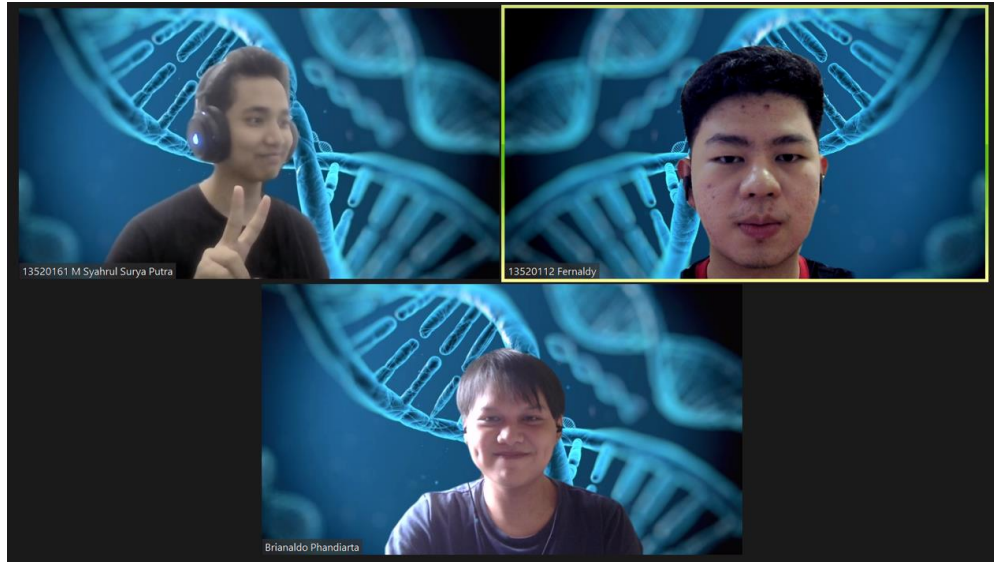


Penerapan String Matching dan Regular Expression dalam DNA

Pattern Matching

Dibuat sebagai Tugas Besar 3

IF2211 Strategi Algoritma



Dikerjakan oleh:

Finding (D)N(A)emo

Fernaldy 13520112

Brianaldo Phandiarta 13520113

M Syahrul Surya Putra 13520161

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

DAFTAR ISI

BAB I DESKRIPSI TUGAS	1
BAB II LANDASAN TEORI	2
I. Pattern Matching	2
II. Knuth-Morris-Pratt (KMP) Algorithm.....	2
III. Boyer-Moore (BM) Algorithm	2
IV. Regular Expression (Regex)	3
V. Hamming Distance.....	3
VI. Penjelasan singkat Aplikasi Web Finding (D)N(A)emo.....	4
BAB III ANALISIS PEMECAHAN MASALAH.....	5
I. Langkah Pemecahan Masalah Setiap Fitur	5
II. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun.....	7
BAB IV IMPLEMENTASI DAN PENGUJIAN	9
I. Spesifikasi Teknis Program	9
II. Tata Cara Penggunaan Program.....	12
III. Pengujian.....	14
IV. Analisis Pengujian	20
BAB V KESIMPULAN DAN SARAN.....	22
V. Kesimpulan	22
VI. Saran	22
VII. Refleksi	22
DAFTAR PUSTAKA	23
LAMPIRAN.....	24

BAB I

DESKRIPSI TUGAS

Manusia umumnya memiliki 46 kromosom di dalam setiap selnya. Kromosom-kromosom tersebut tersusun dari DNA (deoxyribonucleic acid) atau asam deoksiribonukleat. DNA tersusun atas dua zat basa purin, yaitu Adenin (A) dan Guanin (G), serta dua zat basa pirimidin, yaitu sitosin (C) dan timin (T). Masing-masing purin akan berikatan dengan satu pirimidin. DNA merupakan materi genetik yang menentukan sifat dan karakteristik seseorang, seperti warna kulit, mata, rambut, dan bentuk wajah. 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. Ada berbagai jenis tes DNA yang dapat dilakukan, seperti uji pra implantasi, uji pra kelahiran, uji pembawa atau carrier testing, uji forensik, dan DNA sequence analysis.

Salah satu jenis tes DNA yang sangat berkaitan dengan dunia bioinformatika adalah DNA sequence analysis. DNA sequence analysis adalah sebuah cara yang dapat digunakan untuk memprediksi berbagai macam penyakit yang tersimpan pada database berdasarkan urutan sekuens DNA-nya. Sebuah sekuens DNA adalah suatu representasi string of nucleotides yang disimpan pada suatu rantai DNA, sebagai contoh: ATTCGTAAGTAAAGTTA. Teknik pattern matching memegang peranan penting untuk dapat menganalisis sekuens DNA yang sangat panjang dalam waktu singkat. Oleh karena itu, mahasiswa Teknik Informatika berniat untuk membuat suatu aplikasi web berupa DNA Sequence Matching yang menerapkan algoritma String Matching dan Regular Expression untuk membantu penyedia jasa kesehatan dalam memprediksi penyakit pasien. Hasil prediksi juga dapat ditampilkan dalam tabel dan dilengkapi dengan kolom pencarian untuk membantu admin dalam melakukan filtering dan pencarian.

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.

BAB II

LANDASAN TEORI

I. Pattern Matching

Pattern matching adalah suatu teknik untuk mencari lokasi keberadaan suatu *pattern* yaitu suatu *string* dengan panjang m karakter pada suatu teks yaitu suatu *string* dengan panjang n karakter. Asumsi yang diambil adalah panjang *string* teks selalu lebih panjang daripada panjang *string* *pattern*. *Pattern matching* memiliki banyak kegunaan seperti pencarian dalam suatu *file*, *web search engine*, analisis citra, *bioinformatics*, dan lain sebagainya. Hingga saat ini, sudah terdapat banyak implementasi algoritma *pattern matching* dan beberapa di antaranya adalah Knuth-Morris-Pratt (KMP) Algorithm dan Boyer-Moore (BM) Algorithm.

II. Knuth-Morris-Pratt (KMP) Algorithm

Algoritma KMP dicetuskan oleh Donald Ervin Knuth, seorang ilmuwan komputer dan Profesor Emeritus dari Stanford University. Algoritma KMP melakukan pencarian *pattern* pada teks dari kiri ke kanan seperti pencarian dengan menggunakan algoritma *brute force*, tetapi dengan pergeseran *pattern* yang lebih efisien. Apabila terdapat ketidakcocokan pada teks posisi i dan pola posisi j , maka pergeseran yang dilakukan sebesar prefiks *pattern* dari posisi 0 hingga $j-1$ terbesar yang sama dengan sufiks *pattern* dari posisi 1 hingga $j-1$. Kompleksitas waktu algoritma KMP adalah $O(n+m)$. Keuntungan dari algoritma KMP ini adalah tidak pernah bergerak mundur pada teks, sedangkan kelemahannya adalah ketika ukuran alfabet besar.

III. Boyer-Moore (BM) Algorithm

Algoritma BM didasarkan pada teknik *looking-glass* dan *character-jump*. Teknik *looking-glass* adalah teknik yang menelusuri kecocokan pola dengan teks dari posisi belakang pola. Teknik *character-jump* adalah mendefinisikan pergeseran pola berdasarkan tiga kemungkinan kasus apabila ditemukan ketidakcocokan teks posisi i dan pola posisi j . Kasus pertama adalah apabila pola mengandung karakter teks posisi i di sebelah kiri j , maka sejajarkan posisi karakter tersebut dengan karakter teks posisi i . Kasus kedua adalah apabila pola mengandung karakter teks posisi i di sebelah kanan j , maka geser pola sebesar 1 karakter ke kanan. Kasus ketiga adalah kasus selain kasus pertama dan kedua yaitu sejajarkan pola posisi 0 dengan teks posisi $i+1$. Kompleksitas waktu

kasus terburuk algoritma Boyer-Moore adalah $O(nm+A)$. Keuntungan algoritma Boyer-Moore adalah ukuran alfabet yang besar.

IV. Regular Expression (Regex)

Regular expression merupakan pola yang terdiri atas sekumpulan karakter yang mendefinisikan pola yang dicari pada suatu teks. Regex mampu mendefinisikan pola yang fleksibel untuk dicari dalam teks. Beberapa pemanfaatan regex adalah untuk validasi struktur suatu teks, pencarian pola pada teks, penggantian bagian tertentu pada teks, dan lain sebagainya. Beberapa contoh notasi umum pada regex antara lain:

- `.` mendefinisikan semua karakter kecuali newline
- `^` mendefinisikan awal teks
- `$` mendefinisikan akhir teks
- `[abc]` mendefinisikan karakter a, b, atau c
- `[a-z]` mendefinisikan karakter apapun dari a sampai z
- `[^abc]` mendefinisikan karakter apapun kecuali a, b, dan c
- `aa|bb` mendefinisikan aa atau bb
- `?` mendefinisikan 0 atau 1 kemunculan dari karakter sebelumnya
- `*` mendefinisikan 0 atau lebih kemunculan dari karakter sebelumnya
- `+` mendefinisikan 1 atau lebih kemunculan dari karakter sebelumnya

Selain contoh notasi umum di atas, masih banyak notasi umum regex lainnya.

V. Hamming Distance

Hamming distance adalah jumlah substitusi minimum yang diperlukan untuk mengubah suatu string menjadi string lain yang panjangnya sama. Menghitung hamming distance dapat dilakukan dengan menghitung jumlah ketidakcocokan karakter pada posisi yang sama dari dua buah string dengan panjang sama. Dalam melakukan *pattern matching*, *hamming distance* dapat dihitung pada tiap *offset* teks untuk menentukan tingkat kecocokan pola dengan teks. Tingkat kecocokan tertinggi akan diambil berdasarkan *hamming distance* minimum dari pola dengan teks.

VI. Penjelasan singkat Aplikasi Web Finding (D)N(A)emo

Dalam rekayasa perangkat lunak, aplikasi web adalah suatu aplikasi yang diakses menggunakan penjelajah web melalui suatu jaringan seperti internet atau intranet. Aplikasi web Finding (D)N(A)emo merupakan aplikasi web yang dapat diakses melalui jaringan internet. Aplikasi web ini bertujuan untuk melakukan pendeteksian penyakit dengan menggunakan DNA *pattern-matching*. Selain itu, aplikasi web ini juga dapat menyimpan data-data penyakit dan hasil pemeriksaan DNA.

BAB III

ANALISIS PEMECAHAN MASALAH

I. Langkah Pemecahan Masalah Setiap Fitur

Untuk fitur menambah penyakit baru, aplikasi web dapat menerima masukan nama penyakit beserta sekuens DNA dari penyakit tersebut. Untuk memastikan bahwa sekuens DNA penyakit yang dimasukkan adalah valid, akan dilakukan pengecekan terhadap sekuens DNA tersebut dengan menggunakan regex. Berikut adalah regex untuk melakukan sanitasi input DNA:

Regex: `[^AGCT]`

Regex tersebut mencari apakah terdapat karakter selain A, G, C, dan T pada sekuens DNA penyakit. Apabila ditemukan ada karakter selain A, G, C, dan T, maka masukan sekuens DNA penyakit tidak valid. Selain itu, akan dilakukan pengecekan terhadap nama penyakit dengan yang tersimpan di dalam basis data agar tidak terdapat duplikasi penyakit. Apabila valid, nama penyakit beserta sekuens DNA penyakit akan disimpan di dalam basis data.

Untuk fitur prediksi penyakit seseorang, aplikasi web dapat menerima nama orang yang ingin diperiksa, sekuens DNA orang tersebut, dan nama penyakit yang ingin diperiksa. Pengecekan terhadap sekuens DNA orang tersebut juga akan dilakukan dengan regex yang sama seperti pada validasi sekuens DNA penyakit. Selain itu, juga akan dilakukan validasi terhadap nama penyakit yang ingin diperiksa apakah tersedia di dalam basis data. Apabila semua kondisi valid, akan dilakukan pencocokan DNA penyakit terhadap DNA orang tersebut. Terlebih dahulu akan dilakukan *string matching* dengan algoritma KMP untuk mendeteksi kecocokan 100% antara DNA penyakit dengan DNA orang tersebut. Apabila kecocokan tidak 100%, selanjutnya akan dilakukan pencocokan pola dengan memanfaatkan *hamming distance* untuk menentukan tingkat kecocokan DNA penyakit dengan DNA orang tersebut. DNA penyakit dan DNA orang tersebut dikatakan cocok apabila tingkat kecocokannya mencapai 80%. Selanjutnya, tanggal pengecekan, nama orang, penyakit prediksi, tingkat kecocokan, dan status terprediksi akan disimpan di dalam basis data.

Berikut adalah algoritma KMP:

- Melakukan `ComputeFail` yang mengembalikan suatu array of integer, fail. Apabila terdapat ketidakcocokan karakter teks pada posisi *i* dengan karakter pola pada posisi *j*, `fail[j-1]`

menunjukkan posisi j yang baru. Nilai $fail[k]$ dihitung dengan menghitung ukuran prefiks terbesar yang sama dengan sufiks mulai dari $pola[0]$ hingga $pola[k]$

- Melakukan iterasi dari awal teks hingga akhir dengan *iterator* i dan iterasi dari awal pola hingga akhir dengan *iterator* j . Apabila $teks[i]$ dan $pola[j]$ sama, apabila j sudah mencapai akhir pola, maka kembalikan $i - \text{panjang pola} + 1$, sedangkan apabila belum mencapai akhir pola maka keduanya di-*increment*. Apabila $teks[i]$ dan $pola[j]$ tidak sama, apabila $j > 0$ maka nilai j diubah menjadi $fail[j-1]$, sedangkan apabila $j=0$ maka nilai i di-*increment*. Berikut adalah algoritma Boyer-Moore:

- Melakukan komputasi LastOccurence yaitu posisi terakhir ditemukannya suatu karakter pada pola.
- Melakukan iterasi teks dari awal sampai akhir dengan *iterator* i dan iterasi dari akhir pola hingga awal dengan *iterator* j . Apabila $teks[i]$ dan $pola[j]$ sama, apabila $j=0$ maka kembalikan i , sedangkan apabila $j < 0$ maka i dan j di-*decrement*. Apabila $teks[i]$ dan $pola[j]$ tidak sama, apabila $last[teks[i]] < j$, maka sejajarkan posisi karakter tersebut dengan $teks[i]$, sedangkan apabila $last[teks[i]] > j$, maka geser pola sebesar 1 karakter ke kanan, sedangkan apabila $teks[i]$ tidak ada dalam pola maka sejajarkan $pola[0]$ dengan $teks[i+1]$.
- Iterasi berhenti apabila i sudah melebihi panjang teks.

Berikut adalah algoritma *hamming distance*:

- Iterasi dari awal teks hingga akhir dengan *iterator* i .
- Pada tiap iterasi, hitung *hamming distance* dari pola dengan *iterator* j dengan teks pada *offset* i . *Hamming distance* dihitung dengan jumlah $teks[i+j] \neq pola[j]$.
- Untuk mempercepat perhitungan, disimpan nilai minimum *hamming distance* dan jika pada *offset* i berikutnya, apabila *hamming distance* sudah mencapai minimum tersebut, maka iterasi dilanjutkan ke *offset* i berikutnya.

Untuk fitur pencarian hasil prediksi, aplikasi web dapat menerima *query* pencarian berupa tanggal prediksi, nama penyakit, atau tanggal prediksi yang diikuti nama penyakit. Validasi *query* ini akan dilakukan dengan regex. Berikut adalah regex untuk validasi *query*:

Regex untuk bulan dengan 31 hari:

```
^(0[1-9]|[12][1-9]|[123]0|31)[-/](0[13578]|10|12)[-/](202[0-2])|(20[01])[0-9](1[0-9]{1,3})|(0[0-9]{1,3}))$
```

Regex untuk bulan dengan 30 hari:

<pre>^(0[1-9] [12][1-9] [123]0)[-/](0[469] 11)[-/]((202[0-2]) (20[01]))[0-9] (1[0-9]{1,3}) (0[0-9]{1,3}))\$</pre> <p>Regex untuk bulan Februari:</p> <pre>^(0[1-9] [12][1-9] [12]0)[-/](02)[-/]((202[0-2]) (20[01]))[0-9] (1[0-9]{1,3}) (0[0-9]{1,3}))\$</pre>
<p>Regex untuk nama penyakit:</p> <pre>^[a-zA-Z0-9\s]+\$</pre>
<p>Regex untuk tanggal diikuti nama penyakit:</p> <pre>^(((0[1-9] [12][1-9] [123]0 31)[-/](0[13578] 10 12)[-/]((202[0-2]) (20[01]))[0-9] (1[0-9]{1,3}) (0[0-9]{1,3}))) ((0[1-9] [12][1-9] [123]0)[-/](0[469] 11)[-/]((202[0-2]) (20[01]))[0-9] (1[0-9]{1,3}) (0[0-9]{1,3}))) ((0[1-9] [12][1-9] [12]0)[-/](02)[-/]((202[0-2]) (20[01]))[0-9] (1[0-9]{1,3}) (0[0-9]{1,3}))))\s[a-zA-Z0-9\s]+\$</pre>

Regex tersebut dapat menerima tanggal dengan format dd-mm-yyyy dengan asumsi bahwa bulan Februari selalu memiliki 29 hari. Apabila *query* pencarian valid, akan dilakukan pencarian hasil prediksi yang tersimpan di dalam basis data dan menampilkan hasilnya ke layar.

II. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

i) Fitur Fungsional

- Menambahkan data penyakit baru ke basis data Penyakit.
- Melakukan pemeriksaan untaian DNA dengan *pattern* penyakit pada basis data Penyakit dan memasukkan hasilnya ke basis data Hasil.
- Melakukan pencarian data pada basis data Hasil sesuai dengan masukan oleh pengguna.

ii) Arsitektur Aplikasi Web

a) *Frontend*

Frontend dari Finding (D)N(A)emo dibangun dengan menggunakan bahasa javascript menggunakan *framework* React JS. Dalam menerima dan mengirimkan data pada API, digunakan pustaka axios. Situs di-*deploy* secara *online* menggunakan bantuan Netlify.

b) *Backend*

Backend dari Finding (D)N(A)emo dibangun menggunakan golang. Dalam pembangunannya, digunakan pustaka mysql, untuk melakukan koneksi pada basis data; mux, untuk melakukan koneksi API; dan cors untuk melakukan *set-up* CORS. *Backend* di-*deploy* secara *online* menggunakan bantuan Heroku.

c) *Basis data*

Finding (D)N(A)emo menggunakan basis data MySQL yang di-*deploy* secara *online* menggunakan remotemysql.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

I. Spesifikasi Teknis Program

iii) Backend

a) lib

Berisi pustaka yang digunakan untuk melakukan *pattern-matching*.

(a) Regex.go

- Fungsi IsValidDNA

Fungsi ini mengembalikan true jika panjang input > 0 dan input tidak mengandung karakter lain selain AGCT

- Fungsi IsValidQuery

Fungsi ini mengembalikan true jika input merupakan tanggal dengan format dd-mm-yyyy, nama penyakit, atau gabungan keduanya

(b) StringMatching.go

- Fungsi ComputeFail

Fungsi untuk menghitung border function, yaitu jumlah pergeseran yang diperlukan apabila ditemukan ketidakcocokan antara pola dan teks

- Fungsi KmpMatch

Fungsi melakukan pencocokan string dengan KMP Algorithm memanfaatkan ComputeFail. Fungsi akan mengembalikan indeks ditemukannya pola pada teks jika pola ditemukan atau -1 jika pola tidak ditemukan.

- Fungsi BuildLast

Fungsi memetakan setiap karakter pada alfabet ke indeks di mana karakter tersebut terakhir ditemukan pada pola.

- Fungsi BmMatch

Fungsi melakukan pencocokan string dengan Boyer-Moore Algorithm memanfaatkan BuildLast. Fungsi akan mengembalikan indeks

ditemukannya pola pada teks jika pola ditemukan atau -1 jika pola tidak ditemukan.

- Fungsi HammingDistance

Fungsi HammingDistance melakukan perhitungan hamming distance antara pola dengan tiap offset pada teks dan mengembalikan indeks teks di mana hamming distancenya terkecil.

b) controllers

Berisi pustaka yang digunakan untuk memproses data yang dikirimkan dan/atau diterima/dikirimkan oleh *frontend* dan basis data.

(a) hasilController.go

- Prosedur GetHasilByQuery

Prosedur menerima *query* dari *frontend* dan mengirimkan data sesuai *query* dari basis data menuju *frontend*. Pertama, prosedur akan memvalidasi *query* menggunakan pustaka Regex. Kemudian, prosedur akan mencari data yang dicari sesuai *query* dan mengirimkannya pada *frontend*.

- Fungsi AddHasil

Prosedur menerima untaian DNA yang akan diperiksa kecocokannya menggunakan pustaka StringMatching dan kemudian memasukkan hasilnya pada basis data. Untaian DNA juga divalidasi menggunakan pustaka Regex sebelum dilakukan pencocokan.

(b) penyakitController.go

- Prosedur AddPenyakit

Prosedur menerima untaian DNA

c) database

Berisi pustaka yang digunakan untuk melakukan konfigurasi koneksi dengan basis data.

(a) mysql.go

- Fungsi Connect

Mengembalikan koneksi basis data.

d) model

Berisi pustaka tipe data yang digunakan untuk pemrosesan data.

(a) hasil.go

- Tipe data Hasil
Berisi atribut Tanggal, NamaPengguna, NamaPenyakit, Persentase, dan Hasil.
- Tipe data ResponseHasil
Berisi atribut Status, Message, dan Data yang merupakan *array* dari Hasil.

(b) penyakit.go

- Tipe data Penyakit
Berisi atribut NamaPanyakit dan SequenceDNA.
- Tipe data ResponseHasil
Berisi atribut Status, Message, dan Data yang merupakan *array* dari Penyakit.

e) server.go

Berisi program yang menghubungkan API dan basis data.

iv) frontend

(a) src

(i) Component

Berisi fungsionalitas Button dan Form yang digunakan untuk semua Button dan Form yang ada pada web page

(ii) Page

Berisi kode halaman-halaman yang ada web page. Page tersebut terdiri dari:

- Home
- AddDNA

- CheckDNA
- FindDNA

(iii) Templates

Berisi kode template yang digunakan pada semua page yang ada di web page.

Template ini berisi navbar dan animasi transisi tiap page

(iv) Images

Berisi foto yang digunakan sebagai background dan watermark pada navbar

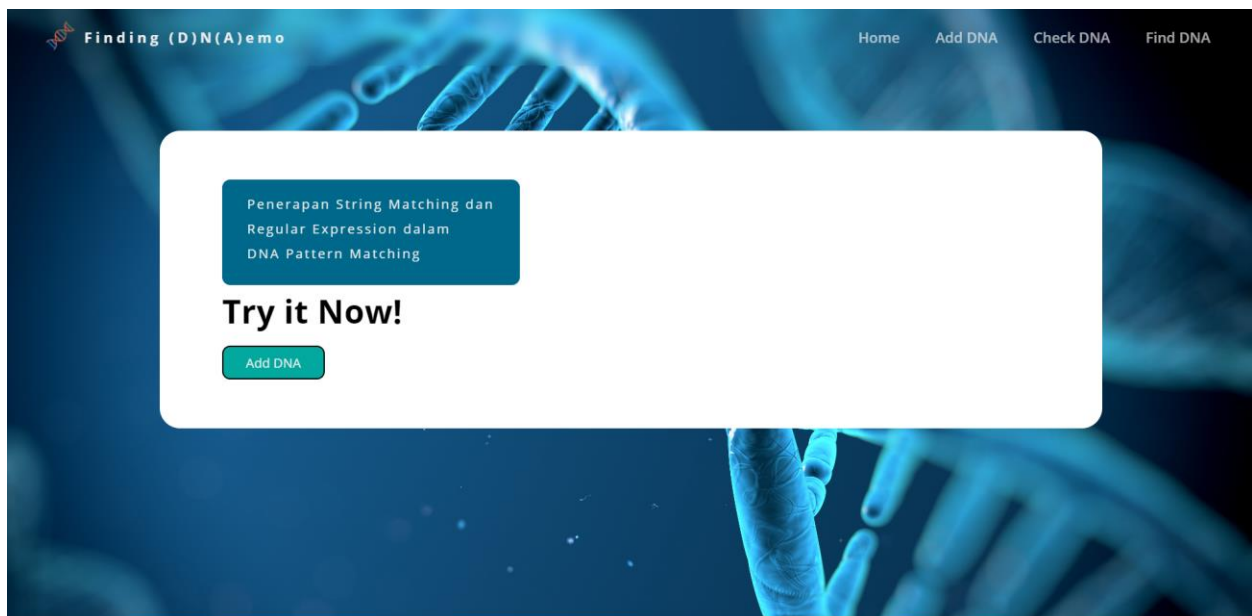
(v) Styles

Berisi kode styling untuk web page

II. Tata Cara Penggunaan Program

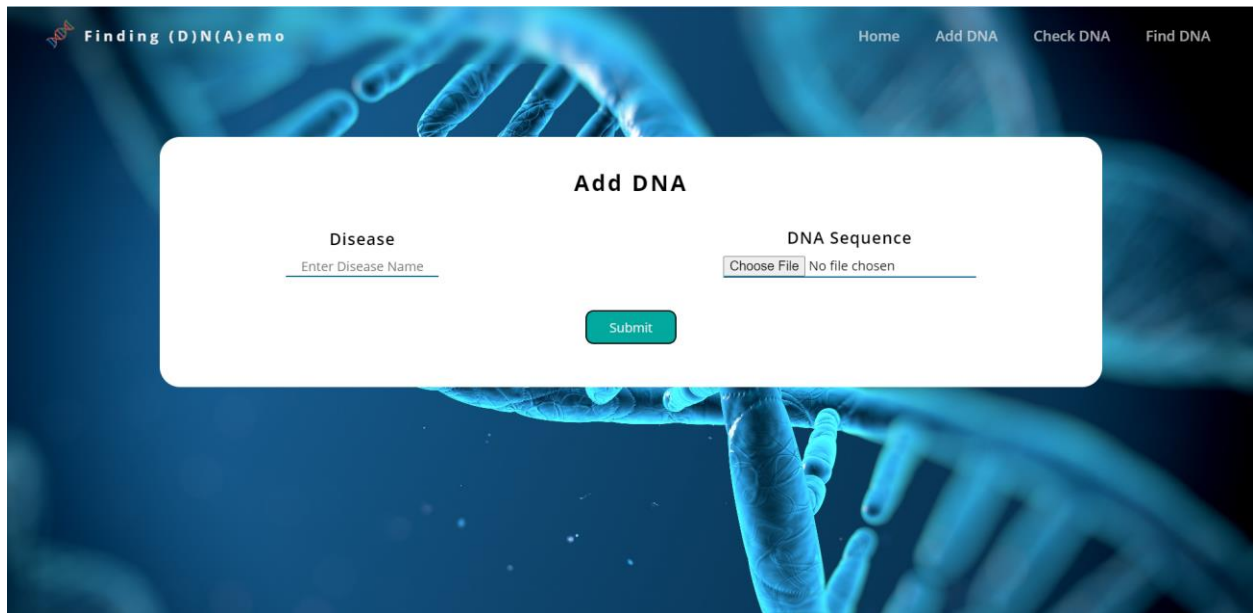
Aplikasi web tersedia pada <https://finding-dnaemo.netlify.app/>.

Berikut adalah halaman utama web:



Pada pojok kanan atas halaman terdapat beberapa *button* yang dapat digunakan untuk berpindah halaman. *Button* Add DNA pada bagian tengah halaman juga bisa digunakan untuk berpindah ke halaman Add DNA.

Berikut adalah halaman Add DNA:



Finding (D)N(A)emo Home Add DNA Check DNA Find DNA

Add DNA

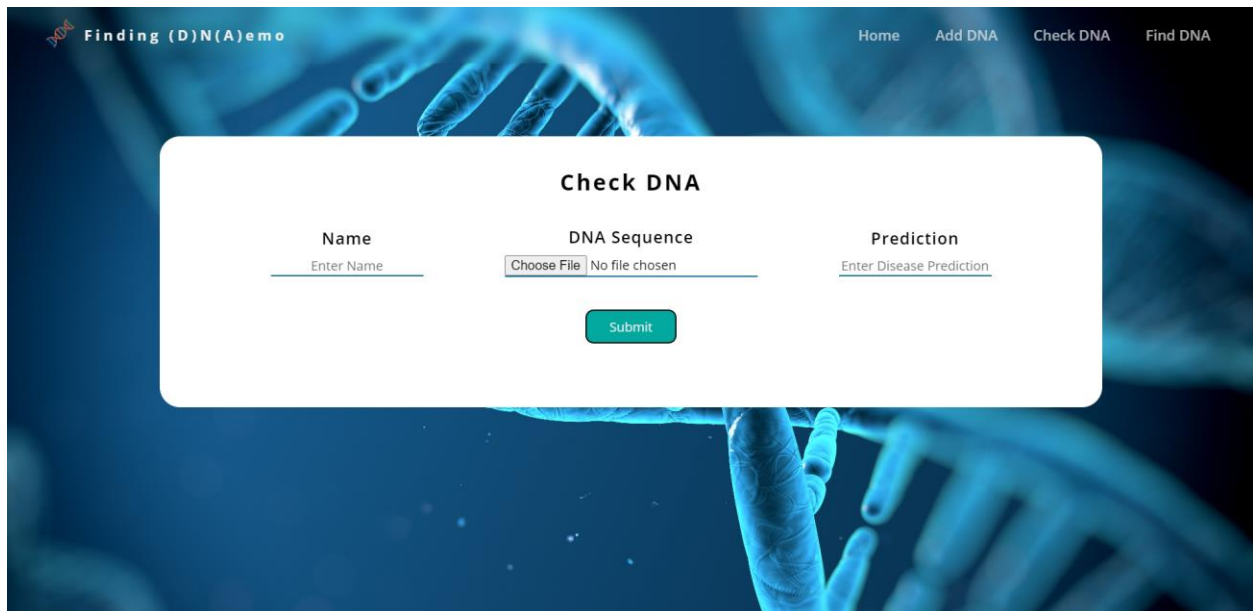
Disease
Enter Disease Name

DNA Sequence
Choose File No file chosen

Submit

Halaman Add DNA digunakan untuk menambahkan penyakit baru ke dalam aplikasi. Pada textbox disease, dapat dimasukkan nama penyakit dan pada file selector dapat dimasukkan file yang berisi sekuens DNA penyakit tersebut. Button submit digunakan untuk melakukan submisi masukan.

Berikut adalah halaman Check DNA:



Finding (D)N(A)emo Home Add DNA Check DNA Find DNA

Check DNA

Name
Enter Name

DNA Sequence
Choose File No file chosen

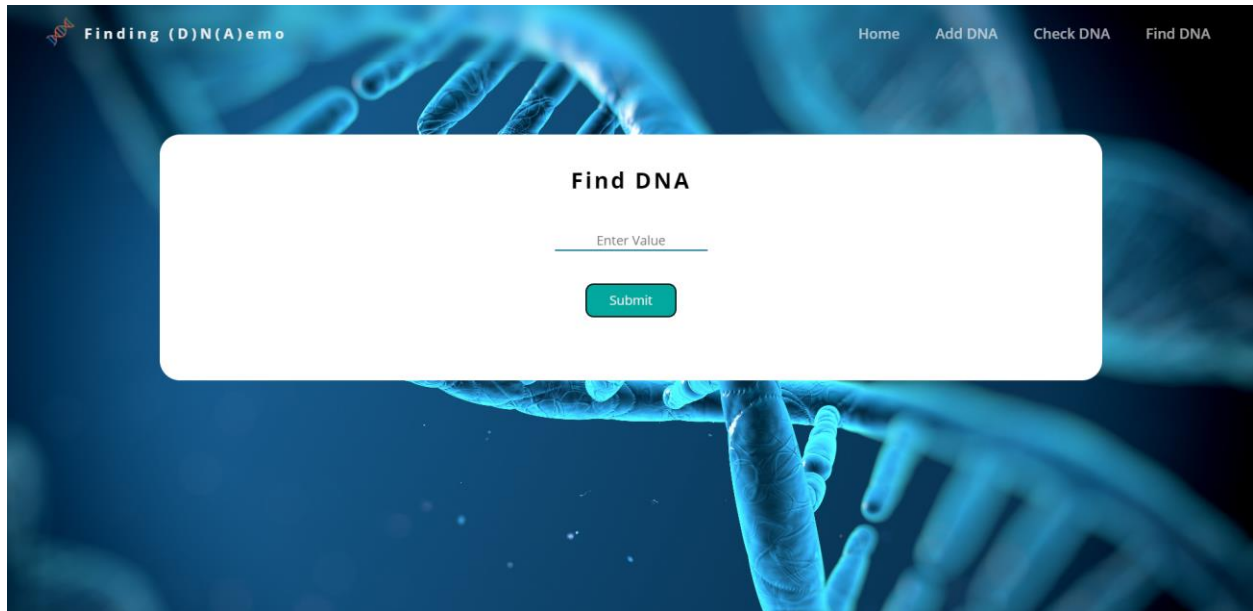
Prediction
Enter Disease Prediction

Submit

Halaman Check DNA digunakan untuk melakukan pemeriksaan terhadap seseorang terhadap penyakit tertentu. Pada textbox name, dapat dimasukkan nama orang yang ingin diperiksa, pada file selector dapat dimasukkan file yang berisi sekuens DNA orang tersebut, dan

pada textbox prediction, dapat dimasukkan nama penyakit yang ingin diperiksa. Button submit digunakan untuk melakukan pengecekan orang tersebut terhadap masukan nama penyakit.

Berikut adalah halaman Find DNA:

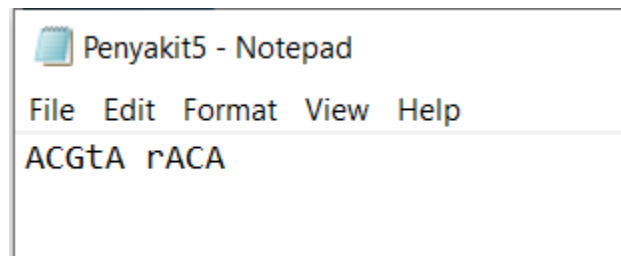


Halaman Find DNA digunakan untuk melakukan pencarian terhadap hasil pemeriksaan. Textbox Find DNA dapat menerima *query* pencarian berupa tanggal dengan format dd-mm-yyyy, nama penyakit, atau gabungan keduanya. Button submit digunakan untuk melakukan pencarian berdasarkan *query* masukan.

III. Pengujian

Berikut berbagai skenario fitur-fitur yang ada:

- a. Penambahan penyakit dengan sekuens DNA tidak valid



finding-dnaemo.netlify.app/add-dna

Home Add DNA Check DNA Find DNA

Add DNA

Disease
Penyakit5

DNA Sequence
Choose File Penyakit5.txt

Submit

Please Check Your Input

b. Penambahan penyakit dengan sekuens DNA valid

Penyakit5 - Notepad

File Edit Format View Help

ACGTA

finding-dnaemo.netlify.app/add-dna

Home Add DNA Check DNA Find DNA

Add DNA

Disease
Penyakit5

DNA Sequence
Choose File Penyakit5.txt

Submit

Success Adding New Disease Sequence

c. Penambahan penyakit dengan nama yang sudah ada

finding-dnaemo.netlify.app/add-dna

Home Add DNA Check DNA Find DNA

Add DNA

Disease: Penyakit5

DNA Sequence: Choose File | Penyakit5.txt

Submit

Please Check Your Input

d. Pemeriksaan dengan sekuens DNA orang tidak valid

Orang8 - Notepad

File Edit Format View Help

ACAACGT ACCACAACCA

finding-dnaemo.netlify.app

Home Add DNA Check DNA Find DNA

Check DNA

Name: Orang8

DNA Sequence: Choose File | Orang8.txt

Prediction: Penyakit5

Submit

Please Check Your Input

e. Pemeriksaan dengan sekuens nama penyakit yang tidak tercatat

Orang8 - Notepad

File Edit Format View Help

ACAACGTACCACAACCA

finding-dnaemo.netlify.app/check-dna

Finding (D)N(A)emo Home Add DNA Check DNA Find DNA

Check DNA

Name	DNA Sequence	Prediction
Orang8	Choose File Orang8.txt	Penyakit6

Submit

Please Check Your Input

f. Pemeriksaan dengan sekuens DNA valid dan nama penyakit tercatat

finding-dnaemo.netlify.app/check-dna

Finding (D)N(A)emo Home Add DNA Check DNA Find DNA

Check DNA

Name	DNA Sequence	Prediction
Orang8	Choose File Orang8.txt	Penyakit5

Submit

Result

2022-04-29 - Orang8 - Penyakit5 - 100% - True

g. Melakukan pencarian dengan tanggal tidak valid

The image displays two screenshots of a web application titled "Finding (D)N(A)emo". The application has a navigation bar with links: Home, Add DNA, Check DNA, and Find DNA. The background features a blue-tinted image of a DNA double helix and a hand. The main content area is a white rounded rectangle titled "Find DNA".

The top screenshot shows the form with the date "30-02-2022" entered in the input field. Below the input field is a green "Submit" button. At the bottom of the form, the text "Please Check Your Input" is displayed.

The bottom screenshot shows the same form, but with the date "01-1-202 Penyakit5" entered in the input field. The "Submit" button and the "Please Check Your Input" message are also present.

h. Melakukan pencarian dengan tanggal valid

The screenshot shows a web browser at the URL `finding-dnaemo.netlify.app/find-dna`. The page has a dark blue background with a DNA helix and fingerprints. The navigation bar includes 'Home', 'Add DNA', 'Check DNA', and 'Find DNA'. The main content area is titled 'Find DNA' and contains a search input field with the text '29-04-2022'. Below the input is a green 'Submit' button. The results are displayed as a list of six items, each in a rounded rectangle:

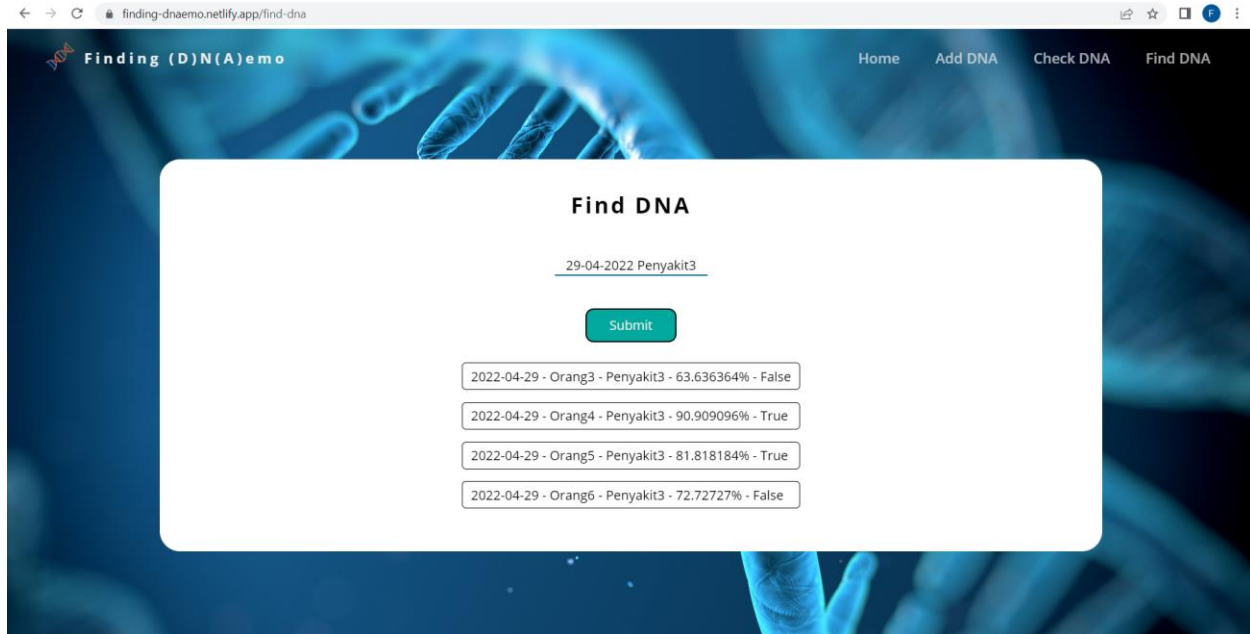
- 2022-04-29 - Orang3 - Penyakit3 - 63.636364% - False
- 2022-04-29 - Orang4 - Penyakit3 - 90.909096% - True
- 2022-04-29 - Orang5 - Penyakit3 - 81.818184% - True
- 2022-04-29 - Orang6 - Penyakit3 - 72.72727% - False
- 2022-04-29 - Orang7 - Penyakit4 - 80% - True
- 2022-04-29 - Orang8 - Penyakit5 - 100% - True

i. Melakukan pencarian dengan nama penyakit

The screenshot shows the same web application with the search input field containing the text 'Penyakit5'. After clicking the 'Submit' button, the results list now only contains one item:

- 2022-04-29 - Orang8 - Penyakit5 - 100% - True

j. Melakukan pencarian dengan tanggal dan nama penyakit



IV. Analisis Pengujian

- a. Penambahan penyakit dengan sekuens DNA tidak valid
Bisa dilihat dari sequence DNA “ACGtA rACA”, terdapat spasi dan huruf kecil pada input. Alhasil, hasil dari test ini ialah ditolak dan akan menghasilkan text error.
- b. Penambahan penyakit dengan sekuens DNA valid
Bisa dilihat dari sequence DNA “ACGTA”, input sudah memenuhi regex berupa “hanya boleh berisi huruf kapital dari ACGTA”. Alhasil, program akan menginput nama DNA dan sequencenya ke database dan menghasilkan text success.
- c. Penambahan penyakit dengan nama yang sudah ada
Bisa dilihat dari nama DNA yang sudah ada, hasil tes akan ditolak dan program akan menghasilkan text error.
- d. Pemeriksaan dengan sekuens DNA orang tidak valid
Bisa dilihat dari sequence DNA “ACAACGT ACCACAACCA”, terdapat spasi pada input. Alhasil, hasil dari test ini ialah ditolak dan akan menghasilkan text error.
- e. Pemeriksaan dengan sekuens nama penyakit yang tidak tercatat

Bisa dilihat dari sequence DNA “ACAACGTACCACAACCA”, sequence DNA sudah valid, namun nama prediksi penyakit tidak terdapat pada database, sehingga, test akan ditolak dan akan menghasilkan text error.

- f. Pemeriksaan dengan sekuens DNA valid dan nama penyakit tercatat

Bisa dilihat dari sequence DNA yang sudah valid dan nama penyakit yang tercatat, program akan menghasilkan hasil test berupa tanggal pengecekan, nama pengecek, nama penyakit prediksi, persen kemiripan, dan apakah mirip atau tidak (True jika persen kemiripan > 80%).

- g. Melakukan pencarian dengan tanggal tidak valid

Karena tanggal tidak valid, program akan mengeluarkan text error.

- h. Melakukan pencarian dengan tanggal valid

Karena tanggal sudah valid, program akan mengeluarkan *history* pengecekan dari database dengan tanggal yang sama dengan pencarian

- i. Melakukan pencarian dengan nama penyakit

Karena nama penyakit sudah valid, program akan mengeluarkan *history* pengecekan dari database dengan nama penyakit yang sama dengan pencarian

- j. Melakukan pencarian dengan tanggal dan nama penyakit

Karena nama penyakit dan tanggal sudah valid, program akan mengeluarkan *history* pengecekan dari database dengan nama penyakit dan tanggal yang sama dengan pencarian.

BAB V

KESIMPULAN DAN SARAN

V. Kesimpulan

Algoritma *string matching* dapat dimanfaatkan untuk melakukan pencocokan DNA penyakit dengan DNA seseorang untuk menunjukkan bahwa orang tersebut menderita penyakit tertentu. KMP dan Boyer-Moore Algorithm merupakan contoh dari algoritma *string matching* yang mencari dengan tepat kemunculan pola pada teks. KMP dan Boyer-Moore memiliki efisiensi yang lebih baik dibandingkan dengan *brute force string matching* karena melakukan pergeseran pola dengan lebih baik. Sementara itu, Hamming Distance Algorithm merupakan contoh *string matching* yang tidak mencari dengan tepat kemunculan pola pada teks, tetapi menghasilkan jarak minimum pola dengan teks. Melalui Hamming Distance Algorithm, dapat diperoleh tingkat kemiripan pola dengan teks. Di sisi lain, Regular Expression dapat dimanfaatkan untuk melakukan validasi masukan agar masukan sesuai dengan nilai yang valid.

VI. Saran

Algoritma *hamming distance* yang diimplementasikan pada program menggunakan pendekatan *brute force* sehingga dapat dicari pendekatan lain atau alternatif algoritma lain yang dapat melakukan deteksi kemiripan pola dengan teks agar program menjadi lebih efisien.

VII. Refleksi

Melalui tugas besar 3 ini, penulis mampu memahami dengan lebih baik algoritma *string matching* khususnya KMP, Boyer-Moore, dan Hamming Distance Algorithm beserta pemanfaatannya dalam pembuatan aplikasi DNA Pattern Matching. Selain itu, penulis juga dapat lebih memahami mengenai pembangunan aplikasi web.

DAFTAR PUSTAKA

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

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>

<https://www.tutorialspoint.com/what-is-hamming-distance>

LAMPIRAN

Link repository:

https://github.com/msyahrulsp/Tubes3_13520112

Link website:

<https://finding-dnaemo.netlify.app/>

Link video demo:

<https://youtu.be/VuQxvU0SOro>