

Laporan Tugas Besar IF2211 Strategi Algoritma
Penerapan String Matching dan Regular Expression dalam
DNA Pattern Matching



Dibuat oleh:

13520006 - Vionie Novencia Thanggesyo

13520012 - Aji Andhika Falah

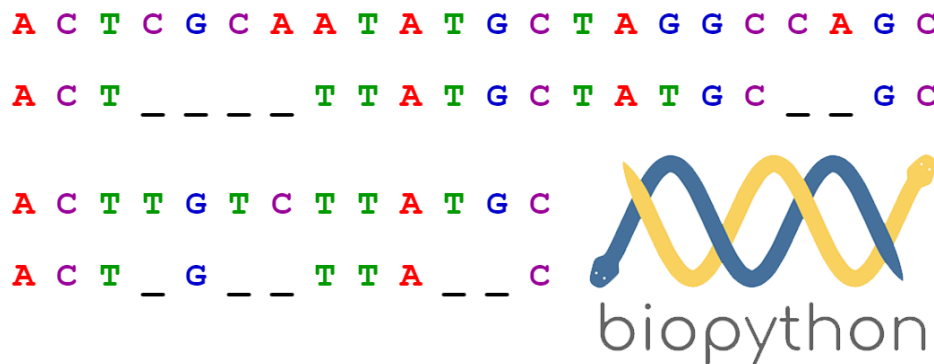
13520122 - Alifia Rahmah

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

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

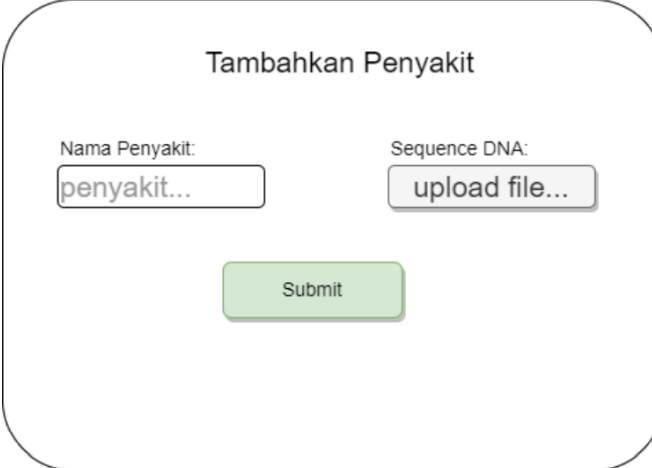
(Sumber: <https://towardsdatascience.com/pairwise-sequence-alignment-using-biopython-d1a9d0ba861f>)

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.

Fitur-Fitur Aplikasi:

1. Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya (dan dimasukkan ke dalam database).
 - a. Implementasi input sequence DNA dalam bentuk file.
 - b. 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).
 - c. Contoh input penyakit:



The image shows a web form titled "Tambahkan Penyakit". It has 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.

Gambar 2. Ilustrasi Input Penyakit

2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.
 - a. Tes DNA dilakukan dengan menerima input nama pengguna, sequence DNA pengguna, dan nama penyakit yang diuji. Asumsi sequence DNA pengguna > sequence DNA penyakit.
 - b. 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).
 - c. Pencocokan sequence DNA dilakukan dengan menggunakan algoritma string matching.
 - d. 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
 - e. Semua komponen hasil tes ini dapat ditampilkan pada halaman web (refer ke poin 3 pada "Fitur-Fitur Aplikasi") dan disimpan pada sebuah tabel database.
 - f. Contoh tampilan web:

Tes DNA

Nama Pengguna:

Sequence DNA:

Prediksi Penyakit:

Hasil Tes

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

3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.
 - a. Kolom pencarian dapat menerima masukan dengan struktur: , contoh “13 April 2022 HIV”. Format penanggalan dibebaskan, jika bisa menerima >1 format lebih baik.
 - b. Kolom pencarian dapat menerima masukan hanya tanggal ataupun hanya nama penyakit. Fitur ini diimplementasikan menggunakan regex.
4. (Bonus) Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA
 - a. Ketika melakukan tes DNA, terdapat persentase kemiripan DNA dalam hasil tes. Contoh hasil tes: 1 April 2022 - Mhs IF - HIV - 75% - False
 - b. Perhitungan tingkat kemiripan dapat dilakukan dengan menggunakan Hamming distance, Levenshtein distance, LCS, atau algoritma lainnya (dapat dijelaskan dalam laporan).
 - c. 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.

BAB II

Landasan Teori

Algoritma KMP

Algoritma Knuth-Morris-Pratt adalah algoritma *string matching* yang dikembangkan oleh Donald E. Knuth pada 1967 dan James H. Morris bersama dengan Vaughan R. Pratt pada tahun 1966. Algoritma ini mirip seperti algoritma *brute force*, mengecek dari kiri ke kanan, akan tetapi dilakukan pergeseran yang lebih hemat sehingga bisa menghindari perbandingan yang sia-sia.

Sebelum pengecekan, algoritma ini memroses pola yang akan dicari terlebih dahulu. Proses ini adalah fungsi pinggiran (*border function*). Fungsi ini mencari prefix terpanjang dari Pola[0..i] yang juga termasuk suffix terpanjang dari Pola[1..i]. Dengan menggunakan fungsi ini, jika ada karakter yang berbeda dengan pola ($P[j] \neq T[i]$), maka digunakan fungsi tersebut untuk “lompat”, menghindari pengecekan yang sia-sia. Langkah-langkah algoritma ini adalah:

1. Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
 - Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
 - Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini dan loop berhenti.
3. Algoritma kemudian menggeser pattern berdasarkan tabel hasil border function, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Algoritma BM

Algoritma Boyer-Moore adalah algoritma *string matching* yang dipublikasikan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1977. Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum.

Algoritma ini menggunakan 2 teknik, *the looking glass* dan *character jump*. Teknik *the looking glass* adalah membandingkan pola dan teks dari akhir pola ke awal pola, dicek secara mundur. Teknik *character jump* terjadi ketika $T[i] = x$ tapi $P[j]$ tidak sama dengan $T[i]$. Teknik ini ada 3 kasus. Kasus pertama x ada di P , maka P akan bergeser ke kanan sehingga x terakhir di P sejajar dengan $T[i]$. Kasus kedua adalah x ada di P , tapi P harus bergeser ke kiri. Karena tidak boleh geser ke kiri, maka geser P sebanyak 1 karakter ke $T[i+1]$. Kasus 3 terjadi ketika tidak ada x di P , maka geser $P[0]$ ke $T[i+1]$. Langkah-langkah dari algoritma ini adalah sebagai berikut:

1. Algoritma Boyer-Moore mulai mencocokkan pattern pada awal teks.
2. Dari kanan ke kiri (*the looking glass*), algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
 - Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
 - Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini dan memberhentikan loop.
3. Jika mismatch, gunakan teknik *character jump*, lalu ulangi langkah ke-2

Regular Expression (Regex)

Regex, singkatan dari Regular Expression, adalah sebuah string yang mendefinisikan sebuah pola pencarian sehingga dapat membantu kita untuk melakukan *matching* (pencocokan), *locate* (pencarian), dan manipulasi teks. Ekspresi reguler merupakan teknik yang dikembangkan dalam bidang ilmu komputer teori dan teori bahasa formal. Konsep ini muncul pada 1950-an ketika matematikawan Amerika Stephen Cole Kleene memformalkan deskripsi sebuah bahasa reguler. Konsep ini menjadi banyak digunakan untuk utilitas pengolahan teks Unix.

Web Development Tech Stack

Front-end

Front-end adalah tampilan dari aplikasi web yang berinteraksi langsung dengan pengguna. Pada program ini, Front -end dibuat dengan menggunakan framework React.Js dan menggunakan bahasa javascript. React adalah library Javascript terpopuler untuk membuat user interface (UI). Tool ini menawarkan respons cepat untuk user input dengan menggunakan metode baru dalam proses rendering website. Komponen dari tool ini dikembangkan oleh Facebook (sekarang Meta). Dahulunya tool ini diluncurkan sebagai tool open-source JavaScript di tahun 2013. Saat ini, React tetap terdepan mendahului kompetitornya seperti Angular dan Bootstrap, dua library JavaScript terlaris.

Back-end

Back-end adalah bagian dari aplikasi web yang berfungsi untuk menyediakan kebutuhan aplikasi web yang tak terlihat dari pengguna. Kebutuhan yang disediakan di antaranya untuk transaksi data. Back-end dari program ini dibuat dengan menggunakan bahasa Golang dan framework Gin. Golang adalah bahasa pemrograman *open-source* yang dibuat oleh Google pada tahun 2007. Golang banyak digunakan untuk berbagai kebutuhan, seperti untuk service Cloud, DevOps, dan Web Development. Gin adalah framework web untuk Golang yang dapat digunakan untuk routing, middleware, dan validasi JSON dari sebuah HTTP request.

Database

Database management system (DBMS) adalah sebuah software pengelolaan database (basis data) agar proses manajemen dan pengorganisasian data dapat berjalan dengan baik. Aplikasi web ini menggunakan DBMS Relasional MySQL. Dalam database relasional, data disimpan dalam sebuah relasi (tabel) yang terdiri dari atribut (kolom) dan record (baris). Relasi dalam sebuah database dapat berhubungan satu sama lain.

BAB III

Analisis Pemecahan Masalah

Langkah Penyelesaian Masalah Tiap Fitur

Algoritma

Algoritma KMP, dibagi menjadi 3 fungsi, mainKMP, KMP, dan computeBorder. Untuk menggunakan algoritma ini, masukkan string dna user dan dna penyakit ke mainKMP dan akan diolah di KMP. Karena KMP butuh border function, maka dibuat fungsi computeBorder. Fungsi ini menghasilkan sebuah array of integer yang berisi hasil *border function*. KMP akan memberi hasilnya dalam bentuk boolean ke mainKMP, lalu mainKMP akan memberikan hasil tersebut, true jika ditemukan, false jika tidak.

Algoritma Boyer-Moore juga dibagi menjadi 3 bagian, mainBMS, BMS, dan LastOcc. Untuk menggunakan algoritma ini, masukkan string dna user dan dna penyakit ke mainBMS dan akan diolah di BMS. Karena KMP butuh last occurrence, maka dibuat fungsi lastOcc. Fungsi ini menghasilkan sebuah array of integer dari 0 sampai 90 yang merepresentasikan ASCII dan berisi index terakhir karakter tersebut muncul. BMS akan memberi hasilnya ke mainBMS, lalu mainBMS akan memberikan hasil tersebut, true jika ditemukan, false jika tidak.

Front-end

Untuk fitur Search, data masukan pengguna dikirim ke back-end dengan menggunakan axios dan dipassing dengan parameter q. Masukan user kemudian diproses oleh back-end. Hasil dari back-end dikembalikan ke front-end untuk ditampilkan. Pada Fitur Tambah Penyakit, front-end menerima masukan data nama penyakit dan file dna sequence. Keduanya dikirimkan ke back-end dengan menggunakan POST request. Sedangkan pada fitur Tes DNA, front-end melakukan GET request ke backend untuk mendapatkan list penyakit, setelah itu, frontend menerima masukan penyakit yang dipilih pengguna, dna sequence pengguna, dan nama pengguna. Ketika pengguna menekan tombol submit, data tersebut dikirimkan ke back-end untuk di proses dengan menggunakan POST request. Hasil dari proses tersebut dikembalikan ke front-end untuk ditampilkan.

Back-end

Database untuk aplikasi web ini terdiri dari dua relasi, disease dan prediction. Relasi disease terdiri dari atribut disease_id, disease_name, dan dna_sequence. Relasi prediction terdiri dari prediction_id, prediction_date, patient_name, dna_sequence, disease_id berupa foreign key menuju disease_id di relasi disease, dan result.

Pada API dari aplikasi, terdapat endpoint untuk relasi disease dan prediction, yang dapat digunakan untuk melakukan request POST dan GET. POST request untuk mengirim input data lalu memprosesnya dengan dilakukan sanitasi input dan melakukan string matching sebelum ditambahkan ke dalam database, dan GET untuk mendapatkan data dalam bentuk JSON. Pada saat melakukan GET request untuk mendapat data prediction, dapat diberikan parameter tambahan query search untuk menampilkan data yang sesuai dengan tanggal atau nama penyakit yang diberikan dalam input search dari front-end, yang akan diproses dengan regular expression untuk mengambil tanggal dan nama penyakit.

Fitur Fungsional dan Arsitektur Aplikasi

Fitur fungsional dari website ini adalah:

- TesDNA
- TambahPenyakit
- Search

Arsitektur aplikasi:

- Frontend
 - Page Search
 - Page TambahPenyakit
 - Page Tes DNA
 - Page Navbar
 - CSS Styling (Style.css, Navbar.css)
 - Frontend utility (formatDate.js)
- Backend
 - Utilitas database (file .env, initDB)
 - Routing
 - GetDisease
 - GetDiseaseById
 - GetPrediction
 - GetPredictionById
 - PostDisease
 - PostPrediction
 - Utilitas lainnya
 - readFile
 - parseDate
 - parseWord
 - Algoritma (file KMP.go berisi algoritma KMP, file BMS.go berisi algoritma Boyer-Moore)

BAB IV

Implementasi dan Pengujian

Spesifikasi Teknis Program

Program ini menggunakan bahasa Golang untuk back-end dan algoritma. API back-end dari aplikasi dibuat terpisah dari aplikasi front-end. Front-end dibuat dengan bahasa javascript dan framework React

Daftar endpoint API dan HTTP request yang bisa dilakukan:

- /api/disease (POST, GET)
- /api/disease/:id (GET, DELETE)
- /api/prediction (POST, GET)
- /api/prediction/:id (GET, DELETE)

Tata Cara Penggunaan Program

Untuk menjalankan front-end gunakan command *npm start* pada folder frontend

1. Search

Masukkan penyakit dan/atau tanggal yang ingin dicari pada kolom search.

Masukan pencarian dapat dalam format <Tanggal><Spasi><Penyakit>, <Tanggal> atau <Penyakit>. Format tanggal dapat berupa YYYY-MM-DD, DD-MM-YYYY, dan DD <Nama bulan> YYYY.

2. Tambah Penyakit

Masukkan nama penyakit kemudian upload file yang mengandung dna sequence penyakit. Lalu tekan submit. Pesan berhasil akan diberikan jika data berhasil ditambahkan ke database.

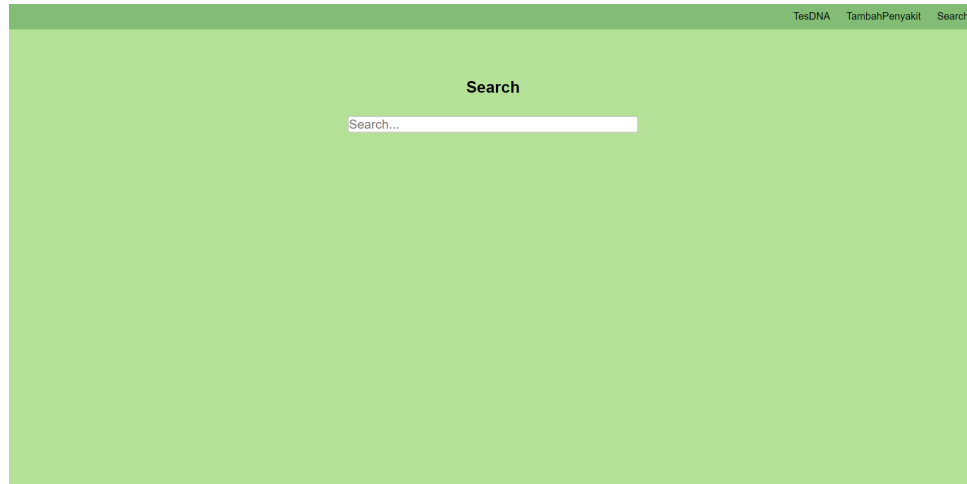
3. Tes DNA

Pilih penyakit yang ingin di uji, upload file dna pengguna, masukkan nama pengguna, kemudian tekan submit. Hasil tes akan ditampilkan di bagian bawah page.

Hasil Pengujian

1. Search

a. Sebelum pengujian



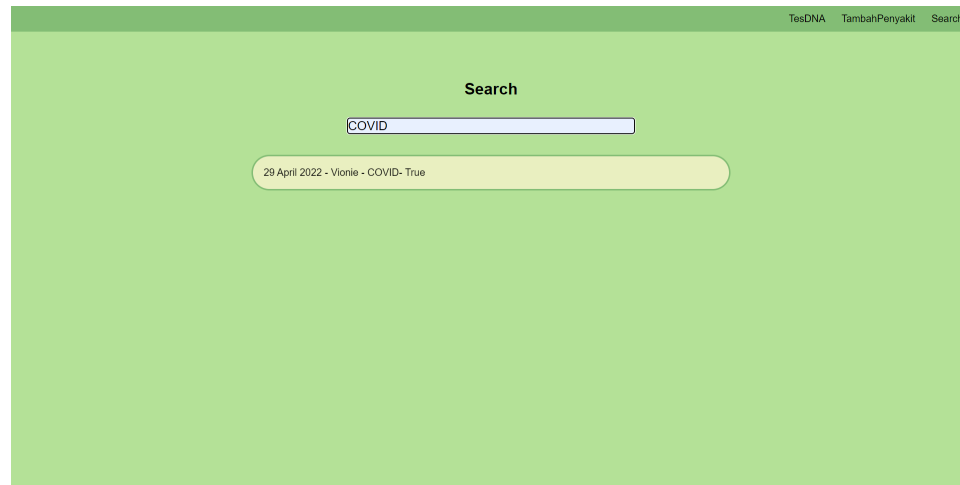
- b. Pengujian dengan format masukan <Tanggal><Spasi><Penyakit>



- c. Pengujian dengan format masukan <Tanggal>



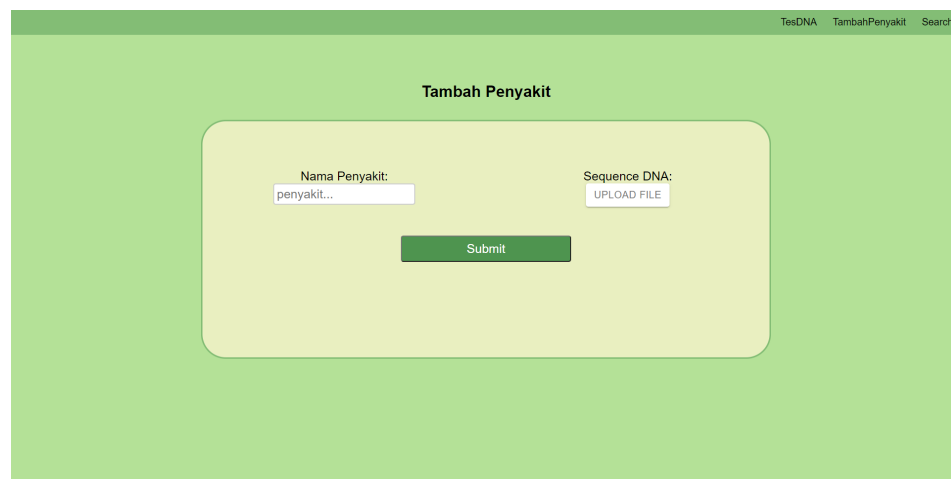
d. Pengujian dengan format <Penyakit>



The screenshot shows a web application interface with a green header bar containing the links "TesDNA", "TambahPenyakit", and "Search". The main content area has a green background. At the top center, the word "Search" is displayed. Below it is a text input field containing the word "COVID". Underneath the input field is a yellow rounded rectangle containing the text "29 April 2022 - Vionie - COVID- True".

2. Tambah Penyakit

a. Sebelum pengujian



The screenshot shows a web application interface with a green header bar containing the links "TesDNA", "TambahPenyakit", and "Search". The main content area has a green background. At the top center, the text "Tambah Penyakit" is displayed. Below it is a yellow rounded rectangle containing a form. The form has two input fields: "Nama Penyakit:" with the placeholder text "penyakit..." and "Sequence DNA:" with the placeholder text "UPLOAD FILE". Below these fields is a green "Submit" button.

b. Pengujian

TesDNA TambahPenyakit Search

Tambah Penyakit

Nama Penyakit:

Sequence DNA:

dna-checker.herokuapp.com says
Data penyakit berhasil ditambahkan

TesDNA TambahPenyakit Search

Tambah Penyakit

Nama Penyakit:

Sequence DNA:

3. Tes DNA
 - a. Sebelum pengujian

TesDNA TambahPenyakit Search

Tes DNA

Prediksi Penyakit:

Sequence DNA:

Nama Pengguna:

Hasil Test

b. Pengujian

The screenshot shows the 'Tes DNA' web application interface. At the top, there is a navigation bar with links: 'TesDNA', 'TambahPenyakit', and 'Search'. The main heading is 'Tes DNA'. Below it, there is a form with three input fields: 'Prediksi Penyakit:' with a dropdown menu showing 'SARS', 'Sequence DNA:' with a text input field containing 'TES2.TXT', and 'Nama Pengguna:' with a text input field containing 'Ziel'. A green 'Submit' button is centered below these fields. Under the button, the text 'Hasil Test' is displayed.

This screenshot shows the same 'Tes DNA' web application interface after the 'Submit' button has been clicked. The form fields remain the same. Below the 'Submit' button, the text 'Hasil Test' is followed by the result: '29 April 2022 - Ziel - SARS - False'.

Analisis Hasil Pengujian

Dari pengujian yang dilakukan, dapat disimpulkan aplikasi berjalan dengan baik dan memberikan hasil sesuai yang diharapkan.

Pada pengujian search, dengan format masukan yang benar, sudah bisa muncul tes-tes DNA yang pernah dilakukan. Pada pengujian tambah penyakit, penyakit yang dimasukkan sudah bisa tersimpan di database dan akan muncul pilihannya di tes DNA. Pada pengujian tes DNA, sudah bisa menunjukkan hasil yang benar jika ada DNA penyakit di bagian DNA input user.

Aplikasi juga berjalan dengan baik.

BAB V

Kesimpulan, Saran, dan Refleksi

Kesimpulan

Dalam menyelesaikan permasalahan *string matching*, bisa diselesaikan dengan algoritma KMP maupun Boyer-Moore. Dengan menggunakan KMP, algoritma akan seperti *brute force* dengan “lompatan” agar efisien. Jika menggunakan Boyer-Moore akan menggunakan teknik *the looking glass* dan *character jump*.

Saran

Struktur program dari Tugas Besar 3 ini kurang modular. Penulis menyarankan agar kedepannya lebih memperhatikan modularitas dalam pembuatan program.

Refleksi

Setelah menyelesaikan Tugas Besar 3 ini, pengetahuan kami mengenai algoritma string matching, regex, dan web development bertambah. Kami mempelajari bahasa baru Golang dan menggunakannya untuk membuat API sebagai backend. Kami juga mempelajari framework React untuk membuat front-end dari website, dihias dengan styling dari CSS. Kami juga mengenal HTTP request dan menggunakannya untuk melakukan pertukaran data antar front-end dan back-end.

Daftar Pustaka

Slide kuliah IF2211 Strategi Algoritma

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm>

Documentation - The Go Programming Language, <https://go.dev/doc/>

Lampiran

Link repository:

https://github.com/Enderageous/Tubes3_Dimulai-dari-ATG

Link deployment:

<https://dna-checker.herokuapp.com/> (Website)

<https://enigmatic-brook-59106.herokuapp.com/> (API)

Link Video:

<https://youtu.be/Ahv3QGWSqGs>