

# **Laporan Tugas Besar III**

## **IF2211 Strategi Algoritma**

### **Semester II Tahun 2021/2022**

**Penerapan String Matching dan Regular Expression  
dalam DNA Pattern Matching**



**Anggota:**

Muhammad Rakha Athaya	13520108
Mohamad Daffa Argakoesoemah	13520118
Frederik Imanuel Louis	13520163

**Institut Teknologi Bandung**  
**2022**

# Daftar Isi

<b>Daftar Isi</b>	<b>1</b>
<b>Bab I</b>	
<b>Deskripsi Tugas</b>	<b>3</b>
<b>Bab II</b>	
<b>Landasan Teori</b>	<b>7</b>
2.1 Algoritma	7
2.1.1 Algoritma Knuth–Morris–Pratt (KMP)	7
2.1.2 Algoritma Boyer–Moore	8
2.1.3 Algoritma Longest Common Subsequence (LCS)	9
2.1.4 Regular Expression (Regex)	9
2.2 Aplikasi Web	10
<b>Bab III</b>	
<b>Analisis Pemecahan Masalah</b>	<b>11</b>
3.1 Penyelesaian Masalah Setiap Fitur	11
3.1.1 Fitur Penambahan Penyakit Baru	11
3.1.2 Fitur Prediksi Penyakit Seseorang	11
3.1.3 Fitur Pencarian Riwayat Tes	12
3.1.4 Fitur Tingkat Kemiripan DNA	12
3.2 Fitur Fungsional dan Arsitektur Aplikasi Web	12
<b>Bab IV</b>	
<b>Implementasi dan Pengujian</b>	<b>14</b>
4.1 Spesifikasi Teknis Aplikasi	14
4.1.1 Frontend	14
4.1.2 Backend	14
4.2 Tata Cara Penggunaan Aplikasi	15
4.2.1 Interface	15
4.2.2 Fitur-fitur Aplikasi	17
4.3 Hasil Pengujian	17
4.3.1 Skenario 1	17
4.3.2 Skenario 2	18
4.3.3 Skenario 3	18
4.3.4 Skenario 4	19
4.3.5 Skenario 5	19
4.3.6 Skenario 6	20
4.3.7 Skenario 7	20
4.3.8 Skenario 8	21

4.4 Analisis Hasil Pengujian	21
<b>Bab V</b>	
<b>Kesimpulan dan Saran</b>	<b>23</b>
5.1 Kesimpulan	23
5.2. Saran	23
5.3. Komentar dan Refleksi	23
<b>Daftar Pustaka</b>	<b>24</b>
<b>Link Repository Github Kelompok Malignant</b>	<b>24</b>
<b>Link Aplikasi Web</b>	<b>24</b>
<b>Link Video Demonstrasi</b>	<b>24</b>


# Bab I

## Deskripsi Tugas

Dalam tugas besar ini, Anda diminta untuk membangun sebuah aplikasi DNA Pattern Matching. Dengan memanfaatkan algoritma String Matching dan Regular Expression yang telah anda pelajari di kelas IF2211 Strategi Algoritma, anda diharapkan dapat membangun sebuah aplikasi interaktif untuk mendeteksi apakah seorang pasien mempunyai penyakit genetik tertentu. Hasil prediksi tersebut dapat disimpan pada basis data untuk kemudian dapat ditampilkan berdasarkan query pencarian.

### Fitur-Fitur Aplikasi:

1. Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya (dan dimasukkan ke dalam database).
  - Implementasi input sequence DNA dalam bentuk file.
  - Dilakukan sanitasi input menggunakan regex untuk memastikan bahwa masukan merupakan sequence DNA yang valid (tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, dan tidak ada spasi).
  - Contoh input penyakit:



The image shows a web form titled "Tambahkan Penyakit". It contains two input fields: "Nama Penyakit:" with a placeholder text "penyakit..." and "Sequence DNA:" with a placeholder text "upload file...". Below these fields is a green "Submit" button.

2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.
  - Tes DNA dilakukan dengan menerima input nama pengguna, sequence DNA pengguna, dan nama penyakit yang diuji. Asumsi sequence DNA pengguna > sequence DNA penyakit.
  - Dilakukan sanitasi input menggunakan regex untuk memastikan bahwa masukan merupakan sequence DNA yang valid (tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, tidak ada spasi, dll).

- Pencocokan sequence DNA dilakukan dengan menggunakan algoritma string matching.
- Hasil dari tes DNA berupa tanggal tes, nama pengguna, nama penyakit yang diuji, dan status hasil tes. Contoh: 1 April 2022 - Mhs IF - HIV - False
- Semua komponen hasil tes ini dapat ditampilkan pada halaman web (refer ke poin 3 pada “Fitur-Fitur Aplikasi”) dan disimpan pada sebuah tabel database.
- Contoh tampilan web:

3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.
  - Kolom pencarian dapat menerima masukan dengan struktur: <tanggal\_prediksi><spasi><nama\_penyakit>, contoh “13 April 2022 HIV”. Format penanggalan dibebaskan, jika bisa menerima >1 format lebih baik.
  - Kolom pencarian dapat menerima masukan hanya tanggal ataupun hanya nama penyakit. Fitur ini diimplementasikan menggunakan regex.
  - Contoh ilustrasi:
    - Masukkan tanggal dan nama penyakit

- Masukkan hanya tanggal

A screenshot of a web form with a light blue background. At the top, there is a rounded rectangular input field containing the text "13 April 2022". Below this, there are six horizontal rectangular boxes, each containing a number followed by a date, a name, a disease, and a boolean value. The boxes are numbered 1 through 6.

No	Date	Name	Disease	Result
1.	13 April 2022	Fulan	Diabetes	True
2.	13 April 2022	Kamal	Sinusitis	False
3.	13 April 2022	Entah	Down Syndrome	False
4.	13 April 2022	Jamal	Polio	True
5.	13 April 2022	Yubai	TBC	True
6.	13 April 2022	Hika	Hepatitis A	False

- Masukkan hanya nama penyakit

A screenshot of a web form with a light blue background. At the top, there is a rounded rectangular input field containing the text "HIV". Below this, there are six horizontal rectangular boxes, each containing a number followed by a date, a name, a disease, and a boolean value. The boxes are numbered 1 through 6.

No	Date	Name	Disease	Result
1.	13 April 2022	Fulan	HIV	True
2.	14 April 2022	Kamal	HIV	False
3.	15 April 2022	Entah	HIV	False
4.	16 April 2022	Jamal	HIV	True
5.	17 April 2022	Yubai	HIV	True
6.	18 April 2022	Hika	HIV	False

4. **(Bonus)** Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA.
- Ketika melakukan tes DNA, terdapat persentase kemiripan DNA dalam hasil tes. Contoh hasil tes: 1 April 2022 - Mhs IF - HIV - 75% - False
  - Perhitungan tingkat kemiripan dapat dilakukan dengan menggunakan Hamming distance, Levenshtein distance, LCS, atau algoritma lainnya (dapat dijelaskan dalam laporan).
  - Tingkat kemiripan DNA dengan nilai lebih dari atau sama dengan 80% dikategorikan sebagai True. Perlu diperhatikan mengimplementasikan atau tidak

mengimplementasikan bonus ini tetap dilakukan pengecekan string matching terlebih dahulu.

- Contoh tampilan:

**Tes DNA**

Nama Pengguna:

Sequence DNA:

Prediksi Penyakit:

---

**Hasil Tes**

<Tanggal> - <pengguna> - <penyakit> - <similarity> - <True/False>

### Spesifikasi Program:

1. Aplikasi berbasis website dengan pembagian Frontend dan Backend yang jelas.
2. Implementasi Backend wajib menggunakan Node.js / Golang, sedangkan Frontend disarankan untuk menggunakan React / Next.js / Vue / Angular. Lihat referensi untuk selengkapnya.
3. Penyimpanan data wajib menggunakan basis data (MySQL / PostgreSQL / MongoDB).
4. Algoritma pencocokan string (KMP dan Boyer-Moore) wajib diimplementasikan pada sisi Backend aplikasi.
5. Informasi yang wajib disimpan pada basis data:
  - a. Jenis Penyakit:
    - i. Nama penyakit
    - ii. Rantai DNA penyusun
  - b. Hasil Prediksi:
    - i. Tanggal prediksi
    - ii. Nama pasien
    - iii. Penyakit prediksi
    - iv. Status terprediksi
6. Jika mengerjakan bonus tingkat kemiripan DNA, simpan hasil tingkat kemiripan tersebut pada basis data.

## Bab II

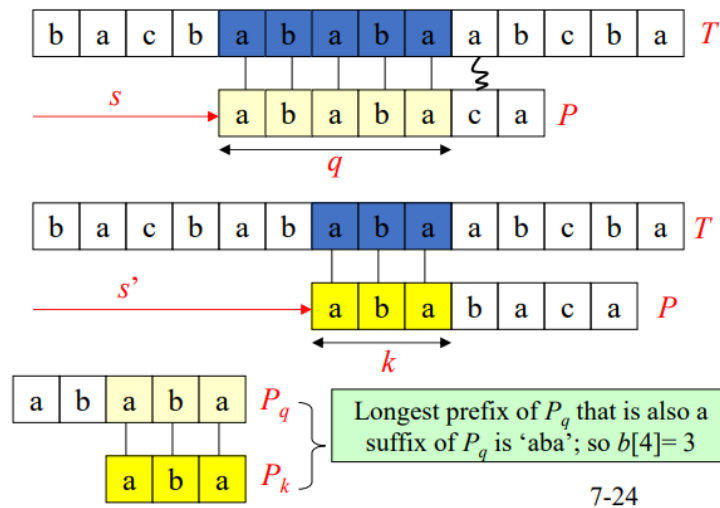
### Landasan Teori

#### 2.1 Algoritma

##### 2.1.1 Algoritma Knuth–Morris–Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma yang dapat digunakan dalam pencocokan string (string matching). Algoritma ini dipublikasikan oleh James H. Morris, Donald Knuth, dan Vaughan Pratt pada tahun 1977 setelah Morris dan Knuth-Pratt menemukannya secara independen. Pada algoritma KMP, pencarian pattern pada teks dicari dengan aturan dari kiri ke kanan seperti algoritma brute force. Namun, Algoritma KMP melakukan pergeseran atau penelusuran secara lebih efektif.

Algoritma KMP melakukan pergeseran ke kiri sebanyak mungkin sehingga langkah pengerjaannya lebih efisien dibandingkan algoritma brute force. Pergeseran pada algoritma KMP dicari menggunakan pencocokan antara prefix terbesar yang juga merupakan suffix.



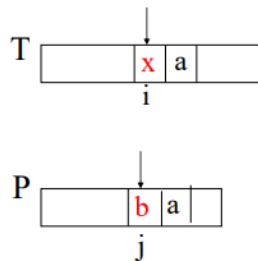
Pada ilustrasi di atas, pencocokan string dimulai dari 'aba' karena 'aba' merupakan prefix terpanjang yang juga merupakan suffix. Kompleksitas Algoritma KMP adalah  $O(m+n)$ .



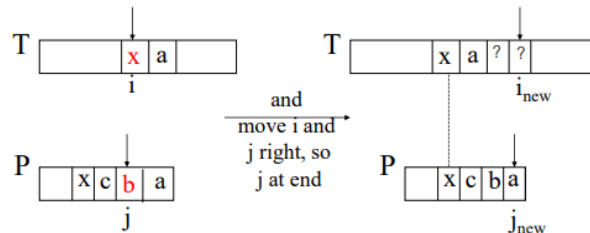
## 2.1.2 Algoritma Boyer–Moore

Algoritma Boyer-Moore adalah algoritma yang dapat digunakan dalam pencocokan string (string matching). Algoritma ini dikembangkan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1977. Algoritma Boyer-Moore menggunakan teknik *looking-glass* dan *character jump* dalam pencocokan stringnya. *Looking-glass technique* adalah teknik yang memeriksa kecocokan string pattern pada teks dengan dimulai dari indeks terakhir pattern. Sementara *character jump technique* merupakan teknik yang diaplikasikan ketika terjadi mismatch antara teks[i] dan pattern[j]. Untuk kasus terburuk, waktu eksekusi algoritma Boyer-Moore adalah  $O(nm+A)$ .

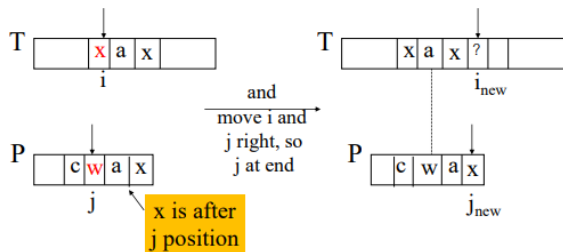
Pada teknik *character-jump*, terdapat 3 kasus yang mungkin terjadi dan diuji secara berurutan, yaitu:



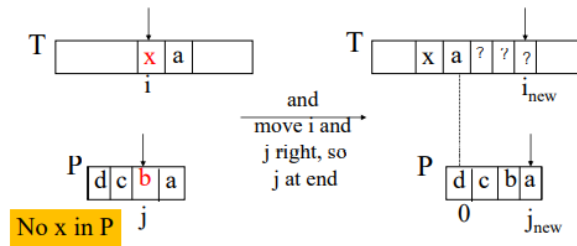
1. P mengandung x, coba geser P ke kanan untuk mencocokkan x pada P dengan T[i].



2. P mengandung x, namun penggeseran ke penemuan terakhir x tidak mungkin, maka coba geser P ke kanan ke T[i+1].



3. Selain kasus di atas, geser P untuk mencocokkan P[0] dengan T[i+1].



### 2.1.3 Algoritma Longest Common Subsequence (LCS)

Subsequence dari sebuah string adalah string yang dapat dibuat dari string awal dengan menghapus beberapa karakter dari string tersebut tanpa mengubah urutan string. Algoritma LCS mencari common subsequence terpanjang dari dua buah string. Algoritma diimplementasi menggunakan Dynamic Programming. Jika kedua string adalah  $A[1..n]$  dan  $B[1..m]$ , maka kita mendefinisikan state  $dp[i][j]$  sebagai LCS dari  $A[1..i]$  dan  $B[1..j]$ , dan kita ingin mencari  $dp[n][m]$ . Dengan bottom-up DP, kita perlu mengisi seluruh elemen tabel memoisasi DP, dimana masing-masing elemen dapat diisi dalam waktu  $O(1)$  dengan transisi DP. Maka, algoritma ini memiliki kompleksitas  $O(nm)$ .

	A	C	C	G	A	T	C	G
G	0	0	0	0	0	0	0	0
A	0	1	1	1	1	2	2	2
C	0	1	2	2	2	2	3	3
A	0	1	2	2	2	3	3	3
T	0	1	2	2	2	3	4	4

Gambar 2.1.3 Pengisian tabel DP algoritma LCS

### 2.1.4 Regular Expression (Regex)

Regular expression adalah urutan karakter yang digunakan sebagai pola pencarian atau pencocokan suatu teks. Berikut adalah beberapa contoh syntax regex:

Regular expression cheatsheet	
<b>Special characters</b> <ul style="list-style-type: none"> <li><code>\</code> escape special characters</li> <li><code>.</code> matches any character</li> <li><code>^</code> matches beginning of string</li> <li><code>\$</code> matches end of string</li> <li><code>[5b-d]</code> matches any chars '5', 'b', 'c' or 'd'</li> <li><code>[^a-c6]</code> matches any char except 'a', 'b', 'c' or '6'</li> <li><code>R S</code> matches either regex <code>R</code> or regex <code>S</code></li> <li><code>()</code> creates a capture group and indicates precedence</li> </ul>	<b>Quantifiers</b> <ul style="list-style-type: none"> <li><code>*</code> 0 or more (append <code>?</code> for non-greedy)</li> <li><code>+</code> 1 or more (append <code>?</code> for non-greedy)</li> <li><code>?</code> 0 or 1 (append <code>?</code> for non-greedy)</li> <li><code>{m}</code> exactly <code>m</code> occurrences</li> <li><code>{m, n}</code> from <code>m</code> to <code>n</code>, <code>m</code> defaults to 0, <code>n</code> to infinity</li> <li><code>{m, n}?</code> from <code>m</code> to <code>n</code>, as few as possible</li> </ul>
<b>Special sequences</b> <ul style="list-style-type: none"> <li><code>\A</code> start of string</li> <li><code>\b</code> matches empty string at word boundary (between <code>\w</code> and <code>\W</code>)</li> <li><code>\B</code> matches empty string not at word boundary</li> <li><code>\d</code> digit</li> <li><code>\D</code> non-digit</li> <li><code>\s</code> whitespace: <code>[\t\n\r\f\v]</code></li> <li><code>\S</code> non-whitespace</li> <li><code>\w</code> alphanumeric: <code>[0-9a-zA-Z_]</code></li> <li><code>\W</code> non-alphanumeric</li> <li><code>\Z</code> end of string</li> <li><code>\g&lt;id&gt;</code> matches a previously defined group</li> </ul>	<b>Special sequences</b> <ul style="list-style-type: none"> <li><code>(?iLmsux)</code> matches empty string, sets re.X flags</li> <li><code>(?:...)</code> non-capturing version of regular parentheses</li> <li><code>(?P...)</code> matches whatever matched previously named group</li> <li><code>(?P=)</code> digit</li> <li><code>(?#...)</code> a comment; ignored</li> <li><code>(?=...)</code> lookahead assertion: matches without consuming</li> <li><code>(?!...)</code> negative lookahead assertion</li> <li><code>(?&lt;...)</code> lookbehind assertion: matches if preceded</li> <li><code>(?&lt;!...)</code> negative lookbehind assertion</li> <li><code>(? (id)yes no)</code> match 'yes' if group 'id' matched, else 'no'</li> </ul>

Based on tartley's [python-regex-cheatsheet](#).

Gambar 2.1.4 Syntax Regular Expression

## 2.2 Aplikasi Web

Aplikasi web dibagi menjadi dua bagian, yaitu *frontend* dan *backend*. Pada bagian *frontend*, *framework* yang digunakan adalah React. Di sisi lain, bagian *backend* menggunakan *framework* Go, dan database manager (DBMS) menggunakan PostgreSQL. Aplikasi web ini memiliki empat fitur seperti yang dijelaskan pada Bab I. Pemrosesan algoritma pencocokan string serta penggunaan *regular expression* dilakukan pada bagian *backend*.

## Bab III

# Analisis Pemecahan Masalah

### 3.1 Penyelesaian Masalah Setiap Fitur

#### 3.1.1 Fitur Penambahan Penyakit Baru

Pertama-pertama, aplikasi web akan menerima masukan berupa nama penyakit dan file teks yang berisi rantai DNA penyakit baru melalui form yang ada pada halaman Tambah Penyakit. Selanjutnya, bagian *frontend* membaca data masukan pengguna dan mengubah file teks *sequence* DNA tersebut dan menjadi tipe data string. Setelah itu, *frontend* mengirimkan POST method kepada *backend* untuk memproses data tersebut lebih lanjut. Pertama-tama, akan dilakukan sanitasi input pada masukan DNA penyakit menggunakan regular expression. Jika rantai DNA tidak valid, maka *backend* akan mengirimkan pesan error. Jika penyakit belum ada pada basis data, *backend* akan memasukkan penyakit tersebut ke dalam basis data. Setelah penyakit berhasil dimasukkan ke dalam basis data, *backend* akan mengirimkan respon kepada *frontend* bahwa penyakit berhasil dimasukkan dan *frontend* akan menampilkan pesan kepada pengguna bahwa penyakit berhasil ditambahkan. Jika penyakit gagal dimasukkan ke dalam basis data, *backend* akan mengirimkan respon sesuai dengan alasan kegagalannya kepada *frontend*. Pesan kegagalan juga akan ditampilkan kepada pengguna.

#### 3.1.2 Fitur Prediksi Penyakit Seseorang

Pertama-pertama, aplikasi web akan menerima masukan berupa nama pengguna, prediksi penyakit, pilihan algoritma pencocokan string, dan file teks yang berisi *sequence* DNA pengguna melalui form yang ada pada halaman Tes DNA. Selanjutnya, bagian *frontend* membaca data masukan pengguna dan mengubah file teks *sequence* DNA tersebut dan menjadi tipe data string. Setelah itu, *frontend* mengirimkan POST method kepada *backend* untuk memproses data tersebut lebih lanjut. Pertama-tama, akan dilakukan sanitasi input pada masukan DNA penyakit menggunakan regular expression. Jika rantai DNA tidak valid, maka *backend* akan mengirimkan pesan error. Jika rantai DNA valid, algoritma pencocokan string pilihan pengguna akan dijalankan antara rantai DNA prediksi penyakit yang ada pada basis data dengan rantai DNA pengguna. Jika keluaran algoritma pencocokan string bernilai True, nilai *similarity* akan di-set menjadi 1. Jika keluarannya False, algoritma LCS akan dijalankan untuk mendapatkan nilai *similarity* antara rantai DNA pengguna dan rantai DNA penyakit yang diprediksikan. Jika nilai *similarity*-nya lebih besar atau sama dengan 80%, pengguna dinilai mengidap penyakit yang diprediksikan. Selanjutnya, *backend* akan memasukkan hasil tes tersebut ke dalam basis data lalu mengirimkan respon berupa hasil tes tersebut kepada *frontend*. Terakhir, *frontend* akan menerima respon tersebut dan menampilkannya kepada pengguna.

### 3.1.3 Fitur Pencarian Riwayat Tes

Pertama-pertama, aplikasi web akan menerima query pencarian berupa tanggal atau nama penyakit, maupun keduanya. Setelah itu, *frontend* mengirimkan POST method kepada *backend* untuk memproses data tersebut lebih lanjut. Pertama-tama, akan dilakukan sanitasi input pada masukan query pencarian dari pengguna menggunakan regular expression. Jika query tidak valid, maka *backend* akan mengirimkan pesan error. Jika query valid, *backend* akan mengambil data riwayat tes dari basis data berdasarkan query tersebut. Selanjutnya, *backend* akan mengirimkan respon berupa data riwayat tes tersebut kepada *frontend*. Terakhir, *frontend* akan menampilkan data riwayat tes tersebut kepada pengguna. Jika data tidak ditemukan, *frontend* juga akan menampilkan pesan bahwa tidak ada data yang ditemukan.

### 3.1.4 Fitur Tingkat Kemiripan DNA

Dalam menentukan tingkat kemiripan, digunakan algoritma Longest Common Subsequence (LCS) antara string DNA penyakit dan DNA pengguna yang ingin diuji. Dalam menghitung tingkat kemiripan string DNA penyakit dan DNA pengguna, digunakan perhitungan tingkat kemiripan yaitu panjang LCS dibagi dengan panjang DNA penyakit. Dengan kata lain, jika DNA penyakit adalah subsequence dari DNA pengguna, maka algoritma LCS akan mengeluarkan tingkat kemiripan 100%. Algoritma ini hanya akan dipanggil jika algoritma KMP atau Boyer-Moore mengeluarkan nilai false.

## 3.2 Fitur Fungsional dan Arsitektur Aplikasi Web

Fitur fungsional yang kami implementasikan dalam aplikasi web ini, antara lain:

1. Pengguna dapat menambahkan penyakit baru berdasarkan masukan file teks rantai DNA
2. Pengguna dapat melakukan tes prediksi penyakit berdasarkan masukan prediksi penyakit dan file teks rantai DNA seseorang
3. Pengguna dapat melihat hasil prediksi penyakitnya
4. Pengguna dapat mencari riwayat tes berdasarkan query yang dimasukkan
5. Pengguna dapat melihat tingkat kemiripan rantai DNA-nya dengan rantai DNA penyakit prediksi setelah melakukan tes

Fitur fungsional yang berupa kemampuan pengguna untuk memasukkan data diaplikasikan dalam bentuk formulir pada aplikasi web. Contohnya seperti pada Gambar 3.2.



The image shows a web form with a dark green background. It contains four main sections: 1. 'Nama Pengguna:' with a text input field containing the placeholder 'Masukkan nama pengguna'. 2. 'Prediksi Penyakit:' with a text input field containing the placeholder 'Masukkan prediksi penyakit'. 3. 'Pilih algoritma string matching:' with a dropdown menu showing 'KMP' and a downward arrow. 4. 'Unggah file teks rantai DNA:' with a file upload area showing 'Choose File' and 'No file chosen'. At the bottom left is a green 'Submit' button.

Gambar 3.2 Contoh formulir pada aplikasi web

Arsitektur yang digunakan dalam aplikasi web ini adalah REST dengan menggunakan *library* Axios untuk membuat HTTP *request* kepada *backend*.

## Bab IV

# Implementasi dan Pengujian

### 4.1 Spesifikasi Teknis Aplikasi

Struktur program secara garis besar terbagi menjadi dua, yaitu *frontend* yang ditulis dengan *framework* React, dan *backend* yang ditulis dalam bahasa Go.

#### 4.1.1 Frontend

Program *frontend* terdiri dari beberapa folder dan file yang merupakan bagian dari React App. Folder *node\_modules* terdiri dari *dependencies* yang dipakai dalam aplikasi web ini. Folder *public* terdiri *public assets*, seperti logo, favicon, dll. Folder *src* merupakan folder utama dari aplikasi web ini yang berisi file Javascript dan CSS. Di dalam folder *src* terdapat file *index.js* yang terdiri dari beberapa fungsi untuk *me-render* HTML dan CSS untuk masing-masing halaman web. Selain itu, terdapat juga fungsi *SideBars* yang *me-render* sebuah *sidebar* yang akan ditampilkan pada setiap halaman. Terdapat 4 fungsi untuk tiap halaman, yaitu Home, TambahPenyakit, TesDNA, dan RiwayatTes. Fungsi-fungsi tersebut kecuali fungsi Home juga melakukan metode POST untuk mengirimkan ke *backend* data yang dimasukkan oleh pengguna. Selain itu, terdapat juga beberapa *react-hook-state* yang berguna untuk mengubah tampilan web setelah pengguna memasukkan data pada form.

#### 4.1.2 Backend

Program *backend* terdiri dari dua bagian. Program *string.go* handle seluruh logic string seperti sanitasi input dengan regex, serta algoritma KMP, Boyer-Moore, dan LCS. Program *main.go* handle seluruh request dari *frontend*, seperti tes DNA, menambahkan penyakit, dan riwayat tes. Untuk melakukan ini, *main.go* terbagi menjadi empat fungsi utama, yaitu fungsi *connect* yang membuat koneksi ke basis data (postgresql yang di-host di heroku), fungsi handler tes DNA, fungsi handler tambah penyakit, dan fungsi handler riwayat tes.

Dalam *string.go*, terdapat dua fungsi yang menangani regex dan tiga fungsi yang menangani string matching. Fungsi *IsDNA* menggunakan regex untuk mengecek apakah argumen fungsi merupakan rantai DNA yang valid. Fungsi *querySplit* menggunakan regex untuk membagi query riwayat tes menjadi bagian tanggal dan nama penyakit. Fungsi KMP melakukan string matching dengan algoritma KMP, dan fungsi Boyer-Moore melakukan string matching dengan algoritma Boyer-Moore. Fungsi LCS menggunakan algoritma DP LCS untuk mencari LCS dari kedua argumen masukan.

Dalam *main.go*, Fungsi handler tes DNA menerima masukan form tes dari *frontend* dan melakukan sanitasi input dengan memanggil fungsi dari *string.go*. Jika input lolos pengecekan, maka handler tes DNA akan melakukan query ke basis data untuk mengambil rantai DNA penyakit, dan jika penyakit ditemukan, handler akan memanggil fungsi string matching yang sesuai untuk melakukan pengujian DNA, kemudian mengembalikan hasil ke *frontend*.

Fungsi handler tambah penyakit menerima masukan nama penyakit dan rantai DNA dari *frontend* dan melakukan sanitasi input dengan memanggil fungsi dari *string.go*. Jika input lolos pengecekan, maka handler tambah penyakit akan melakukan query ke basis data untuk menambahkan penyakit ke basis data. Jika penyakit sudah ada, maka akan dikirimkan pesan error ke *frontend*, dan akan dikirimkan pesan sukses jika query berhasil.

Fungsi handler riwayat tes menerima masukan query dari pengguna, dan memproses query dengan sanitasi regex dari *string.go*. Kemudian, handler riwayat tes akan melakukan query yang sesuai ke basis data, dan mengembalikan hasil (jika ada) ke *frontend*.

## 4.2 Tata Cara Penggunaan Aplikasi

### 4.2.1 Interface

Berikut adalah beberapa tampilan interface utama aplikasi:



Gambar 4.2.1.1 Interface Home Page



**Malignant**

- Home
- Tambah Penyakit
- Tes DNA
- Riwayat Tes

## Tambahkan DNA Penyakit

Tambahkan DNA penyakit agar jangkauan tes semakin luas!

**Nama Penyakit:**

**Unggah file teks rantai DNA:**

**Submit**

Gambar 4.2.1.2 Interface Tambah Penyakit

**Malignant**

- Home
- Tambah Penyakit
- Tes DNA
- Riwayat Tes

## Tes DNA Anda Sekarang!

Ketika seseorang memiliki kelainan genetik atau DNA, misalnya karena penyakit keturunan atau karena faktor lainnya, ia bisa mengalami penyakit tertentu. Oleh karena itu, tes DNA penting untuk dilakukan untuk mengetahui struktur genetik di dalam tubuh seseorang serta mendeteksi kelainan genetik.

**Nama Pengguna:**

**Prediksi Penyakit:**

**Pilih algoritma string matching:**

KMP

Gambar 4.2.1.3 Interface Tes DNA

Malignant

- Home
- Tambah Penyakit
- Tes DNA
- Riwayat Tes

Nama Pengguna:

Masukkan nama pengguna

Prediksi Penyakit:

Masukkan prediksi penyakit

Pilih algoritma string matching:

KMP

Unggah file teks rantai DNA:

Choose File No file chosen

Submit

Gambar 4.2.1.4 Interface Form Tes DNA

Malignant

- Home
- Tambah Penyakit
- Tes DNA
- Riwayat Tes

## Riwayat Tes Anda

Melihat semua riwayat tes Anda dalam satu genggaman. Segera konsultasikan kepada ahlinya mengenai riwayat tes Anda!

Query:

Masukkan query pencarian

Contoh: "18-04-2022 HIV", "18-04-2022", "HIV"

Submit

Gambar 4.2.1.5 Interface Riwayat Tes

## 4.2.2 Fitur-fitur Aplikasi

## 4.3 Hasil Pengujian

### 4.3.1 Skenario 1

Sukses menambahkan penyakit

The screenshot shows the 'Malignant' application interface. On the left is a dark green sidebar with the title 'Malignant' and four menu items: 'Home' (house icon), 'Tambah Penyakit' (gear icon), 'Tes DNA' (person icon), and 'Riwayat Tes' (clock icon). The main content area has a dark blue header with a green wavy pattern. Below this, there are two input fields: 'Nama Penyakit:' with the value 'Penyakit4' and 'Unggah file teks rantai DNA:' with a file selection button 'Choose File' and the filename 'seed-Penyakit4.txt'. A green 'Submit' button is below these fields. At the bottom, a light green message box states 'Penyakit berhasil ditambahkan'.

Gambar 4.3.1 Hasil pengujian skenario 1

#### 4.3.2 Skenario 2

Gagal menambahkan penyakit (rantai DNA salah)

The screenshot shows the 'Malignant' application interface. The sidebar and header are identical to the previous screenshot. In the main content area, the 'Nama Penyakit:' field contains 'PenyakitX'. The 'Unggah file teks rantai DNA:' field shows a file selection button 'Choose File' and the filename 'a.txt'. A green 'Submit' button is present. At the bottom, a pink message box states 'File tidak berisi DNA'.

Gambar 4.3.2 Hasil pengujian skenario 2

#### 4.3.3 Skenario 3

Gagal menambahkan penyakit (penyakit sudah ada)

The screenshot shows the 'Malignant' application interface. On the left is a dark green sidebar with a home icon and the title 'Malignant'. The main content area has a dark green background with a light green wavy pattern at the top and bottom. It contains the following elements:

- Nama Penyakit:** A text input field containing 'Penyakit0'.
- Unggah file teks rantai DNA:** A file upload section with a 'Choose File' button, a text input field containing 'seed-Penyakit0.txt', and a green 'Submit' button.
- Feedback:** A pink rectangular box at the bottom containing the text 'Penyakit sudah ada'.

Gambar 4.3.3 Hasil pengujian skenario 3

#### 4.3.4 Skenario 4

Gagal Tes DNA (rantai DNA salah)

The screenshot shows the 'Malignant' application interface for a failed test. It includes the same sidebar and header as the previous image. The main content area contains:

- Nama Pengguna:** A text input field containing 'Erik'.
- Prediksi Penyakit:** A text input field containing 'Penyakit0'.
- Pilih algoritma string matching:** A dropdown menu with 'KMP' selected.
- Unggah file teks rantai DNA:** A file upload section with a 'Choose File' button, a text input field containing 'a.txt', and a green 'Submit' button.
- Feedback:** A pink rectangular box at the bottom containing the text 'Mohon maaf, tes Anda gagal!' and 'File tidak berisi DNA'.

Gambar 4.3.4 Hasil pengujian skenario 4

#### 4.3.5 Skenario 5

Sukses Tes DNA

**Malignant**

- Home
- Tambah Penyakit
- Tes DNA
- Riwayat Tes

Prediksi Penyakit:

Penyakit4

Pilih algoritma string matching:

Boyer-Moore

Unggah file teks rantai DNA:

Choose File | Test0.txt

Submit

**Tes Anda berhasil!**

Terima kasih telah melakukan tes di Malignant. Segera konsultasikan hasil tes Anda kepada ahlinya!

Tanggal	29-04-2022
Pengguna	Erik
Penyakit	Penyakit4
Tingkat Kesamaan	47%
Hasil	False

Gambar 4.3.5 Hasil pengujian skenario 5

#### 4.3.6 Skenario 6

Searching dengan tanggal

**Malignant**

- Home
- Tambah Penyakit
- Tes DNA
- Riwayat Tes

Query:

26-04-2022

Contoh: "18-04-2022 HIV", "18-04-2022", "HIV"

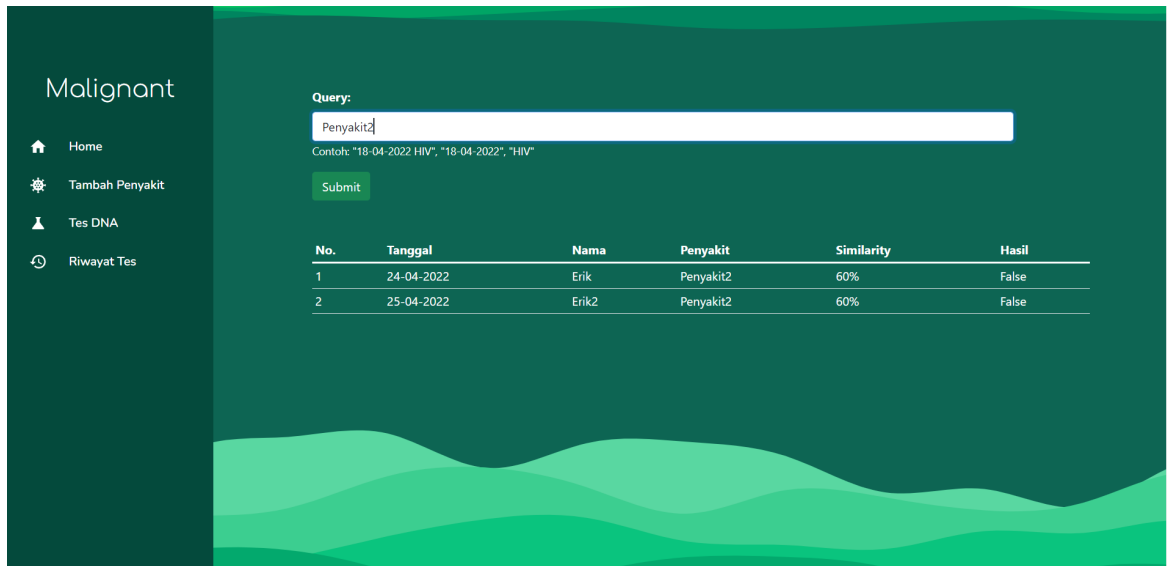
Submit

No.	Tanggal	Nama	Penyakit	Similarity	Hasil
1	26-04-2022	Daffa	Penyakit1	63%	False
2	26-04-2022	Budi	Penyakit1	63%	False
3	26-04-2022	Daffa	Penyakit1	63%	False
4	26-04-2022	Asep	Penyakit1	63%	False
5	26-04-2022	Anto	Penyakit0	47%	False
6	26-04-2022	Anto	Penyakit0	47%	False
7	26-04-2022	Daffa	Penyakit1	63%	False
8	26-04-2022	qwerty	Penyakit0	31%	False
9	26-04-2022	kmptest	Penyakit0	31%	False
10	26-04-2022	kmptest	Penyakit0	31%	False

Gambar 4.3.6 Hasil pengujian skenario 6

#### 4.3.7 Skenario 7

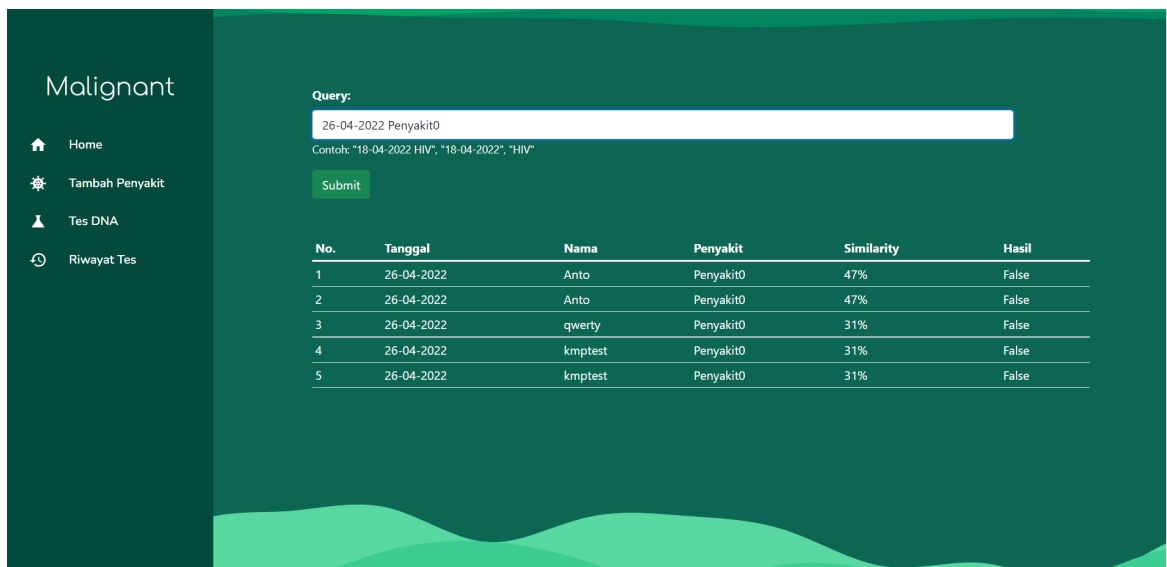
Searching dengan nama penyakit



Gambar 4.3.7 Hasil pengujian skenario 7

### 4.3.8 Skenario 8

Searching dengan tanggal dan nama penyakit



Gambar 4.3.8 Hasil pengujian skenario 8

## 4.4 Analisis Hasil Pengujian

Seluruh hasil pengujian sukses dan sesuai spesifikasi program. Penambahan penyakit tidak dapat dilakukan jika rantai DNA tidak valid atau penyakit sudah ada di basis data (Skenario 1 dan 2). Selain kedua kasus tersebut, penambahan penyakit pasti sukses (Skenario 3). Tes DNA hanya akan gagal jika rantai DNA masukan tidak valid (Skenario 4). Selain itu, tes DNA akan

mengeluarkan hasil (Skenario 5). Pencarian dapat dilakukan berdasarkan tanggal, nama penyakit, atau keduanya (Skenario 6, 7, dan 8).

## Bab V

# Kesimpulan dan Saran

### 5.1 Kesimpulan

Pada Tugas Besar III IF2211 Strategi Algoritma 2021/2022 berjudul “Penerapan String Matching dan Regular Expression dalam DNA Pattern Matching,” kami berhasil membuat dan meluncurkan sebuah aplikasi web “DNA Pattern Matching” yang memanfaatkan implementasi dari algoritma *string matching* dan *regular expression*. Aplikasi web yang dibuat memiliki fitur berupa pencarian pola *sequence* penyakit di dalam *sequence* DNA yang diberikan serta melihat riwayat pencocokan.

### 5.2. Saran

Saran yang dapat kami berikan untuk Tugas Besar III IF2211 Strategi Algoritma 2021/2022 ini, yaitu agar menyediakan video tutorial singkat yang dapat menjelaskan *frontend* atau *backend* dalam framework tertentu beserta cara menghubungkannya. Selain itu, saran untuk kami pribadi terkait hasil tugas kami adalah dapat menambah fitur tambahan seperti *alert* kepada pengguna untuk *field form* yang *required*. Aplikasi web juga dapat dibuat lebih responsif.

### 5.3. Komentar dan Refleksi

Dengan adanya tugas ini, kami menjadi paham cara membuat aplikasi web walaupun kami belum mendapatkan mata kuliah tersebut.



## Daftar Pustaka

- [1] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. 2009.
- [1] <https://www.kaggle.com/code/sohier/introduction-to-regular-expressions/notebook>,| diakses pada 29 April 2022.

<b>Link Repository Github Kelompok Malignant</b>
<a href="https://github.com/dxt99/Tubes3_13520108">https://github.com/dxt99/Tubes3_13520108</a>

<b>Link Aplikasi Web</b>
<a href="https://malignant.netlify.app/">https://malignant.netlify.app/</a>

<b>Link Video Demonstrasi</b>
<a href="https://youtu.be/e3La0BCug9Q">https://youtu.be/e3La0BCug9Q</a>