mana setiap karakter dalam pola dibandingkan dengan setiap karakter dalam teks satu per satu, dari awal hingga akhir teks. Meskipun mudah diimplementasikan, metode ini memiliki kompleksitas waktu O(n*m), di mana n adalah panjang teks dan m adalah panjang pola. Kompleksitas ini menjadikannya tidak praktis untuk teks atau pola yang besar karena memerlukan waktu eksekusi yang sangat lama. Oleh karena itu, diperlukan algoritma yang lebih efisien seperti Boyer-Moore (BM) dan Knuth-Morris-Pratt (KMP) untuk mengatasi masalah ini.



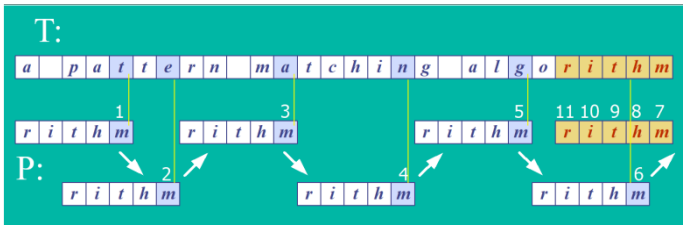
Gambar 1. Ilustrasi algoritma *Brute Force*, diambil dari [1]

B. Algoritma Boyer-Moore (BM)

Algoritma Boyer-Moore (BM) adalah salah satu algoritma string matching yang paling efisien dan banyak digunakan dalam aplikasi nyata. Algoritma ini dikembangkan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1977. Keunggulan utama dari algoritma BM adalah kemampuannya untuk melakukan pencocokan pola dengan cara yang lebih cepat dibandingkan dengan metode brute force, terutama untuk teks yang panjang.

Cara kerja algoritma BM melibatkan dua aturan utama: aturan pergeseran karakter buruk (bad character rule) dan aturan pergeseran sufiks baik (good suffix rule). Aturan pergeseran karakter buruk digunakan ketika terjadi ketidakcocokan antara pola dan teks. Algoritma akan mencari posisi karakter terakhir yang cocok di pola dan menghitung jumlah pergeseran yang diperlukan untuk melanjutkan pencocokan. Aturan pergeseran sufiks baik digunakan ketika sebagian pola cocok dengan teks. Algoritma ini akan menggeser pola sedemikian rupa sehingga bagian yang sudah cocok tetap terjaga, sehingga menghindari pencocokan ulang karakter yang sudah diketahui cocok.

Dengan menggunakan kombinasi kedua aturan ini, algoritma BM dapat secara signifikan mengurangi jumlah perbandingan yang diperlukan dalam proses pencocokan. Algoritma ini memiliki kompleksitas waktu rata-rata O(n/m), di mana n adalah panjang teks dan m adalah panjang pola. Hal ini menjadikan algoritma BM sangat efisien untuk kasus pencarian string pada teks yang panjang dan cocok untuk aplikasi seperti pencarian teks dalam dokumen besar dan pengecekan plagiarisme.



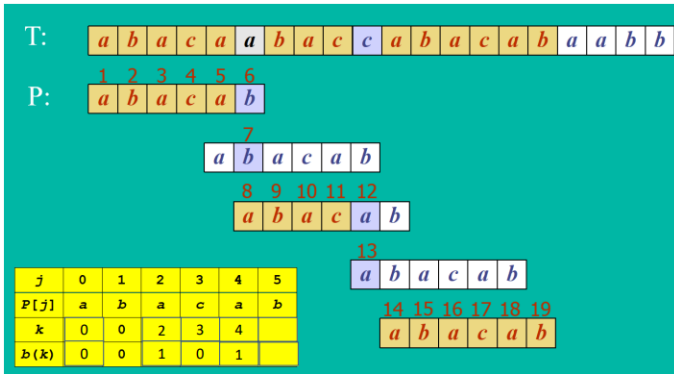
Gambar 2. Ilustrasi algoritma BM, diambil dari [1]

C. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma string matching yang dikembangkan oleh Donald Knuth, Vaughan Pratt, dan James H. Morris pada tahun 1977. Algoritma ini dirancang untuk meningkatkan efisiensi pencocokan pola dengan menggunakan informasi dari pola itu sendiri untuk menghindari pencocokan karakter yang tidak perlu.

Cara kerja algoritma KMP dimulai dengan tahap preprocessing untuk membangun tabel prefiks (juga dikenal sebagai tabel LPS - longest proper prefix which is also suffix). Tabel ini mencatat panjang prefiks terpanjang yang juga merupakan sufiks untuk setiap posisi dalam pola. Saat mencocokkan pola dengan teks, tabel prefiks digunakan untuk menentukan seberapa jauh pola harus digeser ketika terjadi ketidakcocokan. Ini memungkinkan KMP untuk melompati karakter yang sudah diketahui cocok, sehingga mengurangi jumlah perbandingan yang diperlukan dalam proses pencocokan.

Dengan menggunakan tabel prefiks, algoritma KMP memiliki kompleksitas waktu O(n + m), di mana n adalah panjang teks dan m adalah panjang pola. Hal ini menjadikan KMP sangat efisien, terutama untuk teks yang panjang atau pola yang kompleks. Algoritma KMP sangat berguna dalam aplikasi di mana pencocokan pola cepat dan akurat sangat dibutuhkan, seperti dalam pengecekan plagiarisme dan analisis teks besar.



Gambar X. Ilustrasi algortma KMP,
diambil dari [1]

III. IMPLEMENTASI

A. Program

Dalam implementasi program, digunakan algoritma BM dan KMP yang telah dijelaskan sebelumnya untuk mengecek tingkat plagiarisme dua dokumen. Program ini dibuat dengan bahasa pemrograman Python dan hanya mampu mengatasi file yang bertipe docx, pdf, dan txt.

Program berjalan dengan memasuki tahap pertama, yaitu pembacaan dan pemrosesan file dokumen. Hal ini dibagi menjadi tiga segmen berdasarkan tipe file yang telah disebutkan sebelumnya. Ketiga tipe file tersebut dipilih karena merupakan tipe file yang mudah ditemui dan/atau sering digunakan dalam masalah pengecekan plagiarisme. Dalam membaca file bertipe docx dan pdf, digunakan *library* Python *docx* dan *fitz* (dari PyMuPDF). Dokumen akan dibaca oleh program sebagai satu kesatuan *string*. *String* ini kemudian diubah kedalam bentuk *lower case* sehingga perbandingan dua dokumen mengecek plagiarisme secara *incase-sensitive*. Hasil *string* inilah yang selanjutnya akan diproses oleh tahap berikutnya.

```python
import time
import re
import os
import docx
import fitz  # PyMuPDF
from collections import Counter

def read_text_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as
file:
        return file.read()

def read_docx_file(file_path):
    doc = docx.Document(file_path)
    full_text = []
    for para in doc.paragraphs:
        full_text.append(para.text)
    return '\n'.join(full_text)

def read_pdf_file(file_path):
    pdf_document = fitz.open(file_path)
    full_text = []
    for page_num in range(len(pdf_document)):
        page = pdf_document.load_page(page_num)
        full_text.append(page.get_text())
    return '\n'.join(full_text)

def read_file(file_path):
    if file_path.endswith('.txt'):
        return read_text_file(file_path)
    elif file_path.endswith('.docx'):
        return read_docx_file(file_path)
    elif file_path.endswith('.pdf'):
        return read_pdf_file(file_path)
    else:
        raise ValueError("Unsupported file format")

def preprocess_text(text):
    return re.sub(r'\W+', ' ', text).lower()
```

Gambar 3. *Source code* program bagian pembacaan dan pemrosesan file dokumen,
diambil dari dokumen pribadi penulis

Metode yang digunakan dalam membandingkan dua dokumen ini adalah menghitung semua kemunculan kata unik yang ada pada kedua dokumen. Kemudian, dari hasil perhitungan tersebut akan dicocokkan jumlahnya antar satu dokumen dengan yang lainnya. Dalam perhitungan kemunculan kata ini, digunakan fungsi *word_frequencies* yang menerapkan algoritma KMP atau BM sesuai dengan pilihan pengguna pada *Command-Line Interface* (CLI). Fungsi ini akan melakukan pencocokan *string* dari satu kata dengan *string* yang berisi teks full dari sebuah dokumen. Fungsi ini juga menggunakan sebuah *library collections* dalam proses perhitungannya.

```
def word_frequencies(text, algorithm):
    words = text.split()
    frequencies = Counter()
    for word in set(words):
        if algorithm == 'KMP':
            matches = kmp_search(text, word)
        elif algorithm == 'BM':
            matches = bm_search(text, word)
        else:
            raise ValueError(
"Unsupported algorithm. Choose 'KMP' or 'BM'.")
        frequencies[word] = len(matches)
    return frequencies
```

Gambar 4. *Source code* program bagian perhitungan kemunculan kata,
diambil dari dokumen pribadi penulis

Algoritma KMP dan BM pada fungsi *kmp_search* dan *bm_search* akan mengembalikan suatu *array* yang berisikan index-index dimana terdapat kecocokan. Untuk lebih lengkapnya dapat lihat pada [7].

Kemudian, setelah didapatkan banyaknya kemunculan kata, akan masuk ke tahap perhitungan similaritas. Persentase similaritas ini dihitung dengan mengiterasi setiap kata unik yang ada pada kedua dokumen dan menghitung similaritasnya dengan formula umum berikut.

$$\frac{(Jumlah\ kemunculan\ kata\ terkecil\ diantara\ dua\ dokumen)}{Jumlah\ kemunculan\ kata\ terbanyak\ diantara\ dua\ dokumen}$$

```
def calculate_similarity(text1, text2, algorithm):
    freq1 = word_frequencies(text1, algorithm)
    freq2 = word_frequencies(text2, algorithm)

    all_words = set(freq1.keys()).union(set(freq2.keys
()))
    match_count = 0
    total_count = 0

    for word in all_words:
        count1 = freq1.get(word, 0)
        count2 = freq2.get(word, 0)
        match_count += min(count1, count2)
        total_count += max(count1, count2)

    if total_count == 0:
        return 100.0

    sim_percentage = (match_count / total_count) * 100
    return sim_percentage
```

Gambar 5. *Source code* program bagian perhitungan persentase similaritas,
diambil dari dokumen pribadi penulis

B. Pengujian

Dalam analisis penerapan algoritma KMP dan BM ini, akan digunakan enam buah dokumen yang terdiri dari dua dokumen bertipe txt, dua dokumen bertipe docx, dan dua dokumen bertipe pdf. Dokumen ini dapat dilihat pada Github *repository* penulis.

Pertama, pengujian plagiasi dari dua dokumen txt *dummy* yang berisikan tiga paragraf yang sama. Namun, terdapat satu dokumen yang dimodifikasi sebanyak satu kata agar berbeda sedikit dengan dokumen lainnya.

Algoritma: BM
Enter first document filename: satu.txt Enter second document filename: dua.txt Enter the algorithm to use ('KMP' or 'BM'): BM Similarity: 98.18% Execution time: 0.23766827583312988 seconds

Tabel 1. Tabel pengujian pertama dengan algoritma BM,
diambil dari dokumen peribadi penulis

Algoritma: KMP
Enter first document filename: satu.txt Enter second document filename: dua.txt Enter the algorithm to use ('KMP' or 'BM'): KMP Similarity: 98.18% Execution time: 0.17946386337280273 seconds

Tabel 2. Tabel pengujian pertama dengan algoritma KMP,
diambil dari dokumen pribadi penulis

Kedua, pengujian plagiasi dari dua dokumen bertipe docx. Dokumen pertama merupakan laporan tugas besar mata kuliah dasar pemrograman penulis dan dokumen kedua merupakan *template* tugas makalah mata kuliah Strategi Algoritma.

Algoritma: BM
Enter first document filename: word.docx Enter second document filename: word1.docx Enter the algorithm to use ('KMP' or 'BM'): BM Similarity: 8.82% Execution time: 1.5752596855163574 seconds

Tabel 3. Tabel pengujian kedua dengan algoritma BM,
diambil dari dokumen peribadi penulis

Algoritma: KMP

```
Enter first document filename: word.docx
Enter second document filename: word1.docx
Enter the algorithm to use ('KMP' or 'BM'): KMP

Similarity: 8.82%

Execution time: 2.4304094314575195 seconds
```

Tabel 4. Tabel pengujian kedua dengan algoritma KMP, diambil dari dokumen pribadi penulis

Ketiga, pengujian plagiasi dari dua dokumen bertipe pdf. Dokumen pertama merupakan laporan tugas besar dua mata kuliah Strategi Algoritma dan dokumen kedua merupakan laporan HIUPL tugas besar mata kuliah Rekayasa Perangkat Lunak.

Algoritma: BM

```
Enter first document filename: Stima.pdf
Enter second document filename: RPL.pdf
Enter the algorithm to use ('KMP' or 'BM'): BM

Similarity: 29.82%

Execution time: 5.418457746505737 seconds
```

Tabel 5. Tabel pengujian ketiga dengan algoritma BM, diambil dari dokumen peribadi penulis

Algoritma: KMP

```
Enter first document filename: Stima.pdf
Enter second document filename: RPL.pdf
Enter the algorithm to use ('KMP' or 'BM'): KMP

Similarity: 29.82%

Execution time: 9.150707721710205 seconds
```

Tabel 6. Tabel pengujian ketiga dengan algoritma KMP, diambil dari dokumen pribadi penulis

C. Analisis

Algoritma Boyer-Moore (BM) dan Knuth-Morris-Pratt (KMP) keduanya adalah algoritma pencarian/pencocokan string, namun memiliki karakteristik kinerja yang berbeda karena pendekatan pencariannya yang berbeda.

Walaupun ketiga pengujian menghasilkan persentase similaritas yang sama, kedua algoritma ini memiliki performa yang berbeda jika dilihat pada waktu eksekusinya. Pada dokumen *dummy* yang sengaja dibuat dengan kata-kata yang lebih sedikit, algoritma KMP menunjukkan waktu eksekusi yang lebih cepat. Sedangkan pada dokumen asli, seperti laporan tugas besar, algoritma BM lebih unggul dan cepat dalam hal waktu eksekusi.

Algoritma KMP melakukan pra-komputasi sebuah tabel berdasarkan pola dan menggunakan tabel ini untuk melompat ke depan dalam string input ketika terjadi ketidakcocokan. Hal ini menghasilkan kinerja terburuk pada pencarian string sebesar O(n), di mana n adalah panjang string.

Di sisi lain, algoritma BM melakukan pra-komputasi dua tabel berdasarkan pola (aturan karakter buruk dan aturan sufiks baik) dan menggunakan tabel-tabel ini untuk melompat ke depan dalam string input sejauh mungkin ketika terjadi ketidakcocokan. Kinerja algoritma BM tergantung pada sifat pola dan teks, tetapi dalam kasus terburuk, algoritma ini dapat mencapai kinerja O(nm + A), di mana n adalah panjang string, m adalah panjang pola, dan A adalah banyaknya variasi alphabet.

Secara umum, algoritma BM dianggap lebih efisien daripada algoritma KMP untuk sebagian besar teks dan pola dalam bahasa alami, karena cenderung melompati lebih banyak karakter. Namun, kinerja aktual dapat bervariasi tergantung pada teks dan pola spesifik.

Dalam kode yang digunakan, algoritma BM mungkin lebih cepat daripada algoritma KMP karena dapat melompati lebih banyak karakter ketika terjadi ketidakcocokan, sehingga menghasilkan lebih sedikit perbandingan secara keseluruhan. Hal ini terutama berlaku jika pola (kata-kata dalam hal ini) pendek dan ketidakcocokan terjadi di dekat awal pola, yang sering terjadi dalam teks bahasa alami.

IV. KESIMPULAN

Berdasarkan analisis yang telah dilakukan terhadap algoritma Boyer-Moore (BM) dan Knuth-Morris-Pratt (KMP) dalam konteks pengecekan plagiarisme, dapat disimpulkan bahwa kedua algoritma memiliki kelebihan dan kekurangan masing-masing dalam hal kinerja dan efisiensi.

Algoritma KMP menunjukkan waktu eksekusi yang lebih cepat pada dokumen dengan kata-kata yang lebih sedikit, seperti dokumen *dummy*. Hal ini disebabkan oleh kemampuan melompat yang lebih sedikit jika dibandingkan dengan algoritma BM.

Di sisi lain, algoritma BM lebih unggul dalam hal waktu eksekusi pada dokumen asli yang lebih panjang dan kompleks, seperti laporan tugas besar. BM menggunakan dua tabel pra-komputasi berdasarkan aturan karakter buruk dan aturan sufiks baik, yang memungkinkan algoritma ini untuk melompat ke depan sejauh mungkin dalam string input ketika terjadi ketidakcocokan. BM secara umum lebih efisien untuk teks dan pola dalam bahasa alami karena kemampuannya melompati lebih banyak karakter.

Dengan mempertimbangkan hasil pengujian, algoritma Boyer-Moore (BM) lebih disarankan untuk digunakan dalam aplikasi pengecekan plagiarisme, terutama untuk dokumen dengan panjang dan kompleksitas yang lebih tinggi. BM lebih efisien dalam mengurangi jumlah perbandingan yang

diperlukan selama pencarian, sehingga menghasilkan waktu eksekusi yang lebih cepat dan performa yang lebih baik secara keseluruhan.

Untuk perbaikan lebih lanjut dalam pembuatan program pengecekan plagiarisme tingkat lanjut, beberapa hal yang dapat dipertimbangkan sebagai berikut.

Integrasi Algoritma Hybrid yang merupakan penggabungan kekuatan dari kedua algoritma, KMP dan BM, untuk mendapatkan hasil yang lebih optimal. Misalnya, menggunakan KMP untuk pencarian pada dokumen pendek dan BM untuk dokumen panjang.

Optimasi Pra-Komputasi yang merupakan pengembangan metode pra-komputasi yang lebih efisien untuk tabel-tabel yang digunakan dalam kedua algoritma, sehingga dapat mengurangi waktu eksekusi awal dan meningkatkan performa pencarian.

Paralelisme yang merupakan pemanfaatan pemrosesan paralel untuk memproses bagian-bagian dokumen secara bersamaan, yang dapat mempercepat waktu pencarian dan analisis plagiarisme.

Analisis Konteks Semantik yang merupakan penambahan lapisan analisis yang lebih dalam dengan memperhatikan konteks dan semantik dari kata-kata yang digunakan, sehingga pengecekan plagiarisme tidak hanya berdasarkan pencocokan string tetapi juga makna dari teks.

Dengan perbaikan-perbaikan tersebut, program pengecekan plagiarisme dapat menjadi lebih cepat, akurat, dan efisien, membantu dalam mendeteksi plagiarisme dengan lebih efektif di berbagai jenis dokumen.

V. Ucapan Terimakasih

Makalah ini tidak akan selesai tanpa bantuan dari pihak lain. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Allah SWT,
2. orang tua penulis,
3. wali penulis,
4. Bapak dan Ibu dosen pengampu mata kuliah Strategi Algoritma,
5. seluruh mahasiswa Teknik Informatika angkatan 2022, dan 2021,
6. pihak bersangkutan lainnya yang tidak dapat disebutkan satu per satu

yang senantiasa mendukung penulis selama pembuatan makalah ini berlangsung.

Referensi

[1] R. Munir, 'IF2211 Strategi Algoritma - Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf
Diakses pada 1 Juni 2024.

[2] geeksforgeeks.org, 'Boyer Moore Algorithm for Pattern Searching', https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/
Diakses pada 1 Juni 2024.

[3] geeksforgeeks.org, 'KMP Algorithm for Pattern Searching'. https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/
Diakses pada 1 Juni 2024.

[4] kbbi.kemdikbud.go.id, 'KBBI Daring'. https://kbbi.kemdikbud.go.id/entri/plagiarisme
Diakses pada 1 Juni 2024.

[5] teknik.unpas.ac.id, 'Apa yang dimaksud dengan Plagiarisme', https://teknik.unpas.ac.id/blogs/apa-itu-plagiarisme/
Diakses pada 1 Juni 2024.

[6] aicontentfy.com, 'Exploring the Algorithm Behind Plagiarism Checkers', https://aicontentfy.com/en/blog/exploring-algorithm-behind-plagiarism-checkers
Diakses pada 1 Juni 2024.

Pranala

[1] Repository penulis, 'plagiarism_checker', https://github.com/Yusufarsan/plagiarism_checker h

[2] Pengerjaan bonus, video Instagram, https://www.instagram.com/reel/C8HzT1IyRM7/?igsh=MWY2Zm40NXNzdzI5eg==

[3] Pengerjaan bonus, video YouTube, https://youtu.be/Fkmyy0zkmBw?si=Aso0oGsHCPZUY3Ei

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, 2 Juni 2024

Yusuf Ardian Sandi, 13522015