Penggunaan String Matching dalam Pencarian Ayat Alkitab

Pendekatan Algoritma Boyer-Moore dan Knuth-Morris-Pratt

Valentino Chryslie Triadi - 13522164
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail 13522164@std.stei.itb.ac.id

***Abstrak*—Pencarian ayat dalam Alkitab merupakan salah satu hal yang dibutuhkan umat beragama katolik dan Kristen. Kemampuan untuk menemukan ayat-ayat dalam Alkitab tertentu dengan cepat berdasarkan cuplikan teks tidak hanya dapat membantu dalam meningkatkan kemudahan pencarian injil, bab, dan ayat alkitab tetapi dapat memperkaya dan pengalaman membaca alkitab. Makalah ini mengkaji penerapan algoritma Boyer-Moore dan algoritma Knuth-Morris-Pratt dalam konteks melakukan pencarian terhadap injil, bab, dan ayat alkitab yang mengandung cuplikan teks tertentu. Algoritma Boyer-Moore dan Knuth-Morris-Pratt memberikan Solusi yang efektif untuk menavigasi pencarian melalui teks-teks dalam alkitab dengan Tingkat presisi yang tinggi. Algoritma Boyer-Moore memungkinkan penarian yang lebih cepat dengan mengabaikan bagian teks yang tidak cocok, sementara algoritma Knuth-Morrid-Pratt memberikan efisiensi dalam pencarian dengan menggunakan tabel pinggiran Knuth-Morris-Pratt yangm mampu meminimalkan jumlah perbandingan karakter. Algoritma tersebut diintegrasikan dengan perangkat lunak berbasis website, sehingga dapat membantu pengguna dalam melakukan pencarian ayat yang diinginkan dengan lebih cepat dan akurat. Penelitian ini bertujuan untuk melakukan evaluasi performa algoritma dan juga untuk berkontribusi dalam membantu pengguna menemukan ayat Alkitab yang diinginkan. Sehingga perangkat lunak pencarian ayat Alkitab dapat dikembangkan lebih lanjut di kemudian hari.**

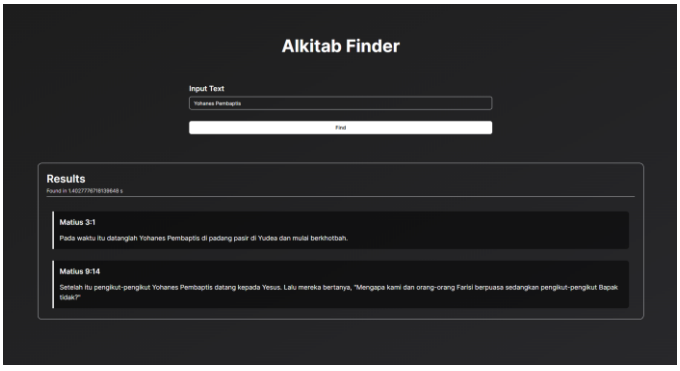
Keywords—Alkitab; cuplikan; pencocokan string; algoritma Boyer-Moore; algoritma Knuth-Morris-Pratt; pencarian; efisiensi pencarian; kontribusi; perangkat lunak berbasis website.

I. PENDAHULUAN

Saat membaca Alkitab, terkadang penulis dibingungkan dengan ayat mana yang harus dibaca dan yang ingin dibaca oleh penulis adalah bagian Alkitab tertentu yang mengandung beberapa cuplikan teks yang diingat. Oleh karena itu, dibutuhkan alat untuk melakukan pencarian ayat alkitab berdasarkan cuplikan teks sehingga dapat membantu memudahkan dalam menemukan ayat alkitab yang diinginkan. Karena jumlah ayat dalam Alkitab yang tidak sedikit, maka diperlukan algoritma yang cukup efisien dan cepat dalam melakukan pencarian, seperti algoritma Boyer-Moore dan Knuth-Morris-Pratt. Algoritma Boyer-Moore dan Knuth-Morris-Pratt mampu memberikan solusi pencocokan string yang efisien dan cepat.

Algoritma Boyer-Moore adalah salah satu metode yang efisien dalam melakukan pencocokan string. Algoritma ini memiliki teknik *looking-glass* dan Teknik *character-jump* yang memungkinkan algoritma ini untuk melakukan pencocokan string dengan jumlah perbandingan karakter yang minimum. Dalam konteks pencarian ayat dalam Alkitab, algoritma ini bisa digunakan untuk melakukan pencocokan string antara seluruh teks yang terdapat di Alkitab dengan pola yang berasal dari masukan cuplikan teks.

Algoritma Knuth-Morris-Pratt adalah salah satu metode yang efisien dalam melakukan pencocokan string. Algoritma ini memiliki fungsi pinggiran Knuth-Morris-Pratt yang memetakan prefiks dan suffiks dalam sebuah pola. Dengan adanya fungsi pinggiran Knuth-Morris-Pratt tersebut, algoritma ini memungkinkan pencocokan string yang efisien dengan melakukan lompatan karakter berdasarkan fungsi pinggiran Knuth-Morris-Pratt. Dalam konteks pencarian ayat dalam Alkitab, algoritma ini bisa digunakan untuk melakukan pencocokan string antara seluruh teks yang terdapat di Alkitab dengan pola yang berasal dari masukan cuplikan teks.



Gambar 1. Ilustrasi Pencarian Ayat Alkitab Berdasarkan Cuplikan Teks

Sumber: Dokumentasi Penulis

Makalah ini bertujuan untuk mengaplikasikan algoritma Boyer-Moore dan algoritma Knuth-Morris-Pratt. Penulis akan melakukan penerapan algoritma Boyer-Moore dan algoritma Knuth-Morris-Pratt dalam membuat perangkat lunak berbasis website untuk memudahkan pencarian ayat dalam Alkitab berdasarkan cuplikan teks tertentu. Penerapan kedua algoritma tersebut melibatkan pemahaman dan analisis terkait struktur atau susunan ayat dalam Alkitab. Sehingga diperlukan data terkait keseluruhan isi dari Alkitab.

Selain mengaplikasikan algoritma Boyer-Moore dan algoritma Knuth-Morris-Pratt, penulis juga akan melakukan analisis dan evaluasi terhadap kinerja dari kedua algoritma tersebut. Analisis dan evaluasi kinerja kedua algoritma tersebut dapat dilakukan dengan mencari waktu eksekusi tersingkat diantara kedua algoritma tersebut. Kemudian penulis hanya akan menggunakan algoritma dengan efisiensi terbaik untuk diterapkan di dalam perangkat lunak berbasis website.

Penelitian ini tidak hanya menawarkan solusi algoritma terbaik untuk melakukan pencarian ayat dalam Alkitab berdasarkan cuplikan teks tertentu saja, melainkan penelitian ini juga bertujuan untuk menciptakan perangkat lunak berbasis website yang dapat digunakan oleh publik dalam melakukan pencarian ayat dalam Alkitab yang diinginkan dengan harapan publik khususnya yang beragama Katolik dan Kristen dapat terbantu dalam menentukan dan membaca ayat Alkitab.

II. Dasar Teori

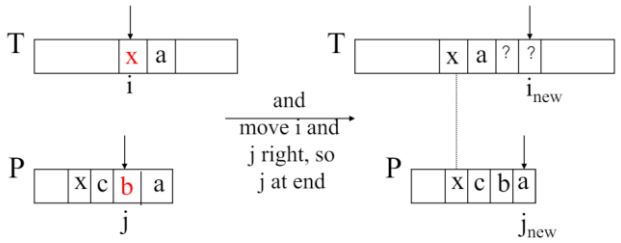
A. *Pencocokan String*

Pencocokan string adalah salah satu algoritma yang bisa digunakan dalam berbagai aspek, seperti dalam hal pencarian teks, *cybersecurity*, autentikasi, identifikasi, dan masih banyak hal lainnya. Algoritma pencocokan string mampu memberikan informasi mengenai apakah sebuah string mengandung pola tertentu. Dengan mengetahui hal tersebut, kita dapat mengetahui bahwa sebuah string berisi pola yang terletak pada karakter ke-sekian. Algoritma ini sangat membantu dalam berbagai aspek, terutama dalam hal pencarian teks.

B. *Algoritma Boyer-Moore*

Algoritma Boyer-Moore merupakan salah satu algoritma pencocokan string yang cukup efisien. Algoritma ini menggunakan dua teknik utama yaitu teknik *looking-glass* dan teknik *character-jump*. Teknik *looking-glass* adalah sebuah teknik membaca pattern yang akan dicari dari akhir ke awal. Selanjutnya, teknik *character-jump* adalah teknik yang dijalankan ketika ditemukan *mismatch* atau ketidak-sesuaian pada saat pencocokan. Terdapat 3 kemungkinan kasus dalam teknik *character-jump*, diantaranya:

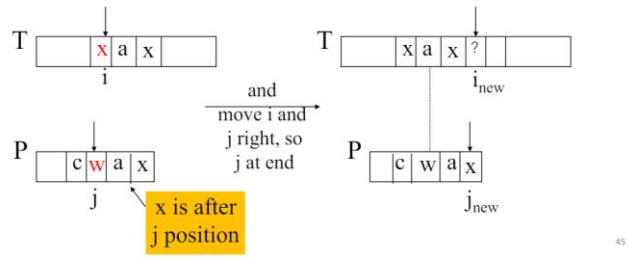
1. Kasus 1, kasus ini terjadi ketika pola P mengandung karakter *mismatch* ‘x’ dimana karakter ‘x’ ditemukan pada index yang berada kurang dari index karakter *mismatch* pada pola P. Dengan kata lain, *last occurrence* dari karakter ‘x’ bernilai lebih kecil daripada index *mismatch* pada pola P. Pada kasus ini, pola P akan digeser sehingga karakter ‘x’ pada pola P bersesuaian dengan karakter ‘x’ (*mismatch*) pada string T. Kemudian, nilai index pencocokan pada pola P (j) akan diubah hingga menunjuk ke karakter akhir pada pola P dan nilai index pencocokan pada string T (i) akan digeser hingga menunjuk ke karakter yang sejajar dengan karakter akhir di string T. Sebagai ilustrasi dapat dilihat pada gambar berikut:



Gambar 2. Ilustrasi Kasus 1 pada Teknik Character-Jump Boyer-Moore

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

2. Kasus 2, kasus ini terjadi ketika pola P mengandung karakter *mismatch* ‘x’ dimana karakter ‘x’ ditemukan pada index yang berada lebih dari index karakter *mismatch* pada pola P. Dengan kata lain, *last occurrence* dari karakter ‘x’ bernilai lebih besar daripada index *mismatch* pada pola P. Pada kasus ini, pola P akan digeser sebanyak satu karakter ke kanan. Kemudian, nilai index pencocokan pada pola P (j) akan diubah hingga menunjuk ke karakter akhir pada pola P dan nilai index pencocokan pada string T (i) akan digeser hingga menunjuk ke karakter yang sejajar dengan karakter akhir di string T. Sebagai ilustrasi dapat dilihat pada gambar berikut:

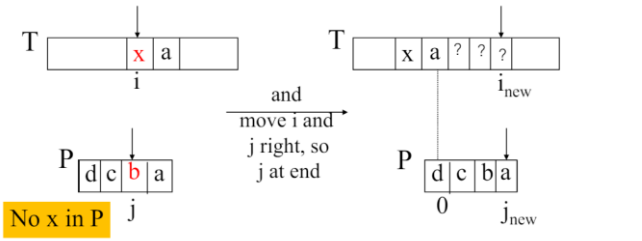


Gambar 3. Ilustrasi Kasus 2 pada Teknik Character-Jump Boyer-Moore

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

3. Kasus 3, kasus ini terjadi ketika pola P tidak mengandung karakter *mismatch* ‘x’. Pada kasus ini, pola P akan digeser sehingga karakter awal dari pola P bersesuaian dengan karakter setelah karakter ‘mismatch’ pada string T. Kemudian, nilai index pencocokan pada pola P (j) akan diubah hingga

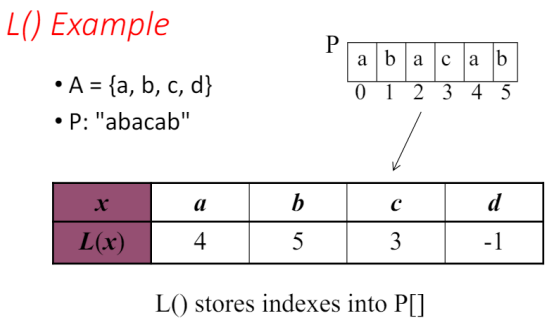
menunjuk ke karakter akhir pada pola P dan nilai index pencocokan pada string T (i) akan digeser hingga menunjuk ke karakter yang sejajar dengan karakter akhir di string T. Sebagai ilustrasi dapat dilihat pada gambar berikut:



Gambar 4. Ilustrasi Kasus 3 pada Teknik Character-Jump Boyer-Moore

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

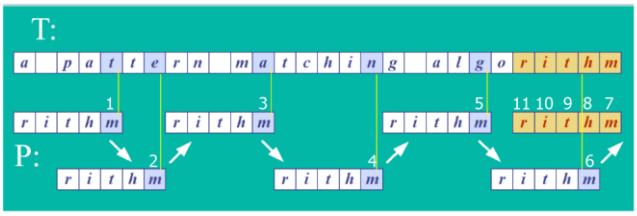
Dalam melakukan pencocokan string dengan algoritma Boyer-Moore, diperlukan bantuan tabel *Last Occurrence* yang berisi informasi index terakhir dari masing-masing karakter di pola. Berikut adalah contoh dari tabel *Last Occurrence*:



Gambar 5. Ilustrasi Tabel Last Occurrence Boyer-Moore

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

Algoritma ini selalu melakukan pencocokan dari bagian belakang pola (sesuai dengan teknik *looking-glass*) dan apabila terjadi *mismatch* maka akan dilakukan penanganan *mismatch* sesuai dengan yang harus dilakukan pada kasus 1, kasus 2 atau kasus 3. Berikut adalah ilustrasi cara kerja algoritma Boyer-Moore:

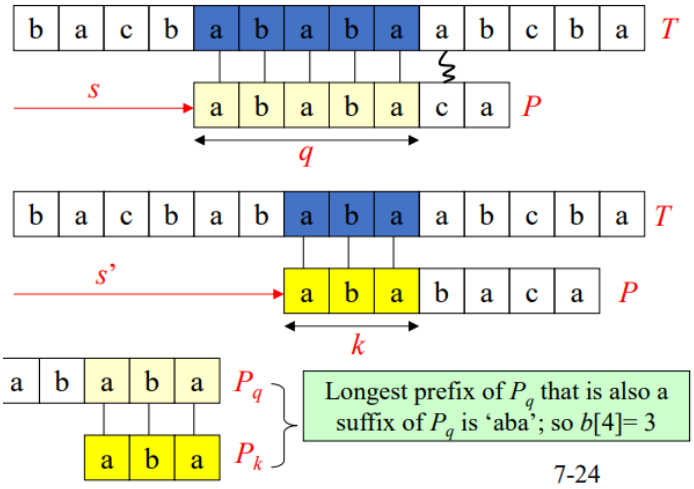


Gambar 6. Ilustrasi Alur Kerja Algoritma Boyer-Moore

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

C. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt merupakan salah satu algoritma pencocokan string yang cukup efisien. Algoritma ini akan melakukan pencocokan string dari kiri ke kanan seperti pada umumnya, namun yang membedakan adalah algoritma ini mampu melakukan *shifting* atau pergeseran pencocokan yang lebih efisien dibanding algoritma *bruteforce*. Pergeseran atau *shifting* yang dilakukan oleh algoritma ini bergantung pada kesamaan suffiks dan prefiks dari sebuah string yang kita cocokan. Pola yang dicocokan akan digeser agar prefiks dari sebuah pola bersesuaian dengan string yang sedang dicocokan. Setelah dilakukan pergeseran, algoritma ini akan melanjutkan pencocokan huruf atau karakter setelah prefix atau suffix yang telah digeser tersebut. Sebagai contoh dapat dilihat pada ilustrasi berikut:



Gambar 7. Ilustrasi algoritma Knuth-Morris-Pratt

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

Ilustrasi diatas menggambarkan jika karakter di string P tidak sesuai dengan karakter di string T pada posisi yang sama, maka algoritma ini akan melakukan *shifting* atau pergeseran dengan memperhatikan prefiks dan suffiks terbesar dari string

P yang sudah bersesuaian dengan string T. Setelah pergeseran untuk menyesuaikan suffix dengan prefix dijalankan, maka algoritma akan memulai pengecekan langsung dari karakter setelah prefix tersebut.

Untuk melakukan pergeseran atau *shifting* agar prefiks dan suffiks terbesar bersesuaian, dapat menggunakan Knuth-Morris-Pratt *Border Function* atau fungsi pinggiran Knuth-Morris-Pratt. KMP Border Function adalah salah satu cara untuk mencari prefiks dan suffiks terbesar yang bersesuaian. Untuk lebih jelasnya dapat dilihat ilustrasi berikut:

(k = j-1)

j	0	1	2	3	4	5	6	7	8	9
P [j]	a	b	a	b	a	b	a	b	c	a
k	0	1	2	3	4	5	6	7	8	
b[k]	0	0	1	2	3	4	5	6	0	

Gambar 8. Ilustrasi Fungsi Pinggiran Knuth-Morris-Pratt

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

Pada ilustrasi di atas, langkah pertama yang harus dilakukan adalah menentukan nilai j, P[j], dan k dengan j adalah index dari string, P[j] adalah list of karakter, dan k adalah j-1. Kemudian tentukan b[k] yang berarti panjang prefix dan suffix terbesar untuk string yang dimulai dari index 0 hingga index k dengan ketentuan nilai maksimal b[k] adalah (k-1).

D. Alkitab

Alkitab merupakan kitab suci agama Katolik dan Kristen. Pada makalah kali ini, Alkitab yang akan penulis jadikan acuan adalah Perjanjian Baru agama Katolik dan penulis hanya akan membahas sedikit mengenai struktur atau susunan isi dari Perjanjian Baru agama Katolik. Struktur atau susunan isi dari Perjanjian Baru agama Katolik terdiri atas 27 Kitab, yakni 'Matius', 'Markus', 'Lukas', 'Yohanes', 'Kisah Para Rasul', 'Roma', 'Korintus I', 'Korintus II', 'Galatia', 'Efesus', 'Filipi', 'Kolose', 'Tesalonika I', 'Tesalonika II', 'Timotius I', 'Timotius II', 'Titus', 'Filemon', 'Ibrani', 'Yakobus', 'Petrus I', 'Petrus II', 'Yohanes I', 'Yohanes II', 'Yohanes III', 'Yudas', dan 'Wahyu'. Masing-masing kitab tersebut, tersusun atas beberapa bab dan untuk setiap bab tersusun atas beberapa ayat beserta isi teksnya.

III. Implementasi dan Pengujian

A. Data Scraping

Dalam mendapatkan data isi Alkitab Perjanjian Baru agama Katolik, penulis melakukan pengambilan data melalui website https://alkitabonline.org/. Data yang diambil dari website tersebut kemudian disimpan dalam struktur data yang berbentuk *tree*. Tujuan dari pemilihan struktur data *tree* adalah agar data yang tersimpan dapat dikategorikan berdasarkan Kitab, bab, dan ayat. Sehingga dalam proses pengolahan hasil pencarian string dapat lebih mudah dan terstruktur. Berikut adalah ilustrasi struktur data untuk menyimpan data Alkitab Perjanjian Baru agama Katolik:

Gambar 9. Struktur Data Penyimpanan Perjanjian Baru

Sumber: Dokumentasi Penulis

Berikut adalah kode program untuk melakukan *scraping* data isi Perjanjian Baru:

```python
import json
import requests
from bs4 import BeautifulSoup


# Chapter List
listChapter = [] # Berisi list chapter yang akan di scrape

# Write to JSON File
data = {}

for chap in listChapter:
    Baseurl = f"https://alkitabonline.org/INDSHVN2ET-{chap}-1.html"

    response = requests.get(Baseurl)
    soup = BeautifulSoup(response.content, "html.parser")
    listSection = soup.find("div", class_="section-box").find_all("a")

    for u in listSection:

        # Define the URL of the website you want to scrape
        url = u.get("href")

        # Send a GET request to the website
        response = requests.get(url)

        # Parse the HTML content of the website using BeautifulSoup
        soup = BeautifulSoup(response.content, "html.parser")

        # Find and extract the desired data from the HTML
        chapter = soup.find("div", class_="chapter").find("div", class_="title").text
        section = soup.find("div", class_="section").find("div", class_="title").text
        content = {} # sup : text

        contains = soup.find("div", class_="the-post-cont")
        for p in contains:
            # print("p: ", p.text)
            sup = (p.find("sup"))
            if sup != -1:
                ayat = sup.text
                text = p.text.removeprefix(ayat)\
                    .replace("\u201c", "'")\
                    .replace("\u201d", "'")\
                    .replace("\u2018", "'")\
                    .replace("\u2019", "'")\
                    .replace("\u2013", "-")\
                    .replace("\u2014", "-")\
                    .replace("\u2026", "...")\
                    .replace("\u00a0", " ")\
                    .strip()
                if (text != "-" and text != ""):
                    content[ayat] = text
        try:
            data[chapter][section] = content
        except:
            data[chapter] = {}
            data[chapter][section] = content

        print(f"Scraped {chapter} {section}...")


with open("data.json", "w") as file:
    json.dump(data, file, indent=4)
```

Gambar 10. Kode Program Scraping

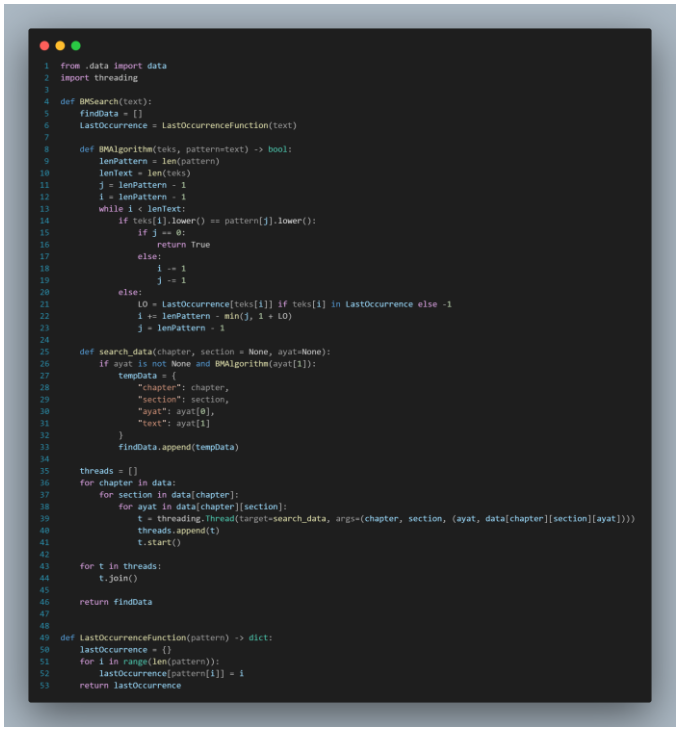
B. Implementasi Algoritma Boyer-Moore dan Algoritma Knuth-Morris-Pratt

Pada algoritma Boyer-Moore dan Knuth-Morris-Pratt, penulis menggunakan masukan dari pengguna sebagai pola, dan membandingkannya dengan keseluruhan teks yang terdapat di basis data. Karena jumlah teks dalam basis data yang sangat banyak, maka penulis melakukan optimisasi dengan menggunakan *multi-threading* dimana untuk setiap ayat pada basis data akan diproses dengan satu buah *thread*. Hal ini dapat meningkatkan efisiensi dari pencarian dan pencocokan string.

```python
from .data import data
import threading

def KMPSearch(text):
    findData = []
    KMPBorder = KMPBorderFunction(text)

    def KMPAlgorithm(teks, pattern=text) -> bool:
        j = 0
        i = 0
        lenPattern = len(pattern)
        lenText = len(teks)
        while i < lenText:
            if teks[i].lower() == pattern[j].lower():
                i += 1
                j += 1
                if j == lenPattern:
                    return True
            else:
                if j != 0:
                    j = KMPBorder[j - 1]
                else:
                    i += 1
        return False

    def search_data(chapter, section, ayat=None):
        if ayat is not None and KMPAlgorithm(ayat[1]):
            tempData = {
                "chapter": chapter,
                "section": section,
                "ayat": ayat[0],
                "text": ayat[1]
            }
            findData.append(tempData)

    threads = []
    for chapter in data:
        for section in data[chapter]:
            for ayat in data[chapter][section]:
                t = threading.Thread(target=search_data, args=(chapter, section, (ayat, data[chapter][section][ayat])))
                threads.append(t)
                t.start()

    for t in threads:
        t.join()

    return findData


def KMPBorderFunction(text) -> list[int]:
    KMPBorder = [0] * (len(text) - 1)
    for i in range(len(text) - 1):
        for j in range(i - 1, 0, -1):
            if text[:j] == text[i - j:i]:
                KMPBorder[i] = j
                break
    return KMPBorder
```

Gambar 11. Kode Program Algoritma KMP

Sumber: Dokumentasi Penulis



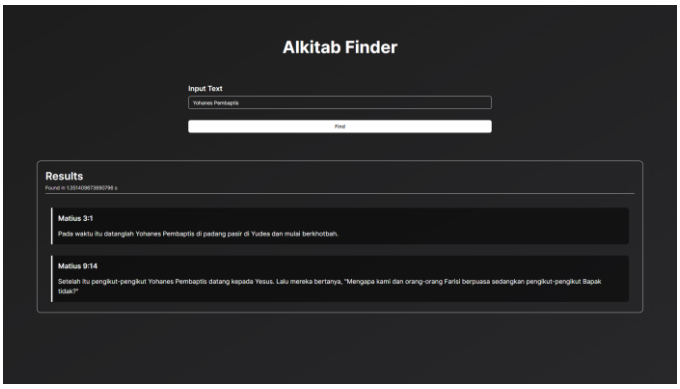
```python
from .data import data
import threading

def BMSearch(text):
    findData = []
    LastOccurrence = LastOccurrenceFunction(text)

    def BMAlgorithm(teks, pattern=text) -> bool:
        lenPattern = len(pattern)
        lenText = len(teks)
        j = lenPattern - 1
        i = lenPattern - 1
        while i < lenText:
            if teks[i].lower() == pattern[j].lower():
                if j == 0:
                    return True
                else:
                    i -= 1
                    j -= 1
            else:
                LO = LastOccurrence[teks[i]] if teks[i] in LastOccurrence else -1
                i += lenPattern - min(j, 1 + LO)
                j = lenPattern - 1

    def search_data(chapter, section = None, ayat=None):
        if ayat is not None and BMAlgorithm(ayat[1]):
            tempData = {
                "chapter": chapter,
                "section": section,
                "ayat": ayat[0],
                "text": ayat[1]
            }
            findData.append(tempData)

    threads = []
    for chapter in data:
        for section in data[chapter]:
            for ayat in data[chapter][section]:
                t = threading.Thread(target=search_data, args=(chapter, section, (ayat, data[chapter][section][ayat])))
                threads.append(t)
                t.start()

    for t in threads:
        t.join()

    return findData


def LastOccurrenceFunction(pattern) -> dict:
    lastOccurrence = {}
    for i in range(len(pattern)):
        lastOccurrence[pattern[i]] = i
    return lastOccurrence
```

Gambar 12. Kode Program Algoritma BM

Sumber: Dokumentasi Penulis

C. Pembuatan Perangkat Lunak Berbasis Website

Perangkat lunak berbasis website dibuat oleh penulis pada kerangka kerja Next.js dengan bahasa pemrograman *Typescript*. Hal ini memungkinkan untuk membuat perangkat lunak berbasis website yang mudah untuk diakses dan menarik. Dalam menghubungkan atau mengintegrasikan implementasi dari algoritma pencarian string, penulis menggunakan bantuan API. Sehingga, memungkinkan untuk antar muka perangkat lunak dan implementasi algoritma pencarian dibuat secara terpisah. Berikut adalah beberapa dokumentasi perangkat lunak berbasis website yang dibuat oleh penulis:



Gambar 13. Antar Muka Perangkat Lunak

Sumber: Dokumentasi Penulis

Gambar 14. Respon API Pencarian

Sumber: Dokumentasi Penulis

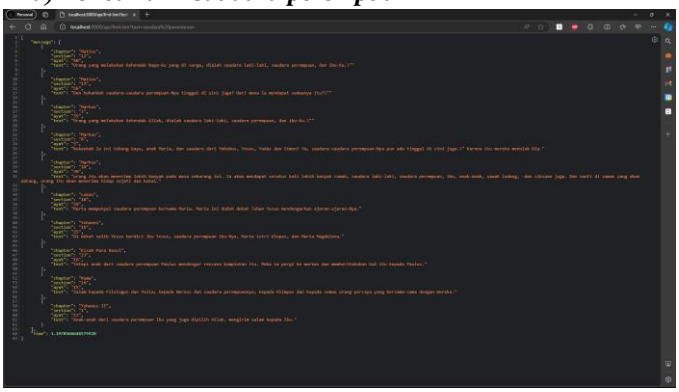
TABLE I. TABEL KONTRAK API

Method	URL	Keterangan
GET	/api/find-bm/?text={text}	API untuk melakukan pencarian text dengan algoritma Boyer-Moore
GET	/api/find-kmp/?text={text}	API untuk melakukan pencarian text dengan algoritma Knuth-Morris-Pratt

D. *Hasil Pengujian*

1) *Pengujian dengan Algoritma Boyer-Moore*

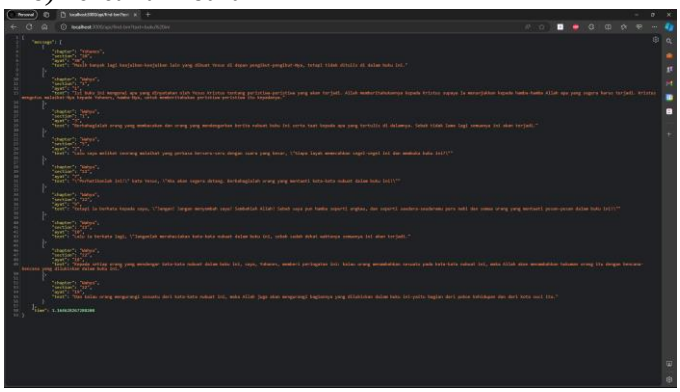
*a) **Pencarian "saudara perempuan"***



Gambar 15. Hasil Pengujian BM "saudara perempuan"

Sumber: Dokumentasi Penulis

b) Pencarian "buku ini"

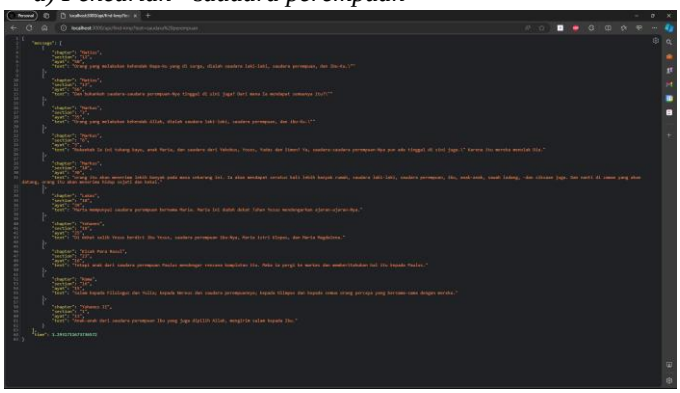


Gambar 16. Hasil Pengujian BM "buku ini"

Sumber: Dokumentasi Penulis

2) *Pengujian dengan Algoritma Knuth-Morris-Pratt*

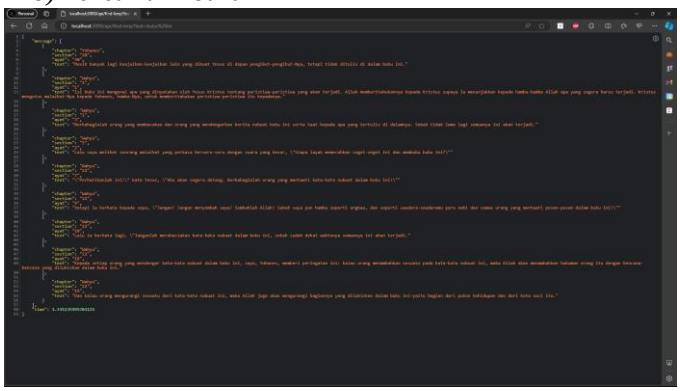
a) Pencarian "saudara perempuan"



Gambar 17. Hasil Pengujian KMP "saudara perempuan"

Sumber: Dokumentasi Penulis

b) Pencarian "buku ini"



Gambar 18. Hasil Pengujian KMP "buku ini"

Sumber: Dokumentasi Penulis

IV. Analisis dan Kesimpulan

A. *Analisis Hasil Uji*

Setelah melakukan pengujian terhadap algoritma Boyer-Moore dan algoritma Knuth-Morris-Pratt. Didapatkan tabel hasil sebagai berikut:

TABLE II. Tabel Hasil Pengujian

Cuplikan teks yang dicari	Waktu Eksekusi	
	Boyer-Moore	Knuth-Morris-Pratt
saudara perempuan	1.198 detik	1.293 detik
buku ini	1.165 detik	1.335 detik

Dari tabel hasil uji tersebut, dapat dilihat bahwa algoritma Boyer-Moore mampu menemukan hasil dalam waktu yang lebih singkat dibandingkan dengan algoritma Knuth-Morris-Pratt. Hal ini dapat didekati dengan cara kerja kedua algoritma tersebut, dimana algoritma Boyer-Moore akan cenderung lebih efektif apabila banyak karakter pada string yang tidak ada pada pola dan algoritma Knuth-Morris-Pratt cenderung lebih efektif apabila pola tidak memiliki prefiks dan suffiks yang besar saat dilakukan pencarian fungsi pinggiran Knuth-Morris-Pratt.

B. *Kesimpulan*

Dari penelitian ini, penulis dapat menyimpulkan bahwa dengan bantuan kode program dan algoritma pencocokan string, untuk mencari kitab, bab, dan ayat dari sebuah cuplikan teks bukanlah hal yang sulit lagi. Pengguna tidak perlu melakukan pencarian satu-persatu dengan menebak kitab, bab, dan ayat yang mengandung cuplikan teks, sehingga akan dipastikan pencarian akan lebih cepat, mudah, dan akurat.

Penggunaan algoritma Boyer-Moore dalam kasus pencarian ayat Alkitab yang mengandung cuplikan teks memiliki Tingkat efisiensi yang lebih tinggi dibandingkan dengan algoritma Knuth-Morris-Pratt. Sehingga dalam pengembangan perangkat lunak berbasis website, penulis memilih untuk menerapkan algoritma Boyer-Moore sebagai algoritma pencarian teks utama.

Link Video Youtube

https://youtu.be/yWKHIRRQ6bg

Link Kode Program

Link Repository: https://github.com/ValentinoTriadi/Alkitab-Finder

Link Projek: https://alkitab-finder.vercel.app/

Ucapan Terima Kasih

Puji Syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah memberi berkat rahmat dan kasih karunia-Nya. Sehingga, penulis dapat menyelesaikan makalah yang berjudul "Penggunaan String Matching dalam Pencarian Ayat Alkitab" dengan baik dan selesai tepat waktu. Tak lupa juga penulis mengucapkan terima kasih kepada keluarga yang telah memberikan dukungan sehingga penulis dapat menyelesaikan Makalah IF2211 Strategi Algoritma – Sem. II Tahun 2023/2024 ini. Terima kasih juga penulis sampaikan kepada dosen-dosen pengampuh mata kuliah IF2120 terutama kepada Ir. Rila Mandala, M.Eng., Ph.D. dan Monterico Adrian, S.T., M.T. selaku dosen pengampuh mata kuliah IF2211 untuk kelas 03 karena telah membimbing penulis dan memberikan sumber belajar untuk bisa memahami keilmuan dalam makalah ini. Penulis berharap bahwa makalah ini nantinya dapat digunakan sebagai referensi baik bagi para pelajar yang penasaran dengan keilmuan terkait atau bahkan sebagai referensi penulis kedepannya.

References

[1] Munir, Rinaldi, "Pencocokan String (String/Pattern Matching) Bahan Kuliah IF2211 Strategi Algoritma". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf (diakses pada 11 Juni 2024).

[2] Fazira, Mita, "PERBANDINGAN ALGORITMA KNUTH-MORRIS-PRATT DAN BOYER MOORE DENGAN METODE PERBANDINGAN EKSPONENSIAL PADA APLIKASI KAMUS BAHASA INDONESIA – JERMAN BERBASIS ANDROID". https://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/download/1418/1145 (diakses pada 11 Juni 2024).

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, 12 Juni 2024

Valentino Chryslie Triadi 13522164