Laporan Tugas Kelompok

Final Project Implementasi Algoritma *Boyer-Moore Horspool* pada Aplikasi Pencatatan Deadline Tugas Berbasis *Graphical User Interface*

Dibuat untuk memenuhi tugas akhir mata kuliah Desain dan Analisis Algoritma

Dosen Pengampu:

I Made Widiartha, S.Si., M.Kom

Disusun oleh:

Kelompok I

I Gede Widnyana	**(2208561016)**
Ni Made Viona Rara Santhi	**(2208561098)**

Kelas: C

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

JIMBARAN

2023

DAFTAR ISI

DAFTAR ISI i
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 1
1.3 Tujuan 2
BAB II Kajian Teori 3
2.1 GUI Tkinter 3
2.2 Algoritma *Boyer-Moore Horspool* 4
2.2.1 Definisi 4
2.2.2 Cara Kerja 5
2.2.2 Algoritma Pseudocode 6
2.2.3 Algoritma Flowchart dari *Boyer Moore Search* 8
2.2.3 Ilustrasi Cara Kerja Algoritma 12
2.2.5 Analisis Kompleksitas Waktu Algoritma pada Program 15
BAB III Implementasi Algoritma 19
3.1 Flowchart Aplikasi 19
3.2 Source Code 21
3.3 Penjelasan Source Code 30
3.4 Dokumentasi Video Running Program 48
BAB IV Kesimpulan 49
DAFTAR PUSTAKA 50
LAMPIRAN 51

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aplikasi Pencatatan Deadline Tugas Mahasiswa adalah solusi yang sangat relevan mengingat tingginya beban kerja dan tugas yang harus dihadapi oleh mahasiswa dalam menjalani kehidupan perkuliahan. Dalam situasi ini, perencanaan yang baik terhadap waktu dan deadline tugas menjadi krusial untuk mencapai keberhasilan akademis. Keterlambatan dalam menyelesaikan tugas dapat berdampak negatif pada kinerja akademis mahasiswa. Oleh karena itu, aplikasi pencatatan deadline tugas menjadi alat yang sangat dibutuhkan untuk membantu mahasiswa mengorganisir dan mengelola jadwal tugas mereka.

Algoritma *Boyer-Moore Horspool*, yang digunakan dalam aplikasi ini untuk melakukan pencarian, dipilih dengan pertimbangan efisiensi dan kecepatan dalam menemukan informasi. Dengan adanya fitur pencarian ini, mahasiswa dapat dengan cepat menemukan tugas-tugas tertentu dalam daftar mereka, memudahkan pengelolaan waktu dan memastikan bahwa tidak ada deadline yang terlewat. Algoritma *Boyer-Moore Horspool* secara khusus dikenal efektif untuk pencarian pola dalam teks, sehingga sangat sesuai untuk kebutuhan pencarian keterangan tugas dalam aplikasi ini.

Pemilihan algoritma ini tidak hanya didasarkan pada kecepatan, tetapi juga pada kemampuannya dalam menangani pencarian yang melibatkan lebih dari satu kata kunci. Dalam konteks aplikasi ini, mahasiswa dapat mencari tugas berdasarkan beberapa kata kunci sehingga pencarian menjadi lebih fleksibel dan sesuai dengan kebutuhan pengguna. Dengan demikian, aplikasi Pencatatan Deadline Tugas Mahasiswa dengan algoritma *Boyer-Moore Horspool* memberikan solusi yang efektif dan efisien dalam mengatasi tantangan manajemen waktu dan tugas di lingkungan perkuliahan.

1.2 Rumusan Masalah

Adapun dua rumusan masalah yang dibahas pada laporan ini adalah sebagai berikut.

1. Bagaimana cara kerja algoritma *Boyer Moore search* ?
2. Bagaimana implementasi algoritma *Boyer Moore search* dalam aplikasi pencatatan deadline tugas mahasiswa?

1.3 Tujuan

Adapun tujuan yang ditetapkan penulis berdasarkan rumusan masalah adalah sebagai berikut.

1. Menjelaskan dan mengeksplorasi cara kerja algoritma *Boyer Moore search.*
2. Menerapkan algoritma *Boyer Moore search* dalam aplikasi pencatatan deadline tugas mahasiswa.

BAB II
Kajian Teori

2.1 GUI Tkinter

Graphical User Interface (GUI) merujuk pada antarmuka pengguna yang memanfaatkan elemen grafis seperti ikon, tombol, jendela, dan menu untuk memfasilitasi interaksi antara pengguna dan komputer. Dengan GUI, pengguna dapat berinteraksi dengan perangkat lunak atau sistem tanpa harus mengingat perintah-perintah teks atau sintaks tertentu. GUI memberikan pengalaman pengguna yang lebih intuitif dan mudah dipahami. Elemen utama dari GUI melibatkan penggunaan elemen visual seperti jendela, tombol, teks, gambar, dan kontrol lainnya untuk menyederhanakan tugas dan memungkinkan pengguna berinteraksi dengan perangkat lunak atau sistem secara visual.

Tkinter adalah modul bawaan (built-in module) dalam Python yang menyediakan alat untuk membuat aplikasi dengan antarmuka grafis. Tkinter adalah antarmuka untuk toolkit GUI Tk, yang merupakan toolkit GUI standar untuk Tcl (Tool Command Language), sebuah bahasa skrip yang sering digunakan untuk pengembangan aplikasi dan skrip di dunia perangkat lunak.

Beberapa konsep kunci dalam Tkinter termasuk:

1. **Jendela Utama (Main Window):** Tkinter menggunakan jendela utama sebagai basis untuk menampilkan elemen-elemen GUI lainnya.
2. **Widget:** Widget adalah elemen-elemen dasar GUI seperti tombol, label, entri teks, dan lainnya. Tkinter menyediakan berbagai widget yang dapat digunakan untuk membangun antarmuka pengguna.
3. **Event Handling:** Tkinter mendukung pemrosesan peristiwa (events) yang terjadi saat pengguna berinteraksi dengan GUI, seperti mengklik tombol atau menggeser mouse. Event handling digunakan untuk menanggapi interaksi pengguna.
4. **Layout Management:** Tkinter menyediakan manajemen tata letak (layout management) untuk mengatur posisi dan tata letak widget di dalam jendela.

5. **Callback Functions:** Callback functions adalah fungsi-fungsi yang dipanggil saat suatu peristiwa terjadi. Misalnya, saat pengguna mengklik tombol, fungsi tertentu dapat dijalankan.

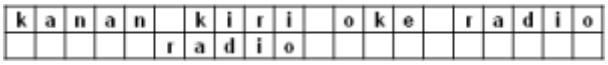
Tkinter adalah pilihan umum untuk pengembangan aplikasi GUI dengan Python karena kemudahannya dalam digunakan dan keberadaannya yang sudah terintegrasi dengan instalasi Python standar. Dengan Tkinter, pengembang dapat membuat aplikasi dengan antarmuka pengguna yang menarik dan mudah digunakan.

2.2 Algoritma *Boyer-Moore Horspool*

2.2.1 Definisi

Boyer-Moore secara rata-rata merupakan algoritma pencarian string yang paling baik jika dibandingkan dengan algoritma pencarian string lainnya seperti Brute-Force ataupun Knuth-Morris-Pratt (Utomo, 2020). Jika kita menggunakan fasilitas Find/Search pada berbagai aplikasi pengolah teks, web browser, dan aplikasi lainnya mungkin saja kita telah memanfaatkan algoritma *Boyer-Moore Horspool* dalam pencarian tersebut, karena algoritma ini paling banyak diimplementasikan dalam berbagai aplikasi untuk fasilitas pencarian teksnya. Algoritma *Boyer-Moore Horspool* adalah salah satu algoritma untuk mencari suatu string di dalam teks, dibuat oleh R.M Boyer dan J.S Moore. Ide utama algoritma ini adalah mencari string dengan melakukan pembandingan karakter mulai dari karakter paling kanan dari string yang dicari (Utomo, 2020). Dengan mengunakan algoritma ini, secara rata-rata proses pencarian akan menjadi lebih cepat jika dibandingakan dengan algoritma lainnya. alasan melakukan pencocokan dari kanan (posisi terakhir string yang dicari) ditunjukan dalam contoh berikut :

Pada contoh diatas, dengan melakukan pembandingan dari posisi paling akhir string dapat dilihat bahwa karakter “n” pada string “kanan” tidak cocok dengan karakter “o” pada string “radio” yang dicari, dan karakter “n” tidak pernah ada dalam string “radio” yang dicari sehingga string “radio” dapat digeser melewati string “kanan” sehingga posisinya menjadi :



Dalam contoh terlihat bahwa algoritma *Boyer-Moore Horspool* memiliki loncatan karakter yang besar sehingga mempercepat pencarian string karena dengan hanya memeriksa sedikit karakter, dapat langsung diketahui bahwa string yang dicari tidak ditemukan dan dapat digeser ke posisi berikutnya.

2.2.2 Cara Kerja

Langkah-langkah algoritma *Boyer-Moore Horspool* sebagai berikut.

1. Buat tabel pergeseran string yang dicari (S) dengan pendekatan Match Heuristic (MH) dan Occurence Heuristic (OH), untuk menentukan jumlah pergeseran yang akan dilakukan jika mendapat karakter tidak cocok pada proses pencocokan dengan string (T) (Sagita dan Prasetiyowati, 2013).
2. Jika dalam proses pembandingan terjadi ketidakcocokan antara pasangan karakter pada S dan karakter pada T, pergeseran dilakukan dengan memilih salah satu nilai pergeseran dari dua tabel analisa string, yang memiliki nilai pergeseran paling besar.
3. Dua kemungkinan penyelesaian dalam melakukan pergeseran S, jika sebelumnya belum ada karakter yang cocok adalah dengan melihat nilai pergeseran hanya pada tabel occurence heuristic : Jika karakter yang tidak cocok tidak ada pada S maka pegeseran adalah sebanyak jumlah karakter pada S. dan jika karakter yang tidak cocok ada pada S, maka banyaknya pergeseran bergantung dari nilai pada tabel.
4. Jika karakter pada teks yang sedang dibandingkan cocok dengan karakter pada S, maka posisi karakter pada S dan T diturunkan sebanyak 1 posisi, kemudian lanjutkan dengan pencocokan pada posisi tersebut dan seterusnya. Jika kemudian terjadi ketidakcocokan karakter S dan T, maka pilih nilai pergeseran terbesar dari dua tabel analisa pattern yaitu nilai dari tabel match heuristic dan nilai tabel occurence heuristic dikurangi dengan jumlah karakter yang telah cocok.
5. Jika semua karakter telah cocok, artinya S telah ditemukan di dalam T, selanjutnya geser pattern sebesar 1 karakter.
6. Lanjutkan sampai akhir string T.

2.2.2 Algoritma Pseudocode

Berikut ini disajikan *pseudocode* algoritma Boyer Moore *Horspool.*

No	Code
1	`Function boyer_moore_search(pattern, text):`
2	`    m <- length of pattern`
3	`    n <- length of text`
4	`    last_occurrence <- dictionary to store the last occurrence index of each character in pattern`
5	
6	`    # Initialize last_occurrence dictionary`
7	`    for i from 0 to m - 1:`
8	`        last_occurrence[pattern[i]] <- i`
9	
10	`    i <- m - 1  # index in pattern`
11	`    j <- m - 1  # index in text`
12	
13	`    while j < n:`
14	`        if pattern[i] == text[j]:`
15	`            if i == 0:`
16	`                return j  # pattern found`
17	`            else:`
18	`                i <- i - 1`
19	`                j <- j - 1`
20	`        else:`
21	`            last_occ <- last_occurrence[text[j]]`
22	`            j <- j + m - min(i, 1 + last_occ)`
23	`            i <- m - 1`
24	
25	`    return None  # pattern not found`
26	
27	`Function boyer_moore_search_name(name, data_array):`
28	`    patterns <- split and lowercase the input name`
29	`    for i, item in enumerate(data_array):`
30	`        keterangan_lower <- lowercase(item["keterangan"])`
31	`        # Check if all patterns are found in keterangan_lower`
32	`        if all(boyer_moore_search(pattern, keterangan_lower) is not None for pattern in patterns):`
33	`            return i  # Return the index instead of the date`
34	
35	`    return None  # pattern not found in any keterangan`

Penjelasan pseudocode:

Pseudocode di atas mendeskripsikan dua fungsi yang menggunakan algoritma pencarian *Boyer-Moore Horspool* untuk mencari kecocokan pola dalam teks.

Fungsi boyer_moore_search

Pseudocode untuk fungsi **boyer_moore_search** dimulai pada baris 1 dan bertujuan untuk mencari pola dalam teks yang diberikan.

- **Baris 1-5**: Deklarasi dan persiapan fungsi pencarian.
- **Baris 6-8**: Inisialisasi dictionary **last_occurrence** yang menyimpan indeks kemunculan terakhir setiap karakter dalam pola.
- **Baris 10-11**: Penetapan indeks **i** dan **j**, yang menunjuk pada posisi akhir pola dan teks yang akan dicek.
- **Baris 13-23**: Loop **while** yang berjalan selama indeks **j** dalam teks lebih kecil dari panjang teks **n**.
    - **Baris 14-19**: Jika karakter pada pola dan teks cocok, lanjutkan dengan memeriksa kecocokan karakter sebelumnya. Jika semua karakter cocok (**i** mencapai 0), kembalikan indeks **j** yang menandakan awal dari kecocokan dalam teks.
    - **Baris 20-23**: Jika tidak cocok, gunakan dictionary **last_occurrence** untuk menemukan seberapa jauh indeks **j** harus melompat dalam teks. Atur ulang **i** ke ujung pola.
- **Baris 25**: Jika pola tidak ditemukan, kembalikan **None**.

Fungsi boyer_moore_search_name

Pseudocode untuk fungsi **boyer_moore_search_name** dimulai pada baris 27 dan bertujuan untuk mencari setiap kata dalam nama (yang telah dipisahkan dan dikecilkan hurufnya) dalam array dari data.

- **Baris 27-28**: Deklarasi fungsi dan pembagian serta pengkecilan huruf dari **name** menjadi beberapa pola.
- **Baris 29-33**: Iterasi melalui setiap elemen dalam **data_array**.
    - **Baris 30**: Ubah **keterangan** dalam item menjadi huruf kecil.
    - **Baris 32**: Periksa apakah semua pola dari **name** ditemukan dalam **keterangan_lower** menggunakan fungsi **boyer_moore_search**. Jika ya, kembalikan indeks item tersebut.

- **Baris 35**: Jika tidak ada kecocokan yang ditemukan di seluruh **data_array**, kembalikan **None**.

2.2.3 Algoritma Flowchart dari *Boyer Moore Search*

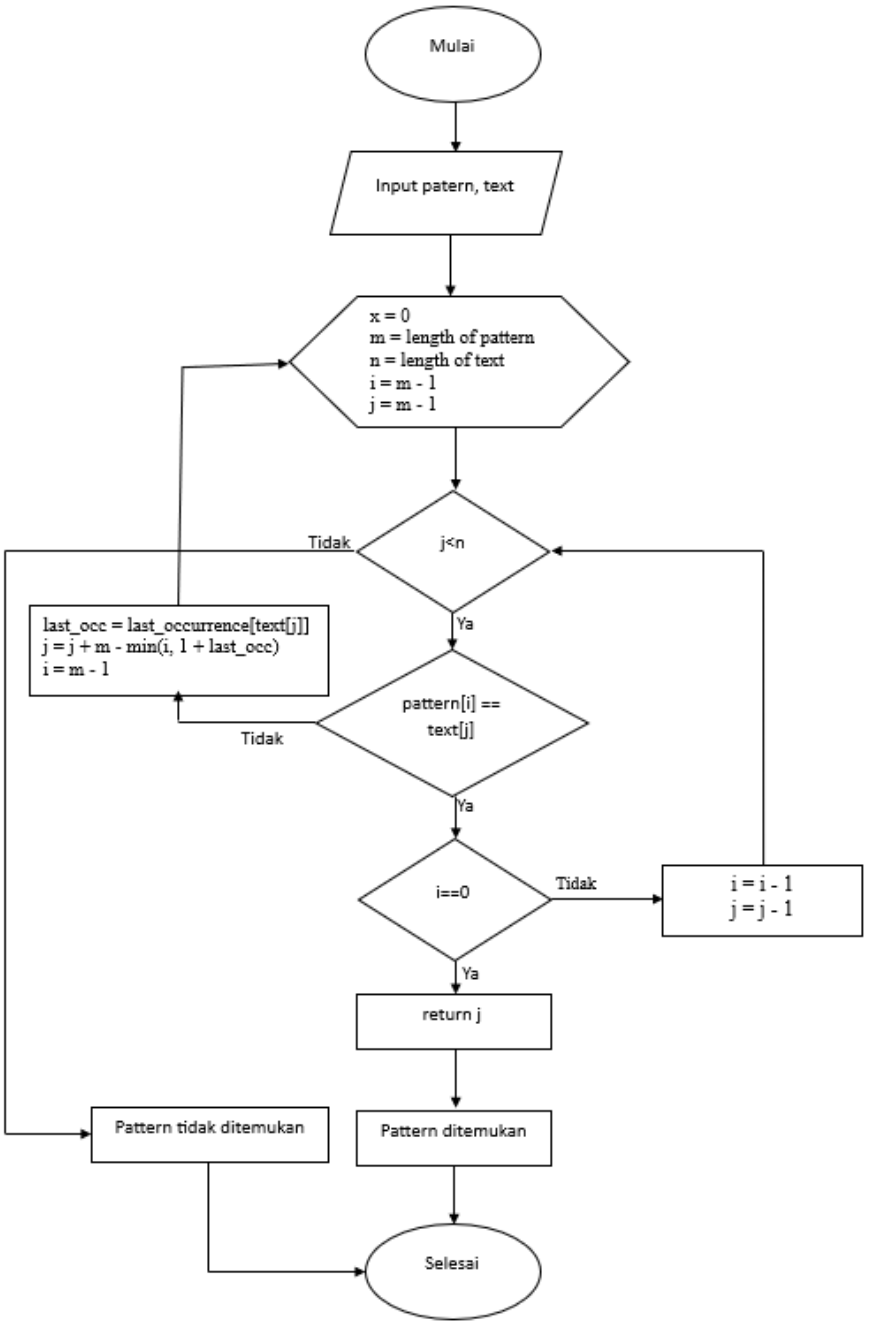
```
Input: pattern, text

x = 0
m = length of pattern
n = length of text
last_occurrence = create_last_occurrence_table(pattern)
i = m - 1
j = m - 1

while j < n:
    if pattern[i] == text[j]:
        if i == 0:
            return j  # pattern found
        else:
            i = i - 1
            j = j - 1
    else:
        last_occ = last_occurrence[text[j]]
        j = j + m - min(i, 1 + last_occ)
        i = m - 1

return None  # pattern not found
```

Flowchart Fungsi boyer_moore_search

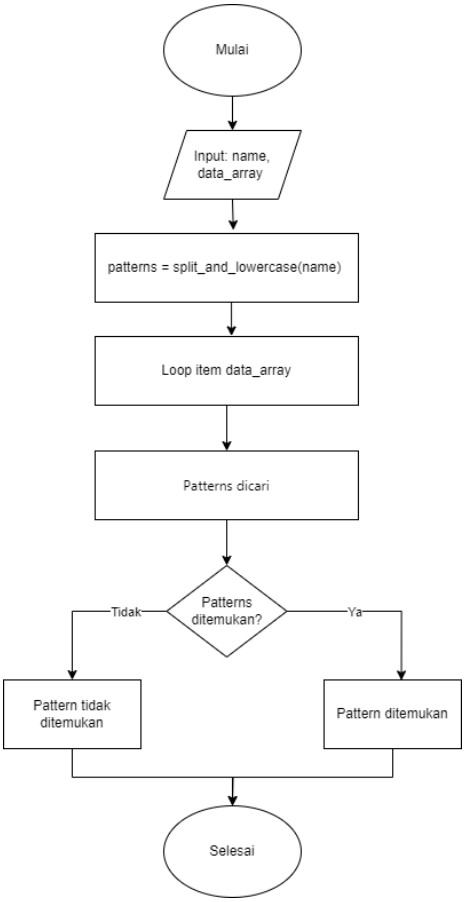


Flowchart di atas merupakan representasi visual dari algoritma pencarian string *Boyer-Moore Horspool*. Algoritma ini adalah metode efisien untuk menemukan kejadian suatu pola dalam teks. Berikut adalah penjelasan detail algoritma tersebut dalam bahasa Indonesia, berdasarkan flowchart.

1. **Mulai**: Proses dimulai.
2. **Input pattern, text**: Masukkan pola (pattern) yang ingin dicari dan teks tempat pencarian akan dilakukan.
3. **Inisialisasi variabel**: Variabel **x** diatur ke 0, **m** adalah panjang pola, **n** adalah panjang teks, **i** dan **j** diatur ke nilai **m - 1**. Fungsi **create_last_occurrence_table(pattern)** digunakan untuk membuat tabel yang berisi informasi terakhir kali setiap karakter muncul dalam pola.
4. **Perulangan pencarian**: Selama **j** kurang dari **n**, algoritma akan melakukan iterasi melalui teks dengan membandingkan teks dan pola dari belakang ke depan.
5. **Pemeriksaan karakter**: Jika karakter pada pola (**pattern[i]**) sama dengan karakter pada teks (**text[j]**):
    - Jika **i** adalah 0, ini berarti seluruh pola telah ditemukan dalam teks pada posisi **j**, dan fungsi akan mengembalikan nilai **j** (pola ditemukan).
    - Jika **i** bukan 0, dekremen **i** dan **j** untuk melanjutkan perbandingan ke karakter sebelumnya.
6. **Karakter tidak cocok**: Jika karakter tidak cocok, algoritma akan mencari tabel kejadian terakhir untuk mengetahui langkah berapa kali indeks **j** harus digeser. Ini dilakukan dengan mengambil nilai **last_occurrence** dari karakter **text[j]** dan menghitung nilai baru **j** dengan rumus **j + m - min(i, 1 + last_occ)**. Set **i** kembali ke **m - 1** untuk memulai pencocokan dengan pola dari awal.
7. **Pola tidak ditemukan**: Jika algoritma selesai melakukan iterasi melalui teks dan pola tidak ditemukan, fungsi akan mengembalikan **None**.
8. **Selesai**: Proses pencarian selesai.

Algoritma *Boyer-Moore Horspool* sangat efisien karena biasanya melompati lebih banyak karakter dibandingkan algoritma pencarian string lainnya. Ini dilakukan dengan memanfaatkan informasi tentang pola dan karakter yang tidak cocok yang telah diperiksa sebelumnya

Flowchart Fungsi boyer_moore_search_name



Flowchart di atas adalah diagram alir yang menjelaskan proses pencarian pola dalam sebuah array data. Berikut adalah penjelasan langkah demi langkah dari flowchart tersebut.

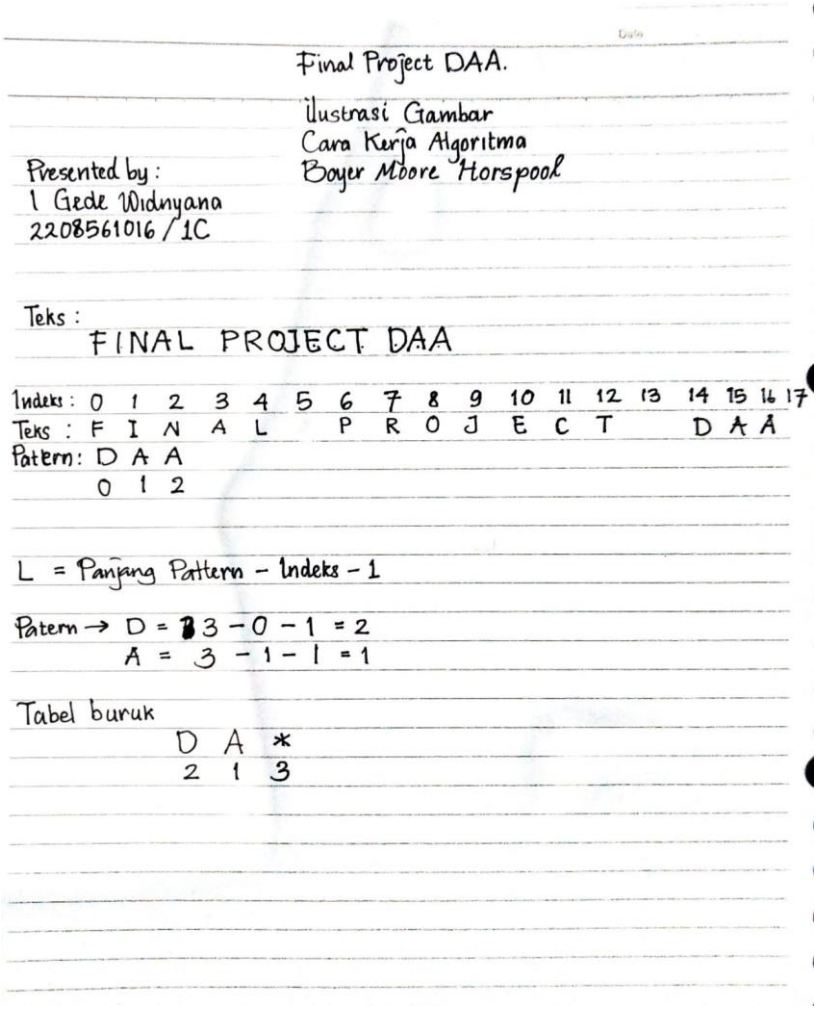
1. **Mulai**: Proses dimulai.
2. **Input**: Input yang diminta adalah **name** dan **data_array**. **name** mungkin merupakan string yang akan dicari, dan **data_array** adalah kumpulan data tempat pencarian akan dilakukan.
3. **Proses Pembentukan Pola**: String **name** diproses dengan fungsi **split_and_lowercase**, yang kemungkinan memecah string menjadi beberapa kata dan mengubah semua huruf menjadi huruf kecil untuk normalisasi.
4. **Looping Melalui Data**: Program kemudian melakukan iterasi melalui setiap item dalam **data_array**. Ini adalah langkah looping di mana setiap elemen array akan diperiksa.
5. **Pencarian Pola**: Untuk setiap item dalam loop, proses pencarian pola dilakukan. Detail tentang bagaimana pola dicari tidak dijelaskan, tetapi bisa

jadi melalui pencocokan string langsung, penggunaan ekspresi reguler, atau algoritma pencarian lainnya.

6. **Pemeriksaan Hasil Pencarian**: Setelah pencarian dilakukan, ada pemeriksaan kondisi untuk menentukan apakah pola telah ditemukan atau tidak.
7. **Hasil Pencarian**:
   - Jika pola **tidak ditemukan**, flowchart mengarah ke output yang menyatakan "Pattern tidak ditemukan" (Pola tidak ditemukan).
   - Jika pola **ditemukan**, flowchart mengarah ke output yang menyatakan "Pattern ditemukan" (Pola ditemukan).
8. **Selesai**: Proses pencarian selesai, dan flowchart berakhir.

2.2.3 Ilustrasi Cara Kerja Algoritma

Berikut ini penulis sajikan ilustrasi cara kerja algoritma *Boyer Moore* menggunakan tools website berikut ini : http://whocouldthat.be/visualizing-string-matching/



Teks : Final Project DAA

Indeks	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Teks	F	I	N	A	L		P	R	O	J	E	C	T		D	A	A
Patern	D	A	A														
Indeks Patern	0	1	2														

Last Occurrence (L) = Panjang Patern - Indeks Patern - 1

Patern huruf

D => 3 - 0 - 1 = 2

A => 3 - 1 - 1 = 1

Tabel Buruk		
D	A	*
2	1	3

Penjelasan Alur Kerja Algoritma *Boyer-Moore Horspool*:

1. **Inisialisasi Last Occurrence Table**:
    - Ini adalah tabel yang menginformasikan seberapa jauh kita harus melompat dalam teks ketika terjadi ketidakcocokan karakter.
    - Tabel ini diisi dengan menghitung **Panjang Patern - Indeks Patern - 1** untuk setiap karakter unik dalam pola.
    - Dalam contoh ini, tabel terakhir berisi:

- Untuk 'D', **3 - 0 - 1 = 2**.
- Untuk 'A', **3 - 1 - 1 = 1**.

2. **Proses Pencarian**:
    - Pencarian dimulai dengan menempatkan pola di akhir teks. Dalam contoh ini, pola "DAA" dimulai dari indeks 14 di teks.
    - Pencarian dimulai dari karakter terakhir pola, yang dalam kasus ini adalah 'A'.
3. **Pengecekan Karakter**:
    - Bandingkan karakter pola dari kanan ke kiri dengan teks yang sesuai.
    - Jika karakter cocok, lanjutkan ke karakter sebelumnya dari pola dan teks.
    - Jika karakter tidak cocok, lihat tabel lompatan terakhir untuk menentukan berapa banyak karakter yang dapat dilewati.
4. **Ketidakcocokan dan Lompatan**:
    - Dalam contoh ini, karakter 'A' pada indeks 15 dari teks cocok dengan 'A' terakhir dari pola.
    - Kemudian kita pindah ke karakter sebelumnya, 'D' di pola dan 'C' di teks. Karena tidak cocok, kita lihat tabel lompatan terakhir.
    - Kita lihat karakter teks yang tidak cocok ('C') tidak ada dalam tabel, jadi kita menggunakan nilai default yang biasanya adalah panjang pola (dalam contoh ini, kita menggunakan nilai 3 dari tabel lompatan untuk karakter '*').
    - Maka, pola melompat 3 karakter ke kanan dan proses pencocokan dimulai lagi dari indeks 14 dari teks.
5. **Pencarian Berlanjut**:
    - Proses ini berlanjut sampai pola cocok dengan teks atau sampai pola tidak dapat lagi melompat ke kanan, yang berarti pola tidak ditemukan dalam teks.

2.2.5 Analisis Kompleksitas Waktu Algoritma pada Program

No	Source Code file
1	`def boyer_moore_horspool_search(pattern, text):`
2	`    m = len(pattern)  # O(1)`
3	`    n = len(text)     # O(1)`
4	`    pattern = pattern.lower()  # O(m)`
5	`    text = text.lower()        # O(n)`
6	`    last_occurrence = {pattern[i]: i for i in range(m)}  # O(m)`
7	
8	`    i = m - 1  # O(1)`
9	`    j = m - 1  # O(1)`
10	
11	`    while j < n:  # O(n) * (O(m) + O(1)) -> O(n*m)`
12	`        if pattern[i] == text[j]:  # O(1)`
13	`            if i == 0:  # O(1)`
14	`                return j  # O(1)`
15	`            else:`
16	`                i -= 1  # O(1)`
17	`                j -= 1  # O(1)`
18	`        else:`
19	`            last_occ = last_occurrence.get(text[j], -1)  # O(1)`
20	`            j = j + m - min(i, 1 + last_occ)  # O(1)`
21	`            i = m - 1  # O(1)`
22	
23	`    return None  # O(1)`
24	
25	`def search_phrase_in_text(phrase, text):`
26	`    return phrase.lower() in text.lower()  # O(m + n)`
27	
28	`def boyer_moore_horspool_search_name(name, data_array):`
29	`    name = name.lower()  # O(m)`
30	
31	`    for i, item in enumerate(data_array):  # O(k) * (O(m + n) + O(1))`
32	`        keterangan_lower = item["keterangan"].lower()  # O(n)`
33	`        if search_phrase_in_text(name, keterangan_lower):  # O(m + n)`
34	`            return i  # O(1)`
35	
36	`    return None  # O(1)`

1. **Fungsi boyer_moore_horspool_search**

Baris Kode	Kompleksitas	Penjelasan
m = len(pattern)	O(1)	Menghitung panjang string pola adalah operasi konstan.
n = len(text)	O(1)	Menghitung panjang string dari teks adalah operasi konstan.
pattern = pattern.lower()	O(m)	Mengubah ke huruf kecil dilakukan untuk setiap karakter dalam **pattern**.
text = text.lower()	O(n)	Mengubah ke huruf kecil dilakukan untuk setiap karakter dalam **text.**
last_occurrence = {pattern[i]: i for i in range(m)}	O(m)	Dictionary comprehension ini berjalan sebanyak **m** kali.
i = m - 1	O(1)	Operasi aritmatika (pengurangan) sederhana.
j = m - 1	O(1)	Operasi aritmatika (pengurangan) sederhana.
while j < n:	O(n*m)	Loop ini dapat berjalan hingga **n** kali. Di dalamnya, terdapat operasi yang berjalan hingga **m** kali.
if pattern[i] == text[j]:	O(1)	Perbandingan karakter adalah operasi konstan.
if i == 0:	O(1)	Perbandingan sederhana.
i -= 1	O(1)	Operasi aritmatika (pengurangan) sederhana.
j -= 1	O(1)	Operasi aritmatika (pengurangan) sederhana.
last_occ = last_occurrence.get(text[j], -1)	O(1)	Akses dictionary adalah operasi konstan.
j = j + m - min(i, 1 + last_occ)	O(1)	Beberapa operasi aritmatika sederhana.
i = m - 1	O(1)	Operasi aritmatika (pengurangan) sederhana.
return None	O(1)	Mengembalikan nilai adalah operasi konstan.

Kompleksitas Waktu: 12 x O(1) + O(n*m) + O(m) + O (m) + O (n) = O(m*n) + O(2m) + O(n) + O(m*n).

Sehingga notasi *Big-O* adalah O(mn).

Kompleksitas ini disebabkan oleh loop while, yang berjalan sebanyak n kali, dan dalam setiap iterasi, ada operasi yang tergantung pada m.

2. Fungsi search_phrase_in_text

Baris Kode	Kompleksitas	Penjelasan
return phrase.lower() in text.lower()	O(m + n)	Mengubah kedua string menjadi lowercase membutuhkan O(m) dan O(n), dan pengecekan **in** juga memerlukan hingga O(n).

Kompleksitas Waktu: O(m + n)

Sehingga notasi *Big-O* adalah **O(m+n)**.

Pengkonversian ke lowercase membutuhkan O(m) dan O(n), dan operasi in juga memerlukan O(n).

3. Fungsi boyer_moore_horspool_search_name

Baris Kode	Kompleksitas	Penjelasan
name = name.lower()	O(m)	Mengubah **name** menjadi lowercase, memerlukan O(m).
for i, item in enumerate(data_array):	O(k * (m + n))	Loop berjalan **k** kali (jumlah elemen dalam **data_array**). Setiap iterasi melibatkan operasi O(m + n).
keterangan_lower = item["keterangan"].lower()	O(n)	Mengubah **keterangan** menjadi lowercase, memerlukan O(n).
if search_phrase_in_text(name, keterangan_lower):	O(m + n)	Memanggil fungsi **search_phrase_in_text** yang memiliki kompleksitas O(m + n).
return i	O(1)	Mengembalikan nilai adalah operasi konstan.
return None	O(1)	Mengembalikan nilai adalah operasi konstan.

Kompleksitas Waktu: O(m) + O(k * (m + n)) + O(n) + O(m + n) + O(2)

Notasi *Big O* adalah **O (k * (m + n))**. Iterasi pada data_array dilakukan k kali (ukuran data_array). Dalam setiap iterasi, operasi pencarian membutuhkan O(m + n).

Kesimpulan :

Kompleksitas waktu terbesar dari algoritma ini adalah O(n*m) untuk fungsi boyer_moore_horspool_search, di mana n adalah panjang teks dan m adalah panjang pola. Untuk fungsi boyer_moore_horspool_search_name, kompleksitasnya adalah O(k * (m + n)), di mana k adalah jumlah elemen dalam data_array.

Kompleksitas Waktu Berdasarkan Referensi

Berdasarkan observasi penulis dalam mengkaji kompleksitas waktu dari sumber website https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/. Hasil analisa kompleksitas waktu dari algoritma boyer-moore horspool berdasarkan sumber itu adalah sebagai berikut.

Kompleksitas Waktu : O(n x m)

Ruang Tambahan : O(1)

Heuristik Karakter Buruk mungkin membutuhkan O(mn) waktu dalam kasus terburuk. Kasus terburuk terjadi ketika semua karakter teks dan pola sama. Misalnya, txt[] = "AAAAAAAAAAAAAAAAAAAA" dan pat[] = "AAAAA". *The Bad Character Heuristic* dapat mengambil O (n / m) dalam kasus terbaik. Kasus terbaik terjadi ketika semua karakter teks dan pola berbeda.

Time Complexity : O(n x m)

Auxiliary Space: O(1)

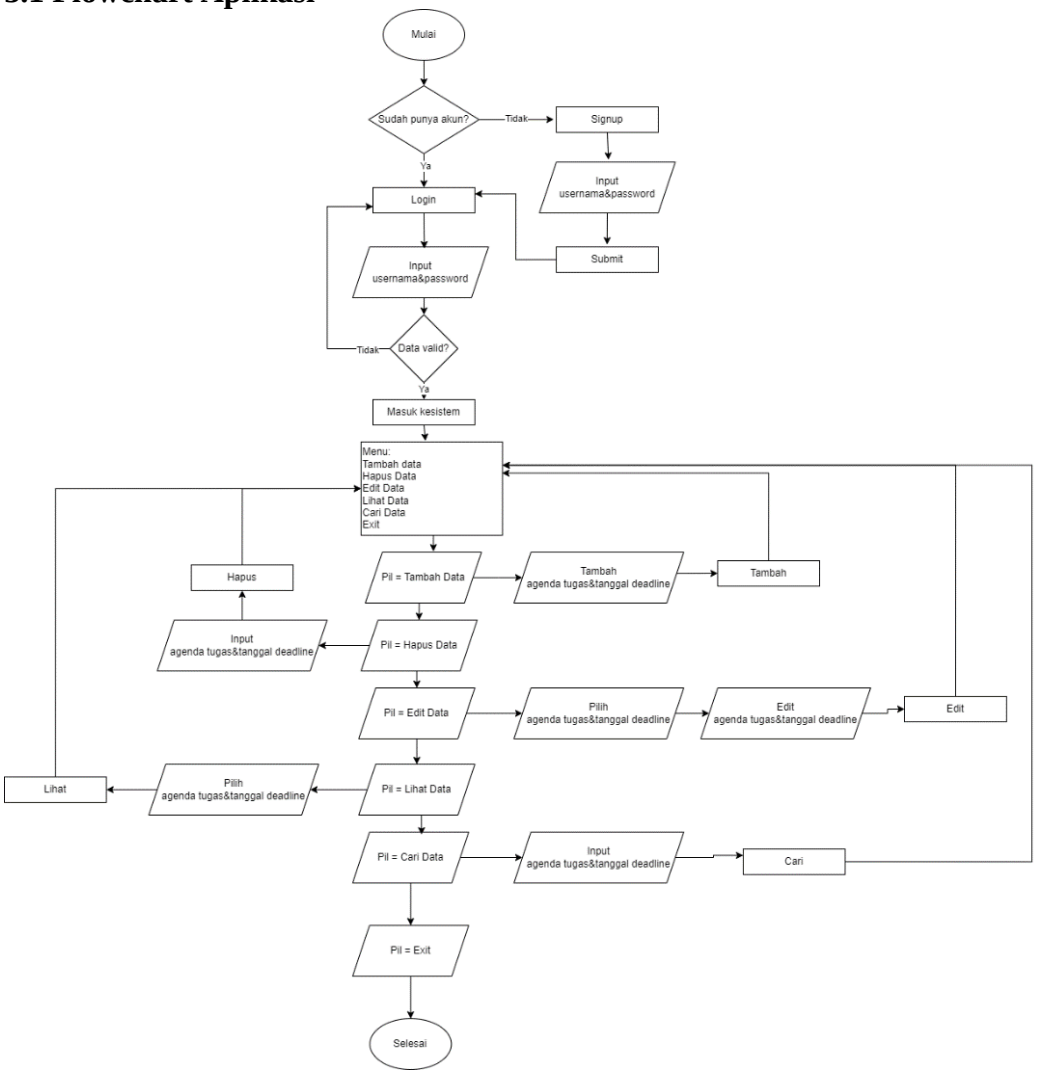
The Bad Character Heuristic may take $O(mn)$ time in worst case. The worst case occurs when all characters of the text and pattern are same. For example, txt[] = "AAAAAAAAAAAAAAAAAA" and pat[] = "AAAAA". The Bad Character Heuristic may take O(n/m) in the best case. The best case occurs when all the characters of the text and pattern are different.

Sumber : Boyer Moore Algorithm for Pattern Searching - GeeksforGeeks

BAB II

Implementasi Algoritma

3.1 Flowchart Aplikasi



Penjelasan flowchart:

Berikut penjelasan detil dari setiap langkah dalam flowchart:

1. **Mulai**: Alur dimulai dari titik ini.
2. **Sudah punya akun?**: Pertanyaan ini memeriksa apakah pengguna sudah memiliki akun.
    - Jika **Tidak**, pengguna akan diarahkan untuk melakukan **Signup**.
        - **Signup**: Pengguna diminta untuk menginput **username** dan **password**, lalu menekan **Submit**.

- Jika **Ya**, pengguna akan diarahkan untuk **Login**.
    - **Login**: Pengguna diminta untuk menginput **username** dan **password**, lalu menekan **Submit**.

3. **Data valid?**: Sistem memeriksa apakah data yang dimasukkan valid.
    - Jika **Tidak**, pengguna kembali ke langkah **Login**.
    - Jika **Ya**, pengguna masuk ke sistem.
4. **Masuk ke sistem**: Pengguna diberikan menu dengan pilihan berikut:
    - **Tambah data**
    - **Hapus data**
    - **Edit data**
    - **Lihat data**
    - **Cari data**
    - **Exit**
5. **Pilihan Menu**:
    - **Pilih Tambah Data**:
        - Pengguna menginput **agenda tugas/tanggal deadline** dan memilih **Tambah**.
    - **Pilih Hapus Data**:
        - Pengguna menginput **agenda tugas/tanggal deadline** dan memilih **Hapus**.
    - **Pilih Edit Data**:
        - Pengguna pertama-tama **Pilih agenda tugas/tanggal deadline**, kemudian menginput data yang baru dan memilih **Edit**.
    - **Pilih Lihat Data**:
        - Pengguna akan melihat daftar **agenda tugas/tanggal deadline**.
    - **Pilih Cari Data**:
        - Pengguna menginput **agenda tugas/tanggal deadline** yang ingin dicari dan memilih **Cari**.
    - **Pilih Exit**:
        - Pengguna memilih **Exit** untuk keluar dari sistem.

6. **Selesai**: Akhir dari alur kerja setelah pengguna memilih **Exit**.

Setiap "Pilih" pada menu kemudian akan mengarahkan kembali ke menu utama setelah aksi yang dipilih selesai dilaksanakan, kecuali untuk pilihan **Exit** yang mengakhiri sesi pengguna di sistem.

3.2 Source Code

Berikut ini adalah *source code* yang membangun program.

No	Source Code
1	`import tkinter as tk`
2	`import sys`
3	`from tkinter import messagebox`
4	`from tkinter import ttk`
5	`from tkinter import simpledialog`
6	
7	`# Set the Cambria font family`
8	`cambria_font = ("Cambria", 12)`
9	
10	`# Definisikan kelas Lg untuk menyimpan informasi login`
11	`class Lg:`
12	`    def __init__(self, username="", password=""):`
13	`        self.username = username`
14	`        self.password = password`
15	
16	`# Fungsi untuk membuat akun baru`
17	`def buatakun():`
18	`    # Buat jendela baru untuk sign-up`
19	`    signup_window = tk.Toplevel(root)`
20	`    signup_window.title("Sign Up")`
21	`    signup_window.geometry("300x150")`
22	
23	`    # Label dan entry untuk username`
24	`    username_label = ttk.Label(signup_window, text="Username:", font=cambria_font)`
25	`    username_label.grid(row=0, column=0, sticky=tk.W)`
26	`    username_entry = ttk.Entry(signup_window)`
27	`    username_entry.grid(row=0, column=1)`
28	
29	`    # Label dan entry untuk password`
30	`    password_label = ttk.Label(signup_window, text="Password:", font=cambria_font)`
31	`    password_label.grid(row=1, column=0, sticky=tk.W)`
32	`    password_entry = ttk.Entry(signup_window, show="*")`

```
    password_entry.grid(row=1, column=1)

    # Tombol untuk menyimpan data sign-up
    signup_button = ttk.Button(signup_window, text="Sign Up", command=lambda: save_signup(username_entry.get(), password_entry.get(), signup_window))
    signup_button.grid(row=2, column=1, pady=10)

# Fungsi untuk menyimpan data sign-up ke dalam file
def save_signup(username, password, signup_window):
    with open('login.txt', 'a') as file:
        file.write(f"{username},{password}\n")
    print("\nAkun berhasil dibuat!")
    signup_window.destroy()

# Fungsi untuk memeriksa login
def check_login():
    lg = loginakun()
    if lg is not None:
        # Jika login berhasil, tampilkan pesan sukses dan arahkan ke file lain
        messagebox.showinfo("Login Success", "Welcome, " + lg.username + "!")
        root.destroy()

# Fungsi untuk melakukan login
def loginakun():
    lg = None
    with open('login.txt', 'r') as file:
        # Minta input username dan password dari pengguna
        username = simpledialog.askstring("Input", "Masukkan username:")
        password = simpledialog.askstring("Input", "Masukkan password:")
        # Loop melalui file untuk mencocokkan dengan data yang tersimpan
        for line in file:
            # Memastikan bahwa line tidak kosong dan memiliki dua nilai yang dapat dipisahkan oleh koma
            if ',' in line:
                stored_username, stored_password = line.strip().split(',', 1)
                if username == stored_username and password == stored_password:
                    lg = Lg(username, password)
                    print("\nLogin berhasil!")
                    break
```

```
70
71      if lg is None:
72          print("\nUsername atau password salah!")
73
74      return lg
75
76  # Fungsi untuk menutup aplikasi
77  def close_application():
78      confirm = messagebox.askokcancel("Konfirmasi", "Apakah
    Anda yakin ingin keluar dari aplikasi?")
79      if confirm:
80          sys.exit()
81
82  # Membuat instance object tkinter
83  root = tk.Tk()
84
85  # Menambahkan judul pada GUI
86  root.title("Aplikasi Pencatatan Deadline Tugas")
87
88  # Menambahkan judul pada GUI
89  root.title("Login")
90
91  # Membuat frame untuk input login
92  frame_login = tk.Frame(root, padx=20, pady=20)
93  frame_login.pack()
94
95  # Label for the application title
96  app_title_label = ttk.Label(frame_login, text="Aplikasi
    Pencatatan Deadline Tugas", font=cambria_font)
97  app_title_label.grid(row=0, column=0, columnspan=2,
    pady=10)
98
99  # Menambahkan tombol login
100 login_button = tk.Button(frame_login, text="Login",
    command=check_login, font=cambria_font)
101 login_button.grid(row=2, column=0, padx=5, pady=10)
102
103 # Menambahkan tombol sign up
104 signup_button = tk.Button(frame_login, text="Sign Up",
    command=buatakun, font=cambria_font)
105 signup_button.grid(row=2, column=1, padx=5, pady=10)
106
107 # Menutup aplikasi saat tombol close pada window login
    ditekan
108 def disable_event():
109     close_application()
110
```

```
root.protocol("WM_DELETE_WINDOW", disable_event)

# Menjalankan event loop tkinter
root.mainloop()

# Define data_array globally
data_array = [
    {"tanggal": "18 Desember 2023", "keterangan": "FP Basdat"},
    {"tanggal": "18 Desember 2023", "keterangan": "FP DAA"},
    {"tanggal": "18 Desember 2023", "keterangan": "FP RPL"},
    {"tanggal": "18 Desember 2023", "keterangan": "FP TBO"},
    {"tanggal": "19 Desember 2023", "keterangan": "FP PBO"},
    {"tanggal": "21 Desember 2023", "keterangan": "FP KDJK"},
    {"tanggal": "21 Desember 2023", "keterangan": "TEST LISAN BASDAT"},
    {"tanggal": "22 Desember 2023", "keterangan": "TEST KDJK"},
    {"tanggal": "28 Desember 2023", "keterangan": "TEST LISAN TBO DAN REVISI"},
    {"tanggal": "21 Desember 2023", "keterangan": "REVISI IMK"}
]

def boyer_moore_horspool_search(pattern, text):
    m = len(pattern)
    n = len(text)
    pattern = pattern.lower()
    text = text.lower()
    last_occurrence = {pattern[i]: i for i in range(m)}
    i = m - 1  # index in pattern
    j = m - 1  # index in text

    while j < n:
        if pattern[i] == text[j]:
            if i == 0:
                return j  # pattern found
            else:
                i -= 1
                j -= 1
        else:
            last_occ = last_occurrence.get(text[j], -1)
            j = j + m - min(i, 1 + last_occ)
            i = m - 1
```

```
150
151     return None
152
153 def search_phrase_in_text(phrase, text):
154     return phrase.lower() in text.lower()
155
156 def boyer_moore_horspool_search_name(name, data_array):
157     # Convert the query to lowercase
158     name = name.lower()
159
160     for i, item in enumerate(data_array):
161         keterangan_lower = item["keterangan"].lower()
162         # Search for the entire query phrase in
keterangan_lower using the new function
163         if search_phrase_in_text(name, keterangan_lower):
164             return i  # Return the index instead of the
date
165
166     return None
167
168 # Fungsi untuk menampilkan hasil pencarian
169 def show_result(result_index):
170     if result_index is not None:
171         message = f"Tanggal deadline
{data_array[result_index]['tanggal']}"
172     else:
173         message = "Agenda Tugas yang Anda cari tidak
ditemukan."
174     result_label.configure(text=message)
175
176
177 # Fungsi untuk menambah data pada array dan memperbarui
tampilan tabel
178 def add_data():
179     keterangan = keterangan_entry.get()
180     tanggal = tanggal_entry.get()  # No conversion to
lowercase for dates
181     data_array.append({"tanggal": tanggal, "keterangan":
keterangan})
182     update_table()
183
184 # Fungsi untuk menghapus data dari array dan memperbarui
tampilan tabel
185 def delete_data():
186     item = table.selection()[0]
187     index = int(table.index(item))
188     del data_array[index]
189     update_table()
```

```
190
191 # Fungsi untuk memperbarui tampilan tabel
192 def update_table():
193     table.delete(*table.get_children())
194     for i, data in enumerate(data_array):
195         table.insert("", "end", text=i+1,
values=(data["keterangan"], data["tanggal"], i))  # Pass i
as the third argument
196
197 # Fungsi untuk mengedit data pada array dan memperbarui
tampilan tabel
198 def edit_data(index):
199     # Tampilkan jendela dialog untuk mengedit data
200     edit_window = tk.Toplevel(root)
201     edit_window.title("Edit Data")
202     edit_window.geometry("300x150")  # Set width and height
to match the "View Data" window
203
204     # Label dan entry untuk input keterangan
205     keterangan_label = ttk.Label(edit_window, text="Agenda
Tugas:")
206     keterangan_label.grid(row=0, column=0, sticky=tk.W)
207     keterangan_entry = ttk.Entry(edit_window)
208     keterangan_entry.insert(0,
data_array[index]["keterangan"])
209     keterangan_entry.grid(row=0, column=1, padx=5, pady=5)
210
211     # Label dan entry untuk input tanggal
212     tanggal_label = ttk.Label(edit_window, text="Tanggal
deadline:")
213     tanggal_label.grid(row=1, column=0, sticky=tk.W)
214     tanggal_entry = ttk.Entry(edit_window)
215     tanggal_entry.insert(0, data_array[index]["tanggal"])
216     tanggal_entry.grid(row=1, column=1, padx=5, pady=5)
217
218     # Tombol untuk menyimpan perubahan
219     save_button = ttk.Button(edit_window, text="Save",
command=lambda: save_changes(index, keterangan_entry.get(),
tanggal_entry.get(), edit_window))
220     save_button.grid(row=2, column=1, pady=10)
221
222 # Fungsi untuk menyimpan perubahan ke dalam data_array dan
memperbarui tampilan tabel
223 def save_changes(index, keterangan, tanggal, edit_window):
224     data_array[index]["keterangan"] = keterangan
225     data_array[index]["tanggal"] = tanggal
226     edit_window.destroy()  # Tutup jendela edit
227     update_table()  # Perbarui tampilan tabel
```

```
228
229 # Membuat instance object tkinter
230 root = tk.Tk()
231
232 # Menambahkan judul pada GUI
233 root.title("LIST DEADLINE TUGAS")
234
235 # Membuat frame untuk tabel dan input
236 frame_table = ttk.Frame(root, padding="20")
237 frame_table.pack(fill="both", expand=True)
238 frame_input = ttk.Frame(root, padding="10")
239 frame_input.pack(fill="x", expand=True)
240 frame_button = ttk.Frame(root, padding="10")
241 frame_button.pack(fill="x", expand=True)
242
243 # Membuat tabel untuk menampilkan data array
244 table = ttk.Treeview(frame_table, columns=("col1"))
245 table.heading("#0", text="No.")
246 table.heading("col1", text="Keterangan")
247
248 # Menampilkan tabel ke dalam GUI
249 table.pack(fill="both", expand=True)
250
251 # Menambahkan input untuk menambah data
252 keterangan_label = ttk.Label(frame_input, text="Agenda
    Tugas :")
253 keterangan_label.pack(side="left")
254 keterangan_entry = ttk.Entry(frame_input)
255 keterangan_entry.pack(side="left")
256 tanggal_label = ttk.Label(frame_input, text="Tanggal
    deadline:")
257 tanggal_label.pack(side="left")
258 tanggal_entry = ttk.Entry(frame_input)
259 tanggal_entry.pack(side="left")
260
261 # Menambahkan tombol untuk menambah dan menghapus data
262 add_button = ttk.Button(frame_button, text="Tambah Data",
    command=add_data)
263 add_button.pack(side="left")
264 delete_button = ttk.Button(frame_button, text="Hapus Data",
    command=delete_data)
265 delete_button.pack(side="left", padx=10)
266
267 def edit_selected_data():
268     selected_item = table.selection()
269     if selected_item:
270         index = table.index(selected_item)
```

```
271         edit_data(index)
272     else:
273         messagebox.showinfo("Info", "Pilih data yang akan
diedit.")
274
275 # Menambahkan tombol "Edit Data"
276 edit_button = ttk.Button(frame_button, text="Edit Data",
command=edit_selected_data)
277 edit_button.pack(side="left", padx=10)
278
279 def view_selected_data():
280     selected_item = table.selection()
281     if selected_item:
282         index = table.index(selected_item)
283         view_data(index)
284     else:
285         messagebox.showinfo("Info", "Pilih data yang akan
dilihat.")
286
287 def view_data(index):
288     # Tampilkan jendela dialog untuk melihat data
289     view_window = tk.Toplevel(root)
290     view_window.title("Lihat Data")
291     view_window.geometry("300x150")  # Set width and height
as needed
292
293     # Label untuk menampilkan keterangan
294     keterangan_label = ttk.Label(view_window, text="Agenda
Tugas:")
295     keterangan_label.grid(row=0, column=0, sticky=tk.W)
296     keterangan_value = ttk.Label(view_window,
text=data_array[index]["keterangan"])
297     keterangan_value.grid(row=0, column=1, padx=5, pady=5)
298
299     # Label untuk menampilkan tanggal
300     tanggal_label = ttk.Label(view_window, text="Tanggal
deadline:")
301     tanggal_label.grid(row=1, column=0, sticky=tk.W)
302     tanggal_value = ttk.Label(view_window,
text=data_array[index]["tanggal"])
303     tanggal_value.grid(row=1, column=1, padx=5, pady=5)
304
305 # Menambahkan tombol "View Data"
306 view_button = ttk.Button(frame_button, text="Lihat Data",
command=view_selected_data)
307 view_button.pack(side="left", padx=10)
308
309 def search_data():
```

```
310     result_index = boyer_moore_horspool_search_name(input_entry.get(), data_array)
311     show_result(result_index)
312
313 # Membuat input label dan button
314 input_label = ttk.Label(frame_input, text="Cari Agenda :")
315 input_label.pack(side="left")
316 input_entry = ttk.Entry(frame_input)
317 input_entry.pack(side="left")
318 search_button = ttk.Button(frame_input, text="Cari", command=search_data)
319 search_button.pack(side="left")
320
321 # Menambahkan label untuk menampilkan hasil pencarian
322 result_label = ttk.Label(root, text="")
323 result_label.pack()
324
325 # Memperbarui tampilan tabel dengan data array
326 update_table()
327
328 # Fungsi untuk menutup aplikasi dengan konfirmasi
329 def exit_application():
330     confirm = messagebox.askokcancel("Konfirmasi", "Apakah Anda yakin ingin keluar dari aplikasi?")
331     if confirm:
332         root.destroy()
333
334 # Menambahkan tombol "Exit" dengan konfirmasi
335 exit_button = ttk.Button(frame_button, text="Exit", command=exit_application)
336 exit_button.pack(side="left", padx=10)
337
338 # Menjalankan GUI
339 root.mainloop()
340
341 # Dibuat oleh :
342 # Kelompok 1 Kelas C
343 # I Gede Widnyana               (2208561016)
344 # Ni Made Viona Rara Santhi     (2208561098)
```

3.3 Penjelasan Source Code

Penjelasan Code Line 1 - 37 sebagai berikut.

No	Code	Penjelasan
1	**import tkinter as tk**	Mengimpor modul **tkinter** yang digunakan untuk membuat antarmuka grafis pada Python, diberi alias **tk** untuk memudahkan penggunaan.
2	**import sys**	Mengimpor modul **sys** yang menyediakan akses ke beberapa variabel yang digunakan atau diatur oleh interpreter Python.
3	**from tkinter import messagebox**	Mengimpor kelas **messagebox** dari modul **tkinter** untuk menampilkan pesan dialog. Digunakan untuk menampilkan pesan informasi atau kesalahan.
4	**from tkinter import ttk**	Mengimpor modul **ttk** dari modul **tkinter** yang berisi widget-themed tambahan yang lebih modern dan estetis.
5	**from tkinter import simpledialog**	Mengimpor modul **simpledialog** dari modul **tkinter** yang menyediakan fasilitas pembuatan dialog sederhana dengan mudah.
7	**cambria_font = ("Cambria", 12)**	Mendefinisikan variabel **cambria_font** untuk menyimpan informasi tentang font yang akan digunakan pada antarmuka pengguna.
10	**class Lg:**	Mendefinisikan kelas **Lg** yang akan digunakan untuk menyimpan informasi login, seperti **username** dan **password**.
11	**def __init__(self, username="", password=""):**	Konstruktor kelas **Lg** dengan dua parameter opsional, **username** dan **password**. Digunakan untuk menginisialisasi objek **Lg** dengan nilai-nilai default atau yang diberikan.
12	**self.username = username**	Menginisialisasi atribut **username** dari objek **Lg** dengan nilai parameter **username**.

No	Code	Penjelasan
13	**self.password = password**	Menginisialisasi atribut **password** dari objek **Lg** dengan nilai parameter **password**.
16	**def buatakun():**	Mendefinisikan fungsi **buatakun** yang bertujuan untuk membuat akun baru pada antarmuka pengguna.
18	**signup_window = tk.Toplevel(root)**	Membuat jendela baru untuk proses pendaftaran (**sign-up**) sebagai child window dari root window utama.
19	**signup_window.title("Sign Up")**	Menetapkan judul jendela sign-up yang ditampilkan kepada pengguna.
20	**signup_window.geometry("300x150")**	Menetapkan ukuran geometri jendela sign-up agar sesuai dengan kebutuhan aplikasi.
23	**username_label = ttk.Label(signup_window, text="Username:", font=cambria_font)**	Membuat label dengan teks "Username:" untuk memberikan petunjuk kepada pengguna.
24	**username_label.grid(row=0, column=0, sticky=tk.W)**	Menempatkan label username di grid jendela sign-up dan mengatur agar melekat di sebelah barat (**West**).
25	**username_entry = ttk.Entry(signup_window)**	Membuat widget entry (input) untuk pengguna memasukkan username baru.
26	**username_entry.grid(row=0, column=1)**	Menempatkan entry widget username di grid jendela sign-up.
29	**password_label = ttk.Label(signup_window, text="Password:", font=cambria_font)**	Membuat label dengan teks "Password:" untuk memberikan petunjuk kepada pengguna.
30	**password_label.grid(row=1, column=0, sticky=tk.W)**	Menempatkan label password di grid jendela sign-up dan mengatur agar melekat di sebelah barat (**West**).
31	**password_entry = ttk.Entry(signup_window, show="*")**	Membuat widget entry untuk pengguna memasukkan password baru dengan karakter tersembunyi.
32	**password_entry.grid(row=1, column=1)**	Menempatkan entry widget password di grid jendela sign-up.

No	Code	Penjelasan
35	**signup_button = ttk.Button(signup_window, text="Sign Up", command=lambda: save_signup(username_entry.get(), password_entry.get(), signup_window))**	Membuat tombol sign-up dengan teks "Sign Up" dan menetapkan fungsi callback menggunakan **lambda**.
36	**signup_button.grid(row=2, column=1, pady=10)**	Menempatkan tombol sign-up di grid jendela dengan memberikan sedikit padding pada bagian bawah (**pady**).

Penjelasan Code Line 38 – 74

No	Code	Penjelasan
38	**def save_signup(username, password, signup_window):**	Mendefinisikan fungsi **save_signup** untuk menyimpan data sign-up ke dalam file.
39	**with open('login.txt', 'a') as file:**	Membuka file 'login.txt' dalam mode append ('**a**') untuk menambahkan data baru ke file.
40	**file.write(f"{username},{password}\n")**	Menulis data username dan password yang baru didaftarkan ke dalam file, dipisahkan oleh koma dan diakhiri dengan newline.
41	**print("\nAkun berhasil dibuat!")**	Menampilkan pesan ke konsol bahwa akun berhasil dibuat.
42	**signup_window.destroy()**	Menutup jendela sign-up setelah data berhasil disimpan.
45	**def check_login():**	Mendefinisikan fungsi **check_login** untuk memeriksa login.
46	**lg = loginakun()**	Memanggil fungsi **loginakun** untuk mendapatkan objek **Lg** yang berisi informasi login.
47	**if lg is not None:**	Memeriksa apakah login berhasil dengan mengecek apakah objek **lg** tidak **None**.

No	Code	Penjelasan
48	**messagebox.showinfo("Login Success", "Welcome, " + lg.username + "!")**	Menampilkan dialog informasi bahwa login berhasil dengan menyertakan nama pengguna.
49	**root.destroy()**	Menutup root window setelah login berhasil.
52	**def loginakun():**	Mendefinisikan fungsi **loginakun** untuk melakukan proses login.
53	**lg = None**	Menginisialisasi variabel **lg** dengan **None** untuk menandakan bahwa login belum berhasil.
54	**with open('login.txt', 'r') as file:**	Membuka file 'login.txt' dalam mode baca ('**r**') untuk membaca data login yang sudah tersimpan.
55	**username = simpledialog.askstring("Input", "Masukkan username:")**	Meminta input username dari pengguna menggunakan dialog sederhana.
56	**password = simpledialog.askstring("Input", "Masukkan password:")**	Meminta input password dari pengguna menggunakan dialog sederhana.
58	**for line in file:**	Melakukan iterasi melalui setiap baris dalam file untuk mencocokkan dengan data yang tersimpan.
59	**if ',' in line:**	Memeriksa apakah baris tersebut mengandung karakter koma (',').
60	**stored_username, stored_password = line.strip().split(',', 1)**	Memisahkan nilai username dan password yang tersimpan dalam file.
61	**if username == stored_username and password == stored_password:**	Membandingkan input pengguna dengan data yang tersimpan dalam file.
62	**lg = Lg(username, password)**	Jika cocok, membuat objek **Lg** dengan informasi login dan menghentikan iterasi.
63	**print("\nLogin berhasil!")**	Menampilkan pesan ke konsol bahwa login berhasil.
64	**break**	Menghentikan loop setelah login berhasil ditemukan.

No	Code	Penjelasan
67	**if lg is None:**	Memeriksa apakah objek **lg** masih **None**, menandakan bahwa login tidak berhasil.
68	**print("\nUsername atau password salah!")**	Menampilkan pesan ke konsol bahwa username atau password yang dimasukkan salah.
70	**return lg**	Mengembalikan objek **lg**, yang berisi informasi login jika login berhasil, atau **None** jika login tidak berhasil.

Penjelasan Line 76 – 114.

No	Code	Penjelasan
72	**def close_application():**	Mendefinisikan fungsi **close_application** untuk menutup aplikasi.
73	**confirm = messagebox.askokcancel("Konfirmasi", "Apakah Anda yakin ingin keluar dari aplikasi?")**	Menampilkan dialog konfirmasi menggunakan **messagebox** dan mendapatkan respons dari pengguna.
74	**if confirm:**	Memeriksa apakah pengguna mengklik tombol OK dalam dialog konfirmasi.
75	**sys.exit()**	Menutup aplikasi menggunakan **sys.exit()**.
78	**root = tk.Tk()**	Membuat instance objek tkinter sebagai root window aplikasi.
81	**root.title("Aplikasi Pencatatan Deadline Tugas")**	Menambahkan judul pada root window aplikasi.
84	**root.title("Login")**	Menambahkan judul pada root window aplikasi (tidak sesuai dengan judul sebelumnya).
87	**frame_login = tk.Frame(root, padx=20, pady=20)**	Membuat frame **frame_login** sebagai container untuk elemen-elemen yang berhubungan dengan proses login.
88	**frame_login.pack()**	Menampilkan frame login di root window.
91	**app_title_label = ttk.Label(frame_login,**	Membuat label untuk judul aplikasi pada frame login.

No	Code	Penjelasan
	text="Aplikasi Pencatatan Deadline Tugas", font=cambria_font)	
92	**app_title_label.grid(row=0, column=0, columnspan=2, pady=10)**	Menempatkan label judul aplikasi di grid frame login dengan mengambil dua kolom dan memberikan padding pada bagian bawah.
95	**login_button = tk.Button(frame_login, text="Login", command=check_login, font=cambria_font)**	Membuat tombol untuk melakukan proses login dengan fungsi callback **check_login**.
96	**login_button.grid(row=2, column=0, padx=5, pady=10)**	Menempatkan tombol login di grid frame login dengan memberikan padding pada bagian kiri, kanan, dan bawah.
99	**signup_button = tk.Button(frame_login, text="Sign Up", command=buatakun, font=cambria_font)**	Membuat tombol untuk membuat akun baru dengan fungsi callback **buatakun**.
100	**signup_button.grid(row=2, column=1, padx=5, pady=10)**	Menempatkan tombol sign-up di grid frame login dengan memberikan padding pada bagian kanan, kiri, dan bawah.
103	**def disable_event():**	Mendefinisikan fungsi **disable_event** untuk menangani event saat tombol close pada window login ditekan.
104	**close_application()**	Memanggil fungsi **close_application** saat event close dihandle.
106	**root.protocol("WM_DELETE_WINDOW", disable_event)**	Menghubungkan event close window dengan fungsi **disable_event**.
109	**root.mainloop()**	Menjalankan event loop tkinter untuk menampilkan GUI dan menunggu interaksi pengguna.

Penjelasan Line 116 - 151.

No	Code	Penjelasan
110	**data_array = [...]**	Mendefinisikan variabel **data_array** sebagai list yang

No	Code	Penjelasan
		berisi beberapa dictionary. Setiap dictionary merepresentasikan data tugas dengan keterangan dan tanggal.
112	**def boyer_moore_horspool_search(pattern, text):**	Mendefinisikan fungsi **boyer_moore_horspool_search** untuk melakukan pencarian pola dengan algoritma *Boyer-Moore Horspool*.
113	**m = len(pattern)**	Menghitung panjang pola yang akan dicari.
114	**n = len(text)**	Menghitung panjang teks tempat pencarian akan dilakukan.
115	**pattern = pattern.lower()**	Mengubah pola dan teks menjadi huruf kecil untuk memastikan pencarian bersifat case-insensitive.
116	**text = text.lower()**	Mengubah pola dan teks menjadi huruf kecil untuk memastikan pencarian bersifat case-insensitive.
117	**last_occurrence = {pattern[i]: i for i in range(m)}**	Membuat dictionary **last_occurrence** yang menyimpan indeks terakhir dari setiap karakter dalam pola.
118	**i = m - 1**	Inisialisasi indeks **i** sebagai indeks terakhir dalam pola.
119	**j = m - 1**	Inisialisasi indeks **j** sebagai indeks terakhir dalam teks.
121	**while j < n:**	Melakukan loop sampai indeks **j** mencapai akhir teks.
122	**if pattern[i] == text[j]:**	Membandingkan karakter pada indeks **i** dalam pola dengan karakter pada indeks **j** dalam teks.
123	**if i == 0:**	Memeriksa apakah indeks **i** sudah mencapai awal pola.
124	**return j**	Jika ya, mengembalikan indeks **j** sebagai posisi awal ditemukannya pola dalam teks.
126	**else:**	Jika belum mencapai awal pola, melakukan penyesuaian indeks.

No	Code	Penjelasan
127	**i -= 1**	Mengurangkan indeks **i** untuk memeriksa karakter berikutnya.
128	**j -= 1**	Mengurangkan indeks **j** untuk memeriksa karakter berikutnya.
130	**else:**	Jika karakter pada indeks **i** dalam pola tidak sama dengan karakter pada indeks **j** dalam teks, melakukan penyesuaian posisi.
131	**last_occ = last_occurrence.get(text[j], -1)**	Mendapatkan indeks terakhir dari karakter pada indeks **j** dalam teks dari dictionary **last_occurrence**.
132	**j = j + m - min(i, 1 + last_occ)**	Menyesuaikan indeks **j** berdasarkan indeks terakhir karakter pada indeks **j** dalam teks dan indeks **i**.
133	**i = m - 1**	Mengembalikan indeks **i** ke posisi terakhir dalam pola.
135	**return None**	Jika pola tidak ditemukan dalam teks, mengembalikan **None**.

Penjelasan Line 153 - 195

No	Code	Penjelasan
136	**def search_phrase_in_text(phrase, text):**	Mendefinisikan fungsi **search_phrase_in_text** untuk mencari apakah suatu frasa (phrase) terdapat dalam teks (text).
137	**return phrase.lower() in text.lower()**	Mengembalikan nilai True jika frasa dalam huruf kecil ditemukan dalam teks (case-insensitive), dan False jika tidak.
139	**def boyer_moore_horspool_search_name (name, data_array):**	Mendefinisikan fungsi **boyer_moore_horspool_search_name** untuk mencari nama dalam **data_array** menggunakan algoritma *Boyer-Moore Horspool*.
140	**name = name.lower()**	Mengonversi nama pencarian menjadi huruf kecil.
142	**for i, item in enumerate(data_array):**	Melakukan iterasi melalui setiap item dalam **data_array** beserta indeksnya.

No	Code	Penjelasan
143	**keterangan_lower = item["keterangan"].lower()**	Mengonversi keterangan setiap item menjadi huruf kecil.
145	**if search_phrase_in_text(name, keterangan_lower):**	Memanggil fungsi **search_phrase_in_text** untuk mencari nama dalam keterangan item.
146	**return i**	Mengembalikan indeks item jika nama ditemukan dalam keterangan.
148	**return None**	Jika nama tidak ditemukan dalam seluruh **data_array**, mengembalikan None.
151	**def show_result(result_index):**	Mendefinisikan fungsi **show_result** untuk menampilkan hasil pencarian.
152	**if result_index is not None:**	Memeriksa apakah indeks hasil pencarian tidak None (artinya nama ditemukan).
153	**message = f"Tanggal deadline {data_array[result_index]['tanggal']} "**	Membuat pesan yang menyertakan tanggal deadline dari item yang ditemukan.
155	**message = "Agenda Tugas yang Anda cari tidak ditemukan."**	Jika nama tidak ditemukan, membuat pesan bahwa agenda tidak ditemukan.
156	**result_label.configure(text=message)**	Mengkonfigurasi teks pada **result_label** sesuai dengan pesan yang telah dibuat.
159	**def add_data():**	Mendefinisikan fungsi **add_data** untuk menambahkan data baru ke dalam array dan memperbarui tampilan tabel.
160	**keterangan = keterangan_entry.get()**	Mengambil nilai dari entry widget **keterangan_entry**.
161	**tanggal = tanggal_entry.get()**	Mengambil nilai dari entry widget **tanggal_entry**.
162	**data_array.append({"tanggal": tanggal, "keterangan": keterangan})**	Menambahkan dictionary baru ke dalam **data_array** dengan keterangan dan tanggal yang diambil dari entry widgets.
163	**update_table()**	Memanggil fungsi **update_table** untuk memperbarui tampilan tabel.

No	Code	Penjelasan
166	**def delete_data():**	Mendefinisikan fungsi **delete_data** untuk menghapus data dari array dan memperbarui tampilan tabel.
167	**item = table.selection()[0]**	Mengambil item yang sedang dipilih pada tabel.
168	**index = int(table.index(item))**	Mengambil indeks dari item yang dipilih.
169	**del data_array[index]**	Menghapus item pada indeks yang diambil dari **data_array**.
170	**update_table()**	Memanggil fungsi **update_table** untuk memperbarui tampilan tabel.
173	**def update_table():**	Mendefinisikan fungsi **update_table** untuk memperbarui tampilan tabel.
174	**table.delete(*table.get_children())**	Menghapus semua item dari tabel.
175	**for i, data in enumerate(data_array):**	Melakukan iterasi melalui setiap item dalam **data_array** beserta indeksnya.
176	**table.insert("", "end", text=i+1, values=(data["keterangan"], data["tanggal"], i))**	Menyisipkan item baru ke dalam tabel dengan nilai yang sesuai.

Penjelasan Line 197 - 227

No	Code	Penjelasan
177	**def edit_data(index):**	Mendefinisikan fungsi **edit_data** untuk mengedit data pada array dan memperbarui tampilan tabel.
178	**edit_window = tk.Toplevel(root)**	Membuat instance objek **edit_window** sebagai top-level window untuk mengedit data.
179	**edit_window.title("Edit Data")**	Menetapkan judul window edit menjadi "Edit Data".
180	**edit_window.geometry("300x150")**	Menetapkan lebar dan tinggi window edit menjadi 300x150 piksel, sesuai dengan window "View Data".
183	**keterangan_label = ttk.Label(edit_window, text="Agenda Tugas:")**	Membuat label untuk input keterangan pada window edit.

No	Code	Penjelasan
184	**keterangan_label.grid(row=0, column=0, sticky=tk.W)**	Menempatkan label keterangan pada grid window edit.
185	**keterangan_entry = ttk.Entry(edit_window)**	Membuat entry widget untuk input keterangan pada window edit.
186	**keterangan_entry.insert(0, data_array[index]["keterangan"])**	Mengisi entry widget keterangan dengan nilai keterangan yang ada pada array pada indeks tertentu.
187	**keterangan_entry.grid(row=0, column=1, padx=5, pady=5)**	Menempatkan entry widget keterangan pada grid window edit.
190	**tanggal_label = ttk.Label(edit_window, text="Tanggal deadline:")**	Membuat label untuk input tanggal pada window edit.
191	**tanggal_label.grid(row=1, column=0, sticky=tk.W)**	Menempatkan label tanggal pada grid window edit.
192	**tanggal_entry = ttk.Entry(edit_window)**	Membuat entry widget untuk input tanggal pada window edit.
193	**tanggal_entry.insert(0, data_array[index]["tanggal"])**	Mengisi entry widget tanggal dengan nilai tanggal yang ada pada array pada indeks tertentu.
194	**tanggal_entry.grid(row=1, column=1, padx=5, pady=5)**	Menempatkan entry widget tanggal pada grid window edit.
197	**save_button = ttk.Button(edit_window, text="Save", command=lambda: save_changes(index, keterangan_entry.get(), tanggal_entry.get(), edit_window))**	Membuat tombol "Save" untuk menyimpan perubahan.
198	**save_button.grid(row=2, column=1, pady=10)**	Menempatkan tombol "Save" pada grid window edit.
201	**def save_changes(index, keterangan, tanggal, edit_window):**	Mendefinisikan fungsi **save_changes** untuk menyimpan perubahan ke dalam **data_array** dan memperbarui tampilan tabel.
202	**data_array[index]["keterangan"] = keterangan**	Memperbarui nilai keterangan pada item dengan indeks tertentu dalam **data_array**.
203	**data_array[index]["tanggal"] = tanggal**	Memperbarui nilai tanggal pada item dengan indeks tertentu dalam **data_array**.

No	Code	Penjelasan
204	**edit_window.destroy()**	Menutup window edit setelah perubahan disimpan.
205	**update_table()**	Memanggil fungsi **update_table** untuk memperbarui tampilan tabel setelah perubahan.

Penjelasan Line 229 - 265

No.	Code	Penjelasan
1	**root.title("LIST DEADLINE TUGAS")**	Memberikan judul pada GUI dengan teks "LIST DEADLINE TUGAS".
2	**frame_table = ttk.Frame(root, padding="20")**	Membuat frame **frame_table** dengan padding sebesar 20 dan parent window root. Frame ini akan digunakan untuk menampung tabel dan mengatur tata letak.
3	**frame_table.pack(fill="both", expand=True)**	Menempatkan **frame_table** di dalam root window dengan mengisi baik ke arah horizontal (x) maupun vertikal (y) dan diperluas agar memenuhi area yang tersedia.
4	**frame_input = ttk.Frame(root, padding="10")**	Membuat frame **frame_input** dengan padding sebesar 10 dan parent window root. Frame ini akan digunakan untuk menampung input keterangan dan tanggal.
5	**frame_input.pack(fill="x", expand=True)**	Menempatkan **frame_input** di dalam root window dengan diperluas hanya ke arah horizontal (x) agar sesuai dengan lebar GUI.
6	**frame_button = ttk.Frame(root, padding="10")**	Membuat frame **frame_button** dengan padding sebesar 10 dan parent window root. Frame ini akan digunakan untuk menampung tombol untuk menambah dan menghapus data.
7	**frame_button.pack(fill="x", expand=True)**	Menempatkan **frame_button** di dalam root window dengan diperluas hanya ke arah horizontal (x) agar sesuai dengan lebar GUI.
8	**table = ttk.Treeview(frame_table, columns=("col1"))**	Membuat objek **table** dari kelas **ttk.Treeview** di dalam frame **frame_table** dengan satu kolom bernama "col1".

No.	Code	Penjelasan
9	**table.heading("#0", text="No.")**	Menentukan teks pada header kolom 0 (kolom indeks) menjadi "No.".
10	**table.heading("col1", text="Keterangan")**	Menentukan teks pada header kolom "col1" menjadi "Keterangan".
11	**table.pack(fill="both", expand=True)**	Menempatkan **table** di dalam frame **frame_table** dengan mengisi baik ke arah horizontal (x) maupun vertikal (y) dan diperluas agar memenuhi area yang tersedia.
12	**keterangan_label = ttk.Label(frame_input, text="Agenda Tugas :")**	Membuat label "Agenda Tugas :" di dalam frame **frame_input**.
13	**keterangan_label.pack(side="left")**	Menempatkan **keterangan_label** di sisi kiri frame **frame_input**.
14	**keterangan_entry = ttk.Entry(frame_input)**	Membuat objek **keterangan_entry** dari kelas **ttk.Entry** di dalam frame **frame_input** untuk memasukkan keterangan.
15	**keterangan_entry.pack(side="left")**	Menempatkan **keterangan_entry** di sisi kiri frame **frame_input**.
16	**tanggal_label = ttk.Label(frame_input, text="Tanggal deadline:")**	Membuat label "Tanggal deadline:" di dalam frame **frame_input**.
17	**tanggal_label.pack(side="left")**	Menempatkan **tanggal_label** di sisi kiri frame **frame_input**.
18	**tanggal_entry = ttk.Entry(frame_input)**	Membuat objek **tanggal_entry** dari kelas **ttk.Entry** di dalam frame **frame_input** untuk memasukkan tanggal deadline.
19	**tanggal_entry.pack(side="left")**	Menempatkan **tanggal_entry** di sisi kiri frame **frame_input**.
20	**add_button = ttk.Button(frame_button, text="Tambah Data", command=add_data)**	Membuat tombol "Tambah Data" di dalam frame **frame_button** dengan memanggil fungsi **add_data** saat tombol diklik.
21	**add_button.pack(side="left")**	Menempatkan tombol "Tambah Data" di sisi kiri frame **frame_button**.
22	**delete_button = ttk.Button(frame_button, text="Hapus Data", command=delete_data)**	Membuat tombol "Hapus Data" di dalam frame **frame_button** dengan memanggil fungsi **delete_data** saat tombol diklik.

No.	Code	Penjelasan
23	**delete_button.pack(side="left", padx=10)**	Menempatkan tombol "Hapus Data" di sisi kiri frame **frame_button** dengan memberikan padding pada sumbu x sebesar 10.

Penjelasan Line 229 - 265

No	Code	Penjelasan
206	**root = tk.Tk()**	Membuat instance objek tkinter dan menyimpannya dalam variabel **root**.
209	**root.title("LIST DEADLINE TUGAS")**	Menetapkan judul GUI menjadi "LIST DEADLINE TUGAS".
212	**frame_table = ttk.Frame(root, padding="20")**	Membuat frame (**frame_table**) untuk menampung tabel dan memberikan padding sebesar 20 piksel.
213	**frame_table.pack(fill="both", expand=True)**	Meletakkan frame tabel ke dalam GUI, mengisi ruang di semua arah dan memperluasnya sejauh mungkin.
214	**frame_input = ttk.Frame(root, padding="10")**	Membuat frame (**frame_input**) untuk menampung elemen input dan memberikan padding sebesar 10 piksel.
215	**frame_input.pack(fill="x", expand=True)**	Meletakkan frame input ke dalam GUI, mengisi ruang secara horizontal dan memperluasnya sejauh mungkin.
216	**frame_button = ttk.Frame(root, padding="10")**	Membuat frame (**frame_button**) untuk menampung tombol dan memberikan padding sebesar 10 piksel.
217	**frame_button.pack(fill="x", expand=True)**	Meletakkan frame tombol ke dalam GUI, mengisi ruang secara horizontal dan memperluasnya sejauh mungkin.
220	**table = ttk.Treeview(frame_table, columns=("col1"))**	Membuat objek **Treeview** sebagai tabel dengan satu kolom ("col1").
221	**table.heading("#0", text="No.")**	Menetapkan teks "No." sebagai heading kolom indeks pada tabel.
222	**table.heading("col1", text="Keterangan")**	Menetapkan teks "Keterangan" sebagai heading kolom data pada tabel.
224	**table.pack(fill="both", expand=True)**	Menampilkan tabel ke dalam GUI, mengisi ruang di semua arah dan memperluasnya sejauh mungkin.
227	**keterangan_label = ttk.Label(frame_input, text="Agenda Tugas :")**	Membuat label "Agenda Tugas :" untuk input keterangan.

No	Code	Penjelasan
228	**keterangan_label.pack(side="left")**	Menempatkan label keterangan di sisi kiri frame input.
229	**keterangan_entry = ttk.Entry(frame_input)**	Membuat entry widget untuk input keterangan.
230	**keterangan_entry.pack(side="left")**	Menempatkan entry widget keterangan di sisi kiri frame input.
231	**tanggal_label = ttk.Label(frame_input, text="Tanggal deadline:")**	Membuat label "Tanggal deadline:" untuk input tanggal.
232	**tanggal_label.pack(side="left")**	Menempatkan label tanggal di sisi kiri frame input.
233	**tanggal_entry = ttk.Entry(frame_input)**	Membuat entry widget untuk input tanggal.
234	**tanggal_entry.pack(side="left")**	Menempatkan entry widget tanggal di sisi kiri frame input.
237	**add_button = ttk.Button(frame_button, text="Tambah Data", command=add_data)**	Membuat tombol "Tambah Data" untuk menambahkan data baru dan menetapkan fungsi **add_data** sebagai command.
238	**add_button.pack(side="left")**	Menempatkan tombol "Tambah Data" di sisi kiri frame tombol.
239	**delete_button = ttk.Button(frame_button, text="Hapus Data", command=delete_data)**	Membuat tombol "Hapus Data" untuk menghapus data dan menetapkan fungsi **delete_data** sebagai command.
240	**delete_button.pack(side="left", padx=10)**	Menempatkan tombol "Hapus Data" di sisi kiri frame tombol, dengan jarak padding horizontal sebesar 10 piksel.

Penjelasan Line 267 – 311

No	Code	Penjelasan
242	**def edit_selected_data():**	Mendefinisikan fungsi **edit_selected_data** untuk mengedit data yang dipilih dari tabel.
243	**selected_item = table.selection()**	Mengambil item yang sedang dipilih pada tabel.
244	**if selected_item:**	Memeriksa apakah ada item yang dipilih.
245	**index = table.index(selected_item)**	Mengambil indeks dari item yang dipilih.

No	Code	Penjelasan
246	**edit_data(index)**	Memanggil fungsi **edit_data** untuk mengedit data pada indeks tertentu.
247	**else:**	Jika tidak ada item yang dipilih, menampilkan pesan informasi.
248	**messagebox.showinfo("Info", "Pilih data yang akan diedit.")**	Menampilkan dialog informasi dengan pesan "Pilih data yang akan diedit.".
251	**edit_button = ttk.Button(frame_button, text="Edit Data", command=edit_selected_data)**	Membuat tombol "Edit Data" untuk memanggil fungsi **edit_selected_data** sebagai command.
252	**edit_button.pack(side="left", padx=10)**	Menempatkan tombol "Edit Data" di sisi kiri frame tombol dengan jarak padding horizontal sebesar 10 piksel.
254	**def view_selected_data():**	Mendefinisikan fungsi **view_selected_data** untuk melihat data yang dipilih dari tabel.
255	**selected_item = table.selection()**	Mengambil item yang sedang dipilih pada tabel.
256	**if selected_item:**	Memeriksa apakah ada item yang dipilih.
257	**index = table.index(selected_item)**	Mengambil indeks dari item yang dipilih.
258	**view_data(index)**	Memanggil fungsi **view_data** untuk melihat data pada indeks tertentu.
259	**else:**	Jika tidak ada item yang dipilih, menampilkan pesan informasi.
260	**messagebox.showinfo("Info", "Pilih data yang akan dilihat.")**	Menampilkan dialog informasi dengan pesan "Pilih data yang akan dilihat.".
263	**def view_data(index):**	Mendefinisikan fungsi **view_data** untuk melihat data pada indeks tertentu.
264	**view_window = tk.Toplevel(root)**	Membuat instance objek **view_window** sebagai top-level window untuk melihat data.
265	**view_window.title("Lihat Data")**	Menetapkan judul window "Lihat Data".
266	**view_window.geometry("300x150")**	Menetapkan lebar dan tinggi window sesuai kebutuhan.

No	Code	Penjelasan
269	**keterangan_label = ttk.Label(view_window, text="Agenda Tugas:")**	Membuat label "Agenda Tugas:" untuk menampilkan keterangan.
270	**keterangan_label.grid(row=0, column=0, sticky=tk.W)**	Menempatkan label keterangan pada grid window lihat data.
271	**keterangan_value = ttk.Label(view_window, text=data_array[index]["keterangan"])**	Membuat label untuk menampilkan nilai keterangan dari item yang dipilih.
272	**keterangan_value.grid(row=0, column=1, padx=5, pady=5)**	Menempatkan label nilai keterangan pada grid window lihat data.
275	**tanggal_label = ttk.Label(view_window, text="Tanggal deadline:")**	Membuat label "Tanggal deadline:" untuk menampilkan tanggal.
276	**tanggal_label.grid(row=1, column=0, sticky=tk.W)**	Menempatkan label tanggal pada grid window lihat data.
277	**tanggal_value = ttk.Label(view_window, text=data_array[index]["tanggal"])**	Membuat label untuk menampilkan nilai tanggal dari item yang dipilih.
278	**tanggal_value.grid(row=1, column=1, padx=5, pady=5)**	Menempatkan label nilai tanggal pada grid window lihat data.
281	**view_button = ttk.Button(frame_button, text="Lihat Data", command=view_selected_data)**	Membuat tombol "Lihat Data" untuk memanggil fungsi **view_selected_data** sebagai command.
282	**view_button.pack(side="left", padx=10)**	Menempatkan tombol "Lihat Data" di sisi kiri frame tombol dengan jarak padding horizontal sebesar 10 piksel.
285	**def search_data():**	Mendefinisikan fungsi **search_data** untuk mencari data berdasarkan input pengguna menggunakan fungsi pencarian **boyer_moore_horspool_search_name**.
286	**result_index = boyer_moore_horspool_search_name(input_entry.get(), data_array)**	Menggunakan fungsi pencarian **boyer_moore_horspool_search_name** untuk mencari indeks data yang cocok dengan input pengguna.
287	**show_result(result_index)**	Menampilkan hasil pencarian menggunakan fungsi **show_result**.

Penjelasan Line Code 313 - 339

No	Code	Penjelasan
288	**input_label = ttk.Label(frame_input, text="Cari Agenda :")**	Membuat label "Cari Agenda :" untuk input pencarian.
289	**input_label.pack(side="left")**	Menempatkan label pencarian di sisi kiri frame input.
290	**input_entry = ttk.Entry(frame_input)**	Membuat entry widget untuk input pencarian.
291	**input_entry.pack(side="left")**	Menempatkan entry widget pencarian di sisi kiri frame input.
292	**search_button = ttk.Button(frame_input, text="Cari", command=search_data)**	Membuat tombol "Cari" untuk memanggil fungsi **search_data** sebagai command.
293	**search_button.pack(side="left")**	Menempatkan tombol "Cari" di sisi kiri frame input.
296	**result_label = ttk.Label(root, text="")**	Membuat label **result_label** untuk menampilkan hasil pencarian.
297	**result_label.pack()**	Menempatkan label hasil pencarian di dalam root.
300	**update_table()**	Memanggil fungsi **update_table** untuk memperbarui tampilan tabel dengan data array.
303	**exit_application():**	Mendefinisikan fungsi **exit_application** untuk menutup aplikasi dengan konfirmasi.
304	**confirm = messagebox.askokcancel("Konfirmasi", "Apakah Anda yakin ingin keluar dari aplikasi?")**	Menampilkan dialog konfirmasi dengan pertanyaan "Apakah Anda yakin ingin keluar dari aplikasi?".
305	**if confirm:**	Memeriksa apakah pengguna mengkonfirmasi keluar.
306	**root.destroy()**	Menghancurkan objek root dan menutup aplikasi.
309	**exit_button = ttk.Button(frame_button, text="Exit", command=exit_application)**	Membuat tombol "Exit" untuk memanggil fungsi **exit_application** sebagai command.
310	**exit_button.pack(side="left", padx=10)**	Menempatkan tombol "Exit" di sisi kiri frame tombol dengan jarak padding horizontal sebesar 10 piksel.

No	Code	Penjelasan
313	**root.mainloop()**	Menjalankan GUI dan memulai event loop tkinter.

3.4 Dokumentasi Video Running Program

Berikut adalah link google drive dokumentasi video saat menjalankan program dari awal *login* hingga *logout*.

https://drive.google.com/file/d/1SplyQpQnel-PCSINljuIIu-_mjLT5Myu/view?usp=drive_link

BAB IV
Kesimpulan

Berdasarkan pembahasan, maka diperoleh kesimpulan sebagai berikut.

1. Algoritma *Boyer-Moore Horspool* Search merupakan algoritma pencarian pola yang efisien dan cepat. Cara kerjanya didasarkan pada konsep pemindaian dari kanan ke kiri pada teks yang dicari. Algoritma ini memanfaatkan informasi kemunculan terakhir dari setiap karakter dalam pola untuk memutuskan langkah pemindaiannya. Dengan kata lain, algoritma ini cenderung meminimalkan jumlah karakter yang harus diperiksa dalam proses pencarian, sehingga meningkatkan efisiensi, terutama pada teks yang panjang.
2. Dalam aplikasi pencatatan deadline tugas mahasiswa, algoritma *Boyer-Moore Horspool* digunakan untuk melakukan pencarian keterangan tugas berdasarkan kata kunci yang dimasukkan pengguna. Penerapan algoritma ini memberikan kelebihan dalam efisiensi waktu, terutama ketika pengguna ingin mencari informasi spesifik tentang tugas dalam daftar yang mungkin sangat panjang. Dalam implementasinya, algoritma *Boyer-Moore Horspool* digunakan untuk mencocokkan kata kunci pencarian dengan keterangan tugas yang tersimpan dalam *data_array.* Dengan menggunakan algoritma ini, aplikasi mampu memberikan hasil pencarian dengan cepat, bahkan dalam kondisi di mana daftar tugas mahasiswa memiliki jumlah entri yang signifikan. Hasil pencarian tersebut kemudian ditampilkan kepada pengguna melalui label pada antarmuka grafis.

DAFTAR PUSTAKA

GeeksforGeeks. 2012. *Boyer Moore Algorithm for Pattern Searching*. URL: https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/. Diakses tanggal 27 Desember 2023.

Sagita, V. dan Prasetiyowati, M. I. 2013. Studi Perbandingan Implementasi Algoritma Boyer-Moore, Turbo Boyer-Moore, dan Tuned Boyer-Moore dalam Pencarian String. *Ultimatics: Jurnal Teknik Informatika*. 5(1): 31-37.

Utomo, W. 2020. *Algoritma boyer moore*. URL: https://www.academia.edu/34831185/Algoritma_boyer_moore. Diakses tanggal 16 Desember 2023.

LAMPIRAN

Berikut adalah lampiran link google drive yang berisi *source code,* laporan, dan video dokumentasi.
https://drive.google.com/drive/folders/1d8ka6o4LoFMx8d5OAsZyT62M4qsVaqVN?usp=drive_link.

Berikut ini adalah link video youtube yang berisi penjelasan detail *running program* dan penjelasan source code serta alur kerja algoritma *Boyer Moore Horspool.*

https://youtu.be/aSBW8_VMpSU