

LAPORAN TUGAS BESAR III
IF2211 STRATEGI ALGORITMA
Penerapan String Matching dan Regular Expression
dalam DNA Pattern Matching

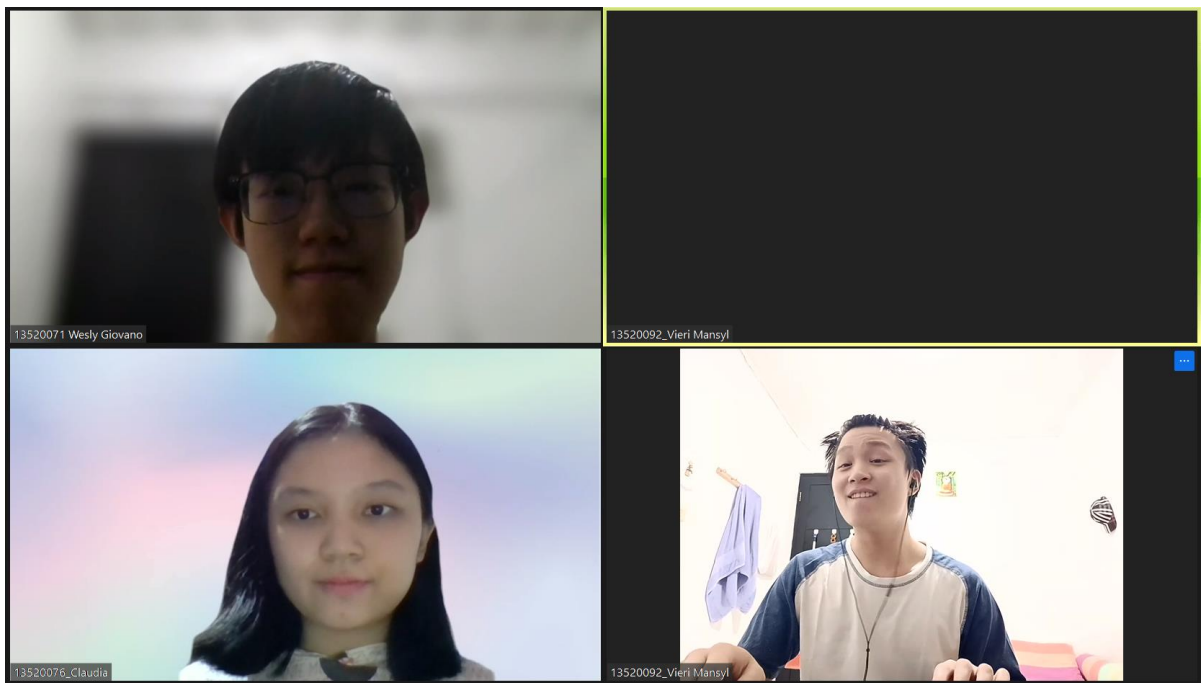
Disusun oleh :

Kelompok 07 – SANGCI REBORN

13520071 – Wesly Giovano

13520076 – Claudia

13520092 – Vieri Mansyl



PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2022

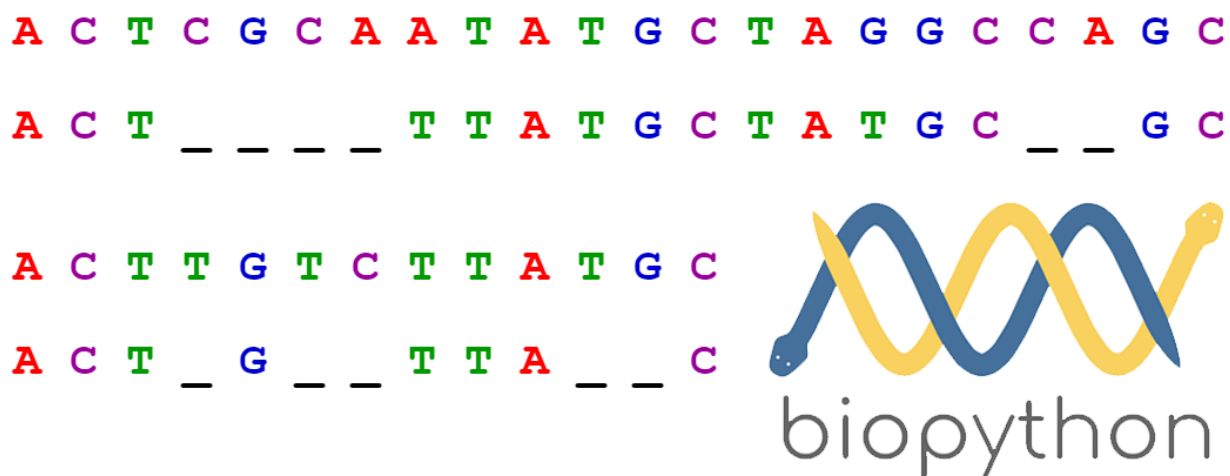
DAFTAR ISI

- BAB I DESKRIPSI TUGAS 1
- BAB II LANDASAN TEORI.....3
 - 2.1 Algoritma *String Matching*..... 3
 - 2.2 Penjelasan singkat mengenai aplikasi *website* yang dibangun..... 5
- BAB III ANALISIS PEMECAHAN MASALAH 6
 - 3.1 Langkah penyelesaian masalah setiap fitur 6
 - 3.2 Fitur fungsional dan arsitektur aplikasi *website* yang dibangun..... 7
- BAB IV IMPLEMENTASI dan PENGUJIAN..... 9
 - 4.1 Spesifikasi teknis program..... 9
 - 4.2 Tata cara penggunaan program..... 10
 - 4.3 Hasil pengujian 11
 - 4.4 Analisis hasil pengujian..... 17
- BAB V KESIMPULAN, SARAN, DAN KOMENTAR..... 17
 - 5.1 Kesimpulan 17
 - 5.2 Saran dan Komentar 17
- LAMPIRAN..... 19
- DAFTAR PUSTAKA 20

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*.



Gambar 1. Ilustrasi Sekuens DNA

Sumber : <https://towardsdatascience.com/pairwise-sequence-alignment-using-biopython-d1a9d0ba861f>

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

2.1 Algoritma *String Matching*

A. Algoritma *Knuth-Morris-Pratt* (KMP)

Algoritma Knuth-Morris-Pratt (KMP) merupakan salah satu algoritma *pattern matching* pada suatu teks dengan pembacaan dari kiri ke kanan – seperti dengan *naïve pattern matching* (pembacaan secara *brute force*) tetapi lebih ‘pintar’. Ide dasar algoritma KMP ialah pada setiap pembacaan suatu string, ketika ditemukan ketidakcocokan setelah beberapa kecocokan sebelumnya, algoritma telah mengetahui beberapa karakter berikutnya dalam teks, sehingga dengan informasi tersebut, algoritma dapat menghindari pencocokan karakter yang terhadap karakter yang pasti telah cocok.

Secara kasar, algoritma KMP ialah sebagai berikut.

1. Diberikan sebuah string text T dan string pattern P
2. Selama pembacaan, ketika terjadi *mismatch* antara suatu karakter di antara kedua string ini pada T[i] dan P[j], maka dilakukan *shifting* sebesar *prefix* terbesar dari P[0..k] yang juga merupakan *suffix* dari P[1..k] untuk $k = j - 1$.
 - a. Untuk mendapatkan posisi j, digunakan suatu fungsi yang dikenal dengan fungsi pinggiran (*border function*) $b(k)$. Fungsi pinggiran bertujuan untuk mendapatkan ukuran terbesar dari prefix P[0..k] yang juga *suffix* dari P[1..k].

Kelebihan dari algoritma KMP ialah algoritma ini tidak pernah melakukan pembacaan ke belakang terhadap teks T, sehingga algoritma ini bagus dalam memproses file teks dengan ukuran yang sangat besar. Hal ini juga mengakibatkan algoritma KMP tidak cocok digunakan ketika variasi dari alfabet semakin meningkat karena berkemungkinan besar terjadi *mismatch* saat *searching* dilakukan.

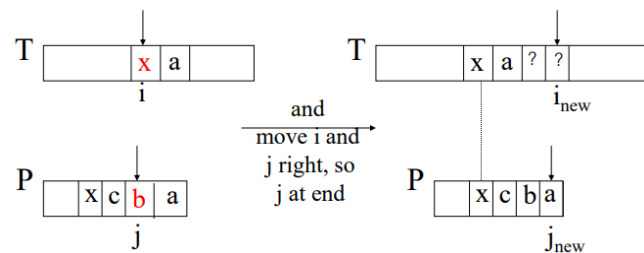
Kompleksitas waktu dari algoritma KMP ialah $O(m+n)$ untuk $O(m)$ sebagai kompleksitas menghitung fungsi pinggiran dan $O(n)$ untuk pencarian string.

B. Algoritma *Boyer-Moore* (BM)

Algoritma *Boyer-Moore* (BM) merupakan salah satu algoritma *pattern matching* pada suatu teks dengan pembacaan dari kanan ke kiri. Algoritma ini dapat diimplementasikan dalam dua jenis teknik pembacaan, yaitu *looking-glass technique* dan *character-jump technique*. Teknik *looking-glass* merupakan teknik *pattern matching* terhadap teks dengan pencocokan dimulai dari bagian akhir pada *pattern* P terhadap teks T, sedangkan teknik *character-jump* merupakan teknik *pattern matching* yang menangani *mismatch* pada saat pencocokan sehingga terjadi pergeseran *pattern* P terhadap teks T.

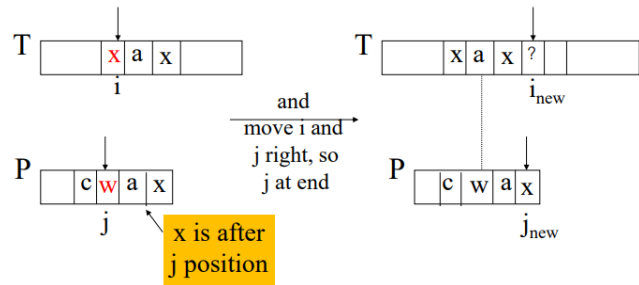
Pada teknik *character-jump*, terdapat 3 kasus yang mungkin terjadi, yaitu sebagai berikut.

1. Ketika terjadi *mismatch* pada $T[i]$ yang mengandung nilai X dan terdapat nilai X di dalam dan di sisi kiri posisi *pattern* $P[j]$, maka posisi *last occurrence pattern* P yang nilai X disejajarkan dengan teks $T[i]$.



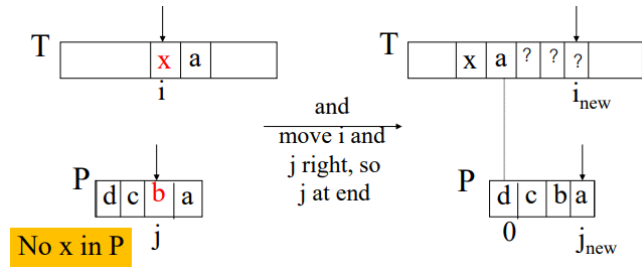
Gambar 2.1.B.1 Kasus pertama dalam Teknik Character-Jump

2. Ketika terjadi *mismatch* pada $T[i]$ yang mengandung nilai X dan terdapat nilai X di dalam namun di sisi kanan dari posisi *pattern* $P[j]$, maka *pattern* P digeser tepat 1 karakter ke kanan sehingga posisi $j = i+1$.



Gambar 2.1.B.2 Kasus kedua dalam Teknik Character-Jump

3. Ketika tidak terjadi kedua kasus diatas, maka *pattern* P akan digeser sehingga posisi $P[0]$ sejajar dengan teks $T[i+1]$.



Gambar 2.1.B.3 Kasus ketiga dalam Teknik Character-Jump

Seperti algoritma KMP yang memiliki fungsi Batasan, pada algoritma BM, terdapat suatu fungsi yang dikenal sebagai fungsi *last occurrence*. Fungsi *last occurrence* bertujuan untuk mencatat kemunculan terakhir dari tiap alfabet pada *pattern* P .

Kompleksitas waktu dari algoritma BM ialah $O(mn + A)$ untuk A = variasi alfabet pada teks. Algoritma BM cocok digunakan ketika ukuran alfabet besar, namun pelan ketika ukuran alfabet kecil.

C. String Matching dengan Regular Expression (Regex)

Regular Expression merupakan sebuah ekspresi berbasis karakter yang digunakan untuk pencarian suatu *pattern* pada suatu teks. *Pattern* yang dimaksud dapat berupa pencarian digit secara umum, huruf, dan sejenisnya, atau juga jumlah huruf yang diinginkan berjumlah sebanyak

jumlah tertentu. *Pattern* dalam Regex bersifat cukup general, sehingga umum digunakan dalam validasi input, misalnya validasi email.

D. *Hamming Distance*

Algoritma *Hamming distance* adalah algoritma yang membandingkan dua buah string dalam hal perbedaan pada setiap karakter/bitnya. Secara formal, jarak antara dua buah *string* dengan metode ini adalah jumlah karakter yang berbeda pada posisinya. *Hamming distance* baik digunakan untuk perbandingan dua buah *string* yang domain karakternya cukup kecil, misalnya bit 0/1 dan basa nukleotida pada rantai DNA/RNA.

2.2 Penjelasan singkat mengenai aplikasi *website* yang dibangun

Website yang telah dibangun merupakan sebuah web sederhana yang memberikan fitur bagi *user* untuk melakukan tes DNA *user* terhadap DNA *pattern* dari penyakit-penyakit yang telah tercatat di dalam *cloud database* pada *website*. Website menyediakan fitur bagi *user* untuk memasukkan *sequence* DNA baru dari sebuah penyakit dan akan disimpan ke dalam *cloud database*. Tidak hanya itu, *user* juga dapat melakukan pencarian terhadap *log* tes DNA yang telah dilakukan berdasarkan tanggal dilakukan tes DNA dan/atau jenis penyakit yang diuji coba.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah penyelesaian masalah setiap fitur

A. Menambahkan penyakit baru

- Melakukan validasi terhadap nama penyakit baru yang diinput melalui *query* pada *database*.
- Apabila nama penyakit tidak ada di dalam *database*, maka dilakukan validasi terhadap *sequence* DNA dari inputan *user* menggunakan Regex.
- Apabila *sequence* DNA dari inputan *user* telah valid, maka inputan nama penyakit beserta *sequence* DNA akan dicatat ke dalam *database*.
- Apabila *sequence* DNA dari inputan *user* invalid berdasarkan pembacaan Regex, maka dikeluarkan pesan gagal dan inputan penyakit baru tidak dimasukkan ke dalam *database*.
- Apabila nama penyakit telah ada di dalam *database*, maka dikeluarkan pesan gagal dan inputan penyakit baru tidak dimasukkan ke dalam *database*.

B. Melakukan tes DNA terhadap suatu penyakit

- Melakukan validasi terhadap prediksi penyakit melalui *query* pada *database*.
- Apabila inputan prediksi penyakit tidak ada di dalam *database*, maka dikeluarkan pesan error.
- Jika inputan prediksi penyakit ada di dalam *database*, maka dilakukan validasi terhadap *sequence* DNA dari inputan *user* menggunakan Regex.
- Apabila *sequence* DNA dari inputan *user* invalid berdasarkan pembacaan Regex, maka dikeluarkan pesan error.
- Apabila *sequence* DNA valid, maka dijalankan algoritma *string matching* dengan menggunakan algoritma KMP dari *sequence* DNA *user* terhadap *sequence* DNA dari prediksi penyakit, yang mana *sequence* DNA dari prediksi penyakit bertindak sebagai *pattern*.
 - Dari algoritma KMP dan algoritma BM yang telah diimplementasikan, digunakan algoritma KMP untuk melakukan *string matching* dengan pertimbangan bahwa ukuran variasi alfabet pada *sequence* DNA sangat kecil, yaitu sebesar 4 satuan, sehingga efisiensi algoritma KMP jauh lebih baik dibanding ketika menggunakan algoritma BM.
- Hasil dari pencocokan *string matching* pada *sequence* DNA inputan *user* terhadap *sequence* DNA dari prediksi penyakit yang diinput akan ditampilkan pada *website* diikuti dengan nama pasien, serta hasil dari *string matching*.
- Sebagai implementasi dari **bonus**, dilakukan perhitungan kemiripan antara *sequence* DNA inputan *user* terhadap *sequence* DNA dari prediksi penyakit. Hasil kemiripan akan ditampilkan pada *website*.

- Hasil dari *string matching*, yaitu tanggal pengambilan tes DNA, nama pasien, *sequence* DNA, kemiripan, serta hasil tes DNA akan dicatat ke dalam *database*.

C. Melakukan *searching* berdasarkan tanggal melakukan tes DNA dan/atau nama penyakit

- Melakukan validasi terhadap inputan tanggal dan/atau nama penyakit menggunakan Regex.
- Apabila tanggal dan/atau nama penyakit invalid berdasarkan Regex, maka tidak ada hasil yang akan keluar.
- Apabila inputan telah valid, maka dilakukan *searching* menggunakan *query* pada *database*.
- Hasil dari *query* akan ditampilkan pada *website*.

3.2 Fitur fungsional dan arsitektur aplikasi *website* yang dibangun

Fitur-fitur fungsional dari *website* yang dibangun ialah sebagai berikut.

1. Menambahkan nama penyakit serta *sequence* DNA-nya ke dalam relasi *penyakit* pada *database*.
2. Melakukan tes DNA berdasarkan *sequence* DNA inputan *user* dengan *sequence* DNA dari ‘prediksi’ penyakit.
3. Menambahkan hasil dari tes DNA berupa tanggal pengambilan tes DNA, nama pasien, *sequence* DNA, kemiripan, serta hasil tes DNA ke dalam relasi *logPasien* pada *database*.
4. Menampilkan *log* tes DNA berdasarkan tanggal pengambilan tes DNA dan/atau nama penyakit yang diambil dari relasi *logPasien* pada *database*.
5. Menampilkan pesan *error* ketika terjadi kesalahan saat melakukan inputan penyakit baru, baik karena nama penyakit masih kosong, ataupun isi dari *text file* kosong atau mengandung alfabet selain A,T,C,G.
6. Menampilkan pesan *error* ketika terjadi kesalahan saat melakukan inputan tes DNA, baik karena nama pasien masih kosong, atau nama ‘prediksi’ penyakit masih kosong, ataupun isi dari *text file* kosong atau mengandung alfabet selain A,T,C,G.
7. Menampilkan pesan gagal saat melakukan inputan tes DNA karena tidak adanya nama penyakit pada *database* yang sesuai dengan nama ‘prediksi’ penyakit.

Arsitektural dari *website* yang dibangun dibagi menjadi 3 bagian utama, yaitu :

1. *Frontend*

Frontend pada *website* yang telah dibangun menggunakan bahasa pemrograman *javascript* dengan *library React*. Website dibagi menjadi 4 laman dengan 1 laman utama yang bertindak sebagai *homepage*, serta 3 laman lainnya sebagai realisasi dari spesifikasi tugas besar, yaitu *Input Disease* (diatur pada *InputDisease.js*) , *DNA Test* (diatur pada *DNATest.js*), dan *Search Test Result* (diatur pada *Searching.js*). Terdapat juga beberapa komponen yang digunakan untuk menerapkan algoritma *string matching* pada folder *component*, seperti *DiseaseForm* untuk meng-*handle* inputan penyakit baru, *Form* untuk

meng-handle inputan tes DNA baru, dan *SearchBar* untuk meng-handle inputan *searching* pada log tes DNA.

2. *Backend*

Backend pada *website* yang telah dibangun menggunakan bahasa pemrograman *GoLang* dengan *framework Gin* untuk menghubungkan dengan *frontend*.

3. *Database*

Website yang telah dibangun menggunakan *cloud hosting database* dengan DBMS *MySQL* sebagai media penyimpanan data penyakit serta log dari tes DNA. *Database* dihubungkan dengan *backend* sehingga manipulasi informasi selalu terjadi pada *backend* terlebih dahulu sebelum *frontend* dapat meminta/memberikan informasi.

BAB IV

IMPLEMENTASI dan PENGUJIAN

4.1 Spesifikasi teknis program

A. Struktur data

Terdapat 3 *struct* yang digunakan untuk menampung isi data dari relasi pada *database*, yaitu:

```
// khusus untuk tipe data Pasien digunakan untuk menampung nama pasien serta
sequence DNA dari pasien serta nama 'prediksi' penyakit untuk menampung
inputan dari user dari website.
type Pasien struct {
    NamaPasien    string `json:"namaPasien"`
    DNASequences string `json:"dnaSequence"`
    NamaPenyakit  string `json:"namaPenyakit"`
}

type LogPasien struct {
    Tgl          string `json:"tanggal"`
    NamaPasien   string `json:"namaPasien"`
    NamaPenyakit string `json:"namaPenyakit"`
    Kemiripan    float64 `json:"kemiripan"`
    Hasil        bool    `json:"hasil"`
}

type Penyakit struct {
    NamaPenyakit string `json:"namaPenyakit"`
    DNASequences string `json:"dnaSequence"`
}
```

Selain itu, pengambilan dan pengiriman data ke dari inputan *user* pada *website* ke program menggunakan struktur data JSON (*Javascript Object Notation*) agar data dapat di-*passing* antara *Frontend* dan *Backend*.

B. Fungsi dan Prosedur

Berikut merupakan fungsi dan prosedur yang diterapkan pada *Backend*.

Fungsi / Prosedur	Deskripsi singkat
isDNAValid(string) → bool	Melakukan validasi terhadap sequence DNA dengan menggunakan regex.
isDNAMatched(string,string) →bool, float64	Melakukan <i>string matching</i> dengan algoritma KMP dan menghitung kemiripan menggunakan <i>Hamming Distance</i> diantara <i>sequence</i> DNA pasien dengan <i>sequence</i> DNA penyakit.
ParsePrediction(string) → bool, time, string	Melakukan <i>parsing</i> terhadap inputan dari <i>user</i> saat melakukan <i>searching</i> terhadap log tes DNA. Fungsi ini juga melakukan validasi terhadap inputan tanggal menggunakan regex. Fungsi ini mengembalikan waktu, nama penyakit, serta sebuah <i>boolean</i> yang menunjukkan kevalidasian inputan dari <i>user</i> .

KMP(string,string) → int	Implementasi <i>string matching</i> dengan algoritma KMP. Mengembalikan nilai posisi kemiripan diantara <i>sequence</i> DNA pasien dengan <i>sequence</i> DNA dari penyakit terkait.
Border(string) → []int	Fungsi batasan sebagai implementasi <i>border function</i> pada algoritma KMP.
BoyerMoore(string,string) →int	Implementasi <i>string matching</i> dengan algoritma BM. Mengembalikan nilai posisi kemiripan diantara <i>sequence</i> DNA pasien dengan <i>sequence</i> DNA dari penyakit terkait.
lastOccurence(string) → []int	Fungsi <i>last occurrence</i> sebagai implementasi dari fungsi <i>last occurrence</i> pada algoritma BM.
HammingDist (string, string) →float64 , error	Menghitung <i>Hamming Distance</i> diantara dua string.
SubstringHammingDist (string,string) → float64	Melakukan kalkulasi kemiripan antara dua <i>sequence</i> DNA menggunakan perhitungan <i>Hamming Distance</i> .
isDateValid(int,int,int) →bool	Melakukan validasi dari tanggal (mengikuti aturan tahun romawi).
MonthNameToInt(string)→int	Mengembalikan bulan berdasarkan kemungkinan masukan bulan dalam bentuk integer (1 s.d. 12).
DateToString(int,Month,int) →string	Mengembalikan tahun,bulan,dan hari dalam bentuk string.
DateToYYYYMMDD(int,Month,int) →string	Mengembalikan tahun,bulan,dan hari dalam bentuk string dengan format YYYY-MM-DD.
postDisease(*gin.Context)	Melakukan <i>post</i> serta menginput data penyakit baru ke dalam <i>database</i> .
postLogs(*gin.Context)	Melakukan <i>post</i> serta menginput data hasil tes DNA baru ke dalam <i>database</i> .
getLogs(*gin.Context)	Melakukan <i>get</i> serta me- <i>retrieve</i> hasil tes DNA berdasarkan masukan tanggal dan / atau nama penyakit dari <i>user</i> dari <i>database</i> .
getDiseasebyName(string) →Penyakit	Mengembalikan data penyakit berdasarkan nama penyakit di dalam <i>database</i> dalam bentuk <i>struct</i> Penyakit.

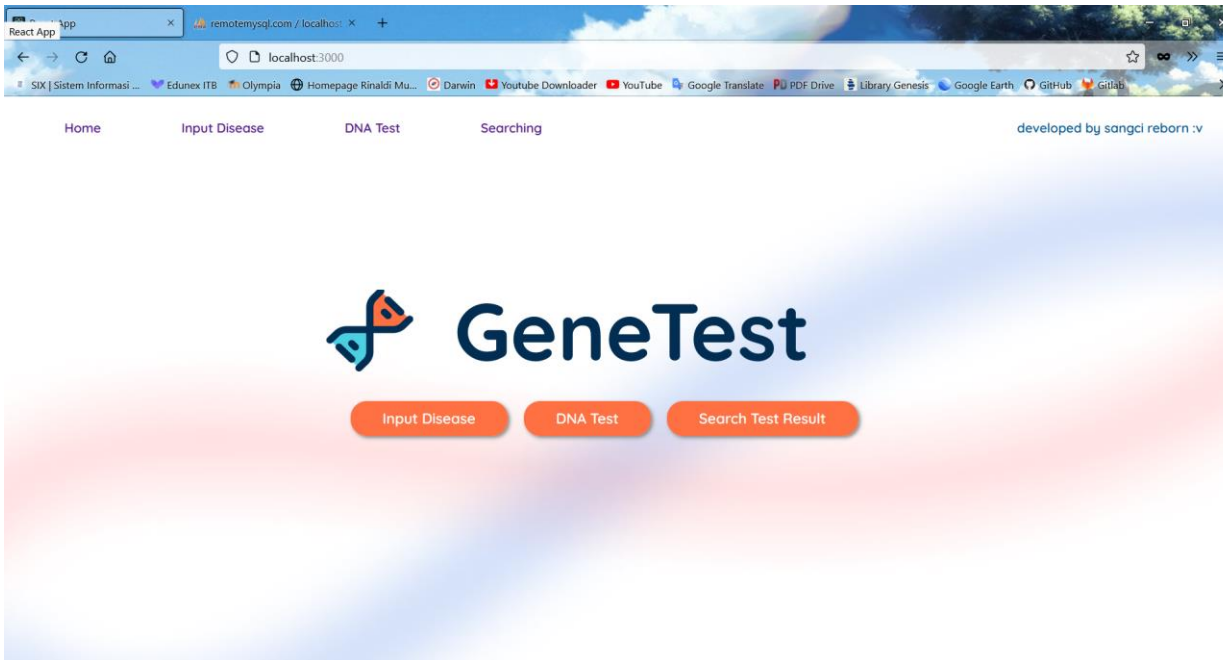
4.2 Tata cara penggunaan program

- *User* dapat menggunakan program website dengan mengikuti instruksi yang tertera pada README.

- Saat memasuki laman *website*, program akan memberikan tampilan *homepage* kepada *user*. *User* dapat menggunakan 3 fitur sebagai berikut.
 - Untuk menggunakan fitur *Input Disease*, *user* akan diminta untuk memasukkan nama penyakit serta *sequence* DNA dari penyakit bersangkutan dalam format *file* *txt*. Tekan tombol submit untuk memasukkan informasi yang telah diinput.
 - Untuk menggunakan fitur *DNA Test*, *user* akan diminta untuk memasukkan nama *user* atau pasien, nama penyakit yang ingin diprediksi, serta *sequence* DNA dari *user* atau pasien tersebut. Tekan tombol submit untuk memasukkan informasi yang telah diinput.
 - Untuk menggunakan fitur *Searching*, *user* akan diminta untuk memasukkan tanggal dan/atau nama penyakit yang ingin dicari pada log tes DNA. Tekan tombol submit untuk melihat informasi log tes DNA berdasarkan inputan dari *user*.

4.3 Hasil pengujian

Tampilan Awal website



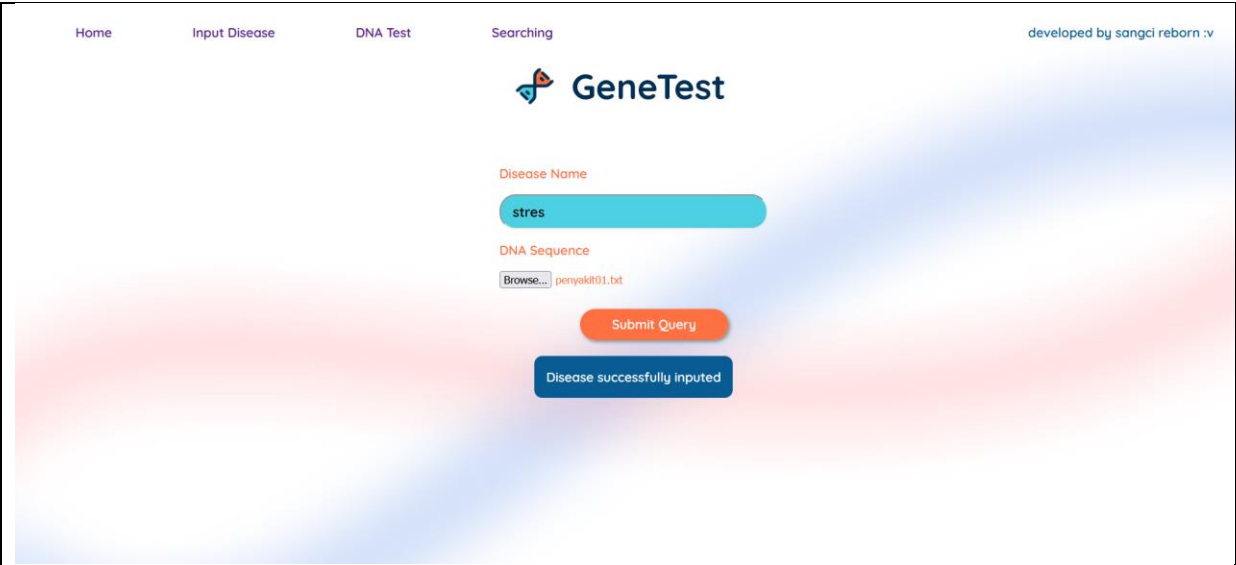
UJI Input Disease

Test case benar

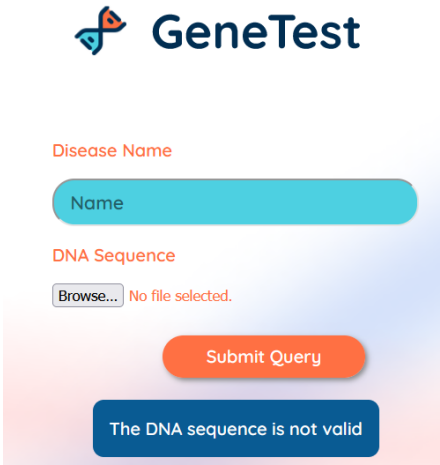
penyakit01.txt - Notepad

File Edit View

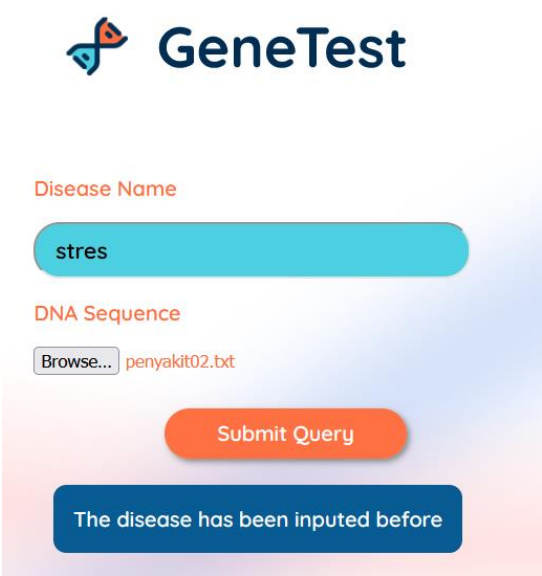
TACGCTCGATCGAAA



Test case salah – form tidak diisi



Test case salah –nama penyakit telah ada pada *database*




UJI DNA Test

Test case benar – uji pada sequence DNA dengan panjang 100

sequence.txt - Notepad

FileEditView

GTGTAAATAAGGTGGGTTCCCACTGCAGAGTCTGCTAAATTGGCTAAACCCCGTGCCCCCTGACTGGCCCTAGCTAAGTTAACATTAAATCTGAGTACGGAGGGCGGTGCCTTTTA

GeneTest

Name

PasienA

Disease Prediction

hiv

DNA Sequence


Browse...

sequence.txt

Submit

29 April 2022 - PasienA - hiv - 100.00% - true

Test case salah – nama pasien tidak diisi

GeneTest

Name

Name

Disease Prediction

hiv

DNA Sequence

Browse...

sequence.txt

Submit

29 April 2022 - - hiv - 100.00% - true

Test case salah – nama penyakit tidak diisi



Name

PasienB

Disease Prediction

Disease Prediction

DNA Sequence

Browse... sequence.txt

Submit

The disease has not been inputed before

Test case salah – file sequence DNA tidak diupload



Name

PasienC

Disease Prediction

hiv

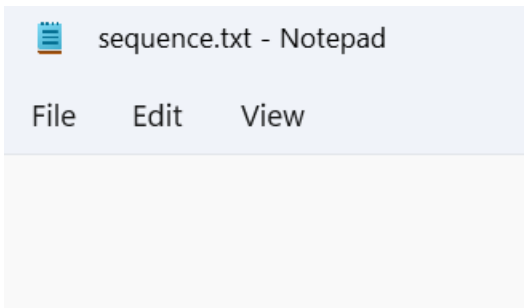
DNA Sequence

Browse... No file selected.

Submit

The DNA sequence is not valid

Test case salah – file sequence DNA tidak berisi





Name

PasienD

Disease Prediction

hiv

DNA Sequence

Browse... sequence.txt

Submit

The DNA sequence is not valid

UJI Searching

Test case benar



28 April 2022

Submit

2022-04-28 - Vieri Mansyl - hiv - 90.48% - true



29 April 2022

Submit

2022-04-29 - Vieri - meriang - 93.33% - true

2022-04-29 - PasienA - hiv - 100.00% - true

2022-04-29 - - hiv - 100.00% - true

2022-04-29 - PasienE - Meriang - 100.00% - true

 GeneTest

29 April 2022 HIV

Submit

2022-04-29 - PasienA - hiv - 100.00% - true

2022-04-29 - - hiv - 100.00% - true

 GeneTest

28 Apr 2022

Submit

2022-04-28 - Vieri Mansyl - hiv - 90.48% - true

Test case salah – tidak ada masukan

 GeneTest

<date><space><disease-name>

Submit

No data match the search

Test case salah – tanggal masukan tidak valid

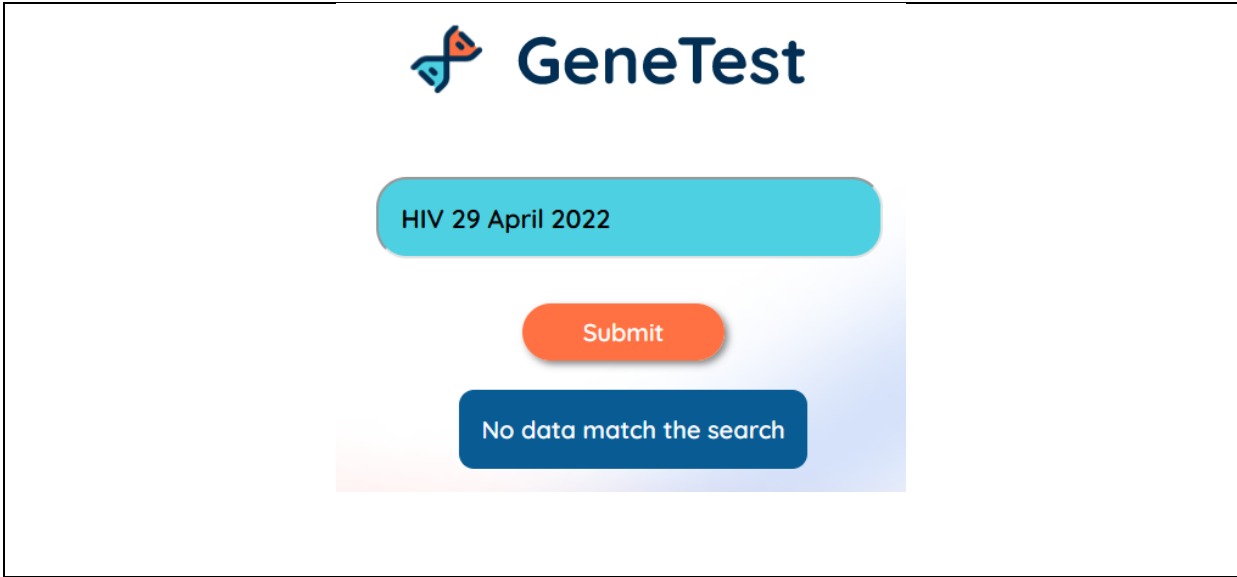
 GeneTest

28 April 2022

Submit

No data match the search

Test case salah – urutan pencarian terbalik



4.4 Analisis hasil pengujian

Berdasarkan hasil pengujian pada subbab 4.3, dapat disimpulkan bahwa aplikasi berbasis web GeneTest yang telah dikembangkan dapat berjalan dengan baik. Aplikasi GeneTest dapat menerima input data penyakit, kemudian mengembalikan pesan berhasil atau pesan kesalahan yang bersesuaian dengan kesalahan. Kemudian, aplikasi juga dapat menerima input pengetesan DNA dari pasien dan prediksi nama penyakit; aplikasi juga berhasil mengembalikan hasil pemrosesan dan pesan kesalahan jika terdapat kesalahan pada input. Terakhir, aplikasi dapat melakukan searching berdasarkan query yang dimasukkan; aplikasi berhasil mengembalikan daftar hasil dan juga pesan kesalahan jika terdapat kesalahan pada query.

BAB V

KESIMPULAN, SARAN, DAN KOMENTAR

5.1 Kesimpulan

Aplikasi GeneTest yang telah dibangun berhasil mengimplementasikan algoritma *string matching* dan *Regex* untuk menjalankan seluruh fitur yang disediakan untuk melayani *user* dalam melakukan perbandingan antara *sequence* DNA *user* terhadap *sequence* DNA dari suatu penyakit serta validasi terhadap masukan inputan menggunakan *Regex*. Program juga berhasil melakukan pencarian *string* dengan menggunakan algoritma KMP beserta kalkulasi kemiripan antara *sequence* DNA *user* terhadap *sequence* DNA penyakit berdasarkan hasil perhitungan *Hamming distance*. *Frontend* dan *Backend* serta *Database* berhasil dihubungkan dengan menggunakan *framework* *Gin* dan *library* *React* sehingga membentuk aplikasi website yang telah dibangun.

5.2 Saran dan Komentar

A. Saran

Hasil dari website yang telah dibangun masih mengandung banyak kekurangan, dimulai dari tidak elegannya kode program yang telah dibuat sampai dengan UI yang masih bisa lebih responsif terhadap bervariasinya jenis masukan yang dapat diberikan oleh *user*.

B. Komentor

Dari Tugas Besar ini, penulis beserta koleganya dipacu untuk berkembang melalui tugas yang menarik. Tidak hanya mendorong penulis dan koleganya untuk mengaplikasikan materi pembelajaran yang diberikan selama perkuliahan, penulis dan koleganya juga ditantang untuk bekerja sama dalam satu tim serta *me-manage* waktu untuk menyesuaikan waktu tenggat dari tugas-tugas yang telah diberikan.

Melalui Tugas Besar ini, penulis beserta koleganya mengucapkan terima kasih kepada pengajar mata kuliah IF2211 Strategi Algoritma yang telah menuang ilmu kepada penulis dan koleganya serta mahasiswa lainnya, juga atas tugas-tugas yang seru dan banyak ini. Tidak hanya itu, penulis ingin mengucapkan terima kasih kepada para asisten yang telah membantu membimbing penulis beserta angkatannya atas arahan selama 1 tahun ini.

LAMPIRAN

Tautan yang terkait :

- Github repository

https://github.com/weslygio/Tubes3_13520071

DAFTAR PUSTAKA

- Tugas Besar 3 (Tubes 3): Penerapan String Matching dan Regular Expression dalam DNA Pattern Matching.

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Tugas-Besar-3-IF2211-Strategi-Algoritma-2022.pdf>

- Pencocokan string (String matching/pattern matching)

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

- Pencocokan string dengan Regular Expression (Regex)

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

- Gambar sekuens DNA

<https://towardsdatascience.com/pairwise-sequence-alignment-using-biopython-d1a9d0ba861f>

- Golang Documentation

<https://go.dev/doc/>

- StackOverflow

<https://stackoverflow.com/search?q=react>

<https://stackoverflow.com/search?q=golang>