

Laporan Tugas Besar 3 IF2211 Strategi Algoritma  
Semester II tahun 2021/2022

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



Dipersiapkan oleh:

**Kelompok 21: AlgeoComeback**

13520077	Rava Naufal Attar
13520082	Jeremy Rionaldo Pasaribu
13520104	Yakobus Iryanto Prasethio

*Sekolah Teknik Elektro dan Informatika*

*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*

## **Daftar Isi**

Daftar Isi.....	1
BAB 1: Deskripsi Masalah.....	2
BAB 2: Landasan Teori .....	3
BAB 3: Analisis Pemecahan Masalah .....	5
BAB 4: Implementasi Dan Pengujian.....	7
BAB 5: Kesimpulan, Saran, dan Refleksi.....	15
Daftar Pustaka.....	16
Lampiran Link.....	17

## BAB 1: Deskripsi Masalah

### 1.1. 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.

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.

## BAB 2: Landasan Teori

### 2.1. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah suatu algoritma yang digunakan untuk mencocokkan string. Algoritma ini mencocokkan string dari arah kiri ke kanan, sehingga mirip dengan algoritma brute force. Perbedaan antara algoritma KMP dengan algoritma brute force adalah algoritma KMP menggeser pattern yang dicarinya secara lebih pintar dibandingkan dengan algoritma brute force. Ketika ada mismatch yang terjadi diantara string text  $T[i]$  dan string pattern  $P[j]$ , maka algoritma akan menggeser pattern seukuran prefix terbesar dari  $P[0..j-1]$  yang merupakan suffix dari  $P[1..j-1]$ .

Algoritma KMP membutuhkan sebuah border function untuk menemukan kesamaan antara prefix dari pattern dengan pattern itu sendiri. Misalkan  $j$  adalah posisi mismatch di pattern dan  $k$  adalah posisi sebelum mismatch terjadi ( $k = j-1$ ). Maka border function  $b(k)$  didefinisikan sebagai ukuran terbesar prefix  $P[0..k]$  yang juga merupakan suffix  $P[1..k]$ . Sebutan lain dari border function ini adalah failure function.

### 2.2. Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan algoritma pencocokan string yang memiliki dua buah teknik, yaitu teknik *looking-glass* dan teknik *character-jump*. Teknik *Looking-glass* merupakan teknik untuk mencari pattern  $P$  dalam teks  $T$  dengan melakukan iterasi mundur karakter  $P$  mulai dari karakter bagian akhir  $P$  sampai ke awal. Teknik *character-jump* merupakan teknik pemindahan pattern ketika terjadi *mismatch* pada suatu teks ( $P[j] \neq T[i]$ ). Teknik *character-jump* menggunakan *last occurrence function* yang menghitung indeks terbesar dalam *pattern*  $P$  dari semua karakter yang ada.

- Kasus 1: jika  $P$  mengandung  $x$ , geser pola  $p$  sampai *last occurrence* dari  $x$  dalam  $P$ .
- Kasus 2: jika  $P$  mengandung  $x$  tetapi pergeseran pola  $p$  sampai *last occurrence* tidak memungkinkan, geser pola  $p$  1 karakter ke  $T[i+1]$ .
- Kasus 3: jika bukan merupakan kasus 1 dan 2, pattern  $P$  mulai dari karakter awal ( $P[0]$ ) untuk sejajar dengan  $T[i+1]$ .

### 2.3. Regex

Dalam string matching ada yang dikenal dengan nama regular expression atau regex. Regex adalah string yang membentuk sebuah pattern yang dapat digunakan untuk mencocokkan dan mengatur sebuah text string. Regex memiliki aturan penulisan yang sudah didefinisikan dan harus diikuti. Sebagai contoh, karakter “^” menandakan awalan dari sebuah string, karakter “\$” menandakan akhiran dari sebuah string, karakter “\*” menandakan jumlah kemunculan sebuah karakter boleh nol atau banyak, dan masih banyak lagi.

Dengan menggunakan regex, tentunya meningkatkan fleksibilitas pembentukan pattern untuk sebuah text string. Untuk mencari pattern tertentu dalam sebuah teks,

hanya diperlukan sebuah regex, yang tentunya jauh lebih efisien dibandingkan menggunakan string matching yang mencoba pattern untuk setiap teks yang berbeda

## BAB 3: Analisis Pemecahan Masalah

### 3.1. Langkah Penyelesaian Masalah Setiap Fitur

- a. Menambahkan penyakit baru
  - Membaca nama penyakit baru yang diberikan oleh user
  - Membaca string rantai DNA penyakit dari sebuah text file yang diberikan oleh user
  - Menyimpan nama penyakit dan string rantai DNA ke tabel *jenis\_penyakit* di dalam database
- b. Melakukan testing prediksi penyakit
  - Membaca nama pasien yang diberikan oleh user
  - Membaca nama prediksi penyakit yang diberikan oleh user
  - Membaca string rantai DNA pasien
  - Membaca algoritma yang dipilih oleh user
  - Melakukan algoritma string matching sesuai dengan algoritma yang dipilih (antara KMP atau Boyer-Moore). Algoritma akan mengembalikan nilai *true* apabila terdapat match antara *pattern* dengan *text* yang diberikan.
  - Melakukan algoritma distance matching menggunakan algoritma Damerau-Levenshtein untuk menemukan keakuratan *pattern* terhadap *teks*.
  - Menyimpan hasil prediksi yang sudah dikalkulasi beserta nilai akurasi ke tabel *hasil\_prediksi* di dalam database
- c. Melakukan pencarian testing yang pernah dilakukan
  - Melakukan sanitasi *regex* terhadap masukan pengguna
  - Mencari hasil sanitasi *regex* pada *database*
  - Memunculkan semua data dari *database* sesuai masukan pengguna

### 3.2. Fitur Fungsional dan Arsitektur Web

- a. Fitur fungsional dari website adalah sebagai berikut:
  - Menambahkan penyakit baru ke dalam database
  - Melakukan testing prediksi penyakit
  - Melakukan pencarian testing yang pernah dilakukan
- b. Arsitektur website dibangun dari 2 bagian:
  - Frontend  
Frontend dari website yang kami bangun berupa *single page application* dengan *create-react-app* menggunakan framework React JS dan Tailwind CSS. *Dependencies* tambahan yang diperlukan antara lain *axios*, *react-router-dom*, *history*, dan *classnames*. Pada bagian frontend tersegmentasi menjadi beberapa folder yaitu, folder *assets* berisi gambar-gambar yang diperlukan pada website, folder *components* berisi beberapa komponen dasar untuk website, folder *pages* berisi halaman-halaman pada website, dan

folder *utils* berisi beberapa keperluan untuk validasi *string* menggunakan regex.

- Backend

Backend dari website kami dibangun dengan bahasa pemrograman Golang dan framework Gin. Database yang kami gunakan adalah MongoDB, dengan menggunakan cloud yang disediakan oleh MongoDB sendiri, yaitu Atlas. Pada folder backend, terdapat folder router yang menyambungkan sebuah route dengan fungsi di backend, folder models yang memiliki tipe struktur data yang diperlukan, folder libs yang menyimpan algoritma string matching dan distance matching, folder database yang menyimpan koneksi antara fungsi backend dengan database, dan folder controller yang menyimpan fungsi utama seperti test, search, dan add.

## BAB 4: Implementasi Dan Pengujian

### 4.1. Spesifikasi Teknis Program

#### a. Struktur Data

Struktur Data yang digunakan adalah *Javascript Object Notation* (JSON). Struktur data ini digunakan dalam proses POST dan GET. Contoh data JSON:

```
{
  "nama_pasien" : "John Doe",
  "penyakit_prediksi" : "Nubes",
  "method" : "KMP",
  "text" : "ACTCATCGTCGTACGTAGATCGATCGATCAC"
}
{
  "tanggal_prediksi" : "28 April 2022",
  "nama_pasien" : "John Doe",
  "penyakit_prediksi" : "Nubes",
  "accuracy" : "87.0",
  "status_prediksi" : true
}
```

#### b. Fungsi dan Prosedur

Fugnsi/Prosedur	Penjelasan
BorderFunction(pattern: string): [] <u>int</u>	Menerima masukan pattern berupa string dan mengembalikan ukuran terbesar prefix dari pattern[0..k] dan suffix dari pattern[1..K]
KMPMatch(text: string, pattern: string): <u>boolean</u>	Melakukan pencocokan string pattern secara algoritma Knuth-Morris-Pratt (KMP) terhadap text dan mengembalikan <i>true</i> jika substring dari teks sesuai dengan pattern. Jika tidak sesuai akan mengembalikan <i>false</i> .
findMin(i, j): <u>integer</u>	Menerima masukan i dan j berupa integer dan mengembalikan nilai terkecil antara i dan j
findMax(i, j): <u>integer</u>	Menerima masukan i dan j berupa integer dan mengembalikan nilai terbesar antara i dan j
minimum(a, b, c int): int	Menerima masukan a, b, c berupa integer dan memngembalikan nilai terkecil antara a, b, dan c
findLastOccurence(pattern: string): [] <u>int</u>	Menerima masukan pattern berupa string dan mengembalikan indeks



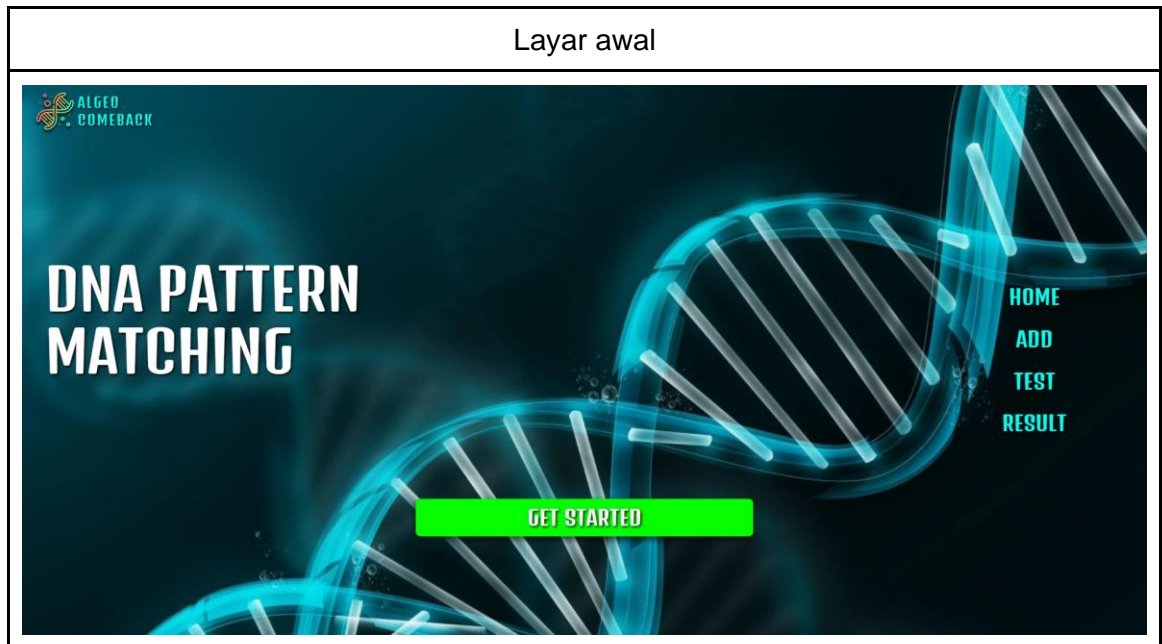
	terbesar dalam <i>pattern</i> dari semua karakter yang ada
<code>SearchBoyerMoore(text: string, pattern: string): bool</code>	Melakukan pencocokan string pattern secara algoritma Boyer-Moore (BM) terhadap text dan mengembalikan <i>true</i> jika substring dari teks sesuai dengan pattern. Jika tidak sesuai akan mengembalikan <i>false</i> .
<code>dnaValidation(sequenceDna): <u>boolean</u></code>	Mengembalikan <i>true</i> jika <i>pattern sequence</i> DNA merupakan DNA yang valid, <i>false</i> jika tidak
<code>searchValidation1(searchValue, data): <u>boolean</u></code>	Mengembalikan <i>true</i> jika pattern search value berupa 'DD Nama_Bulan YYYY Nama_Penyakit', <i>false</i> jika tidak
<code>searchValidation2(searchValue, data): <u>boolean</u></code>	Mengembalikan <i>true</i> jika <i>pattern search value</i> berupa 'DD Nama_Bulan YYYY', <i>false</i> jika tidak
<code>searchValidation3(searchValue, data): <u>boolean</u></code>	Mengembalikan <i>true</i> jika <i>pattern search value</i> berupa 'Nama_Penyakit', <i>false</i> jika tidak

#### 4.2. Tata Cara Penggunaan Program

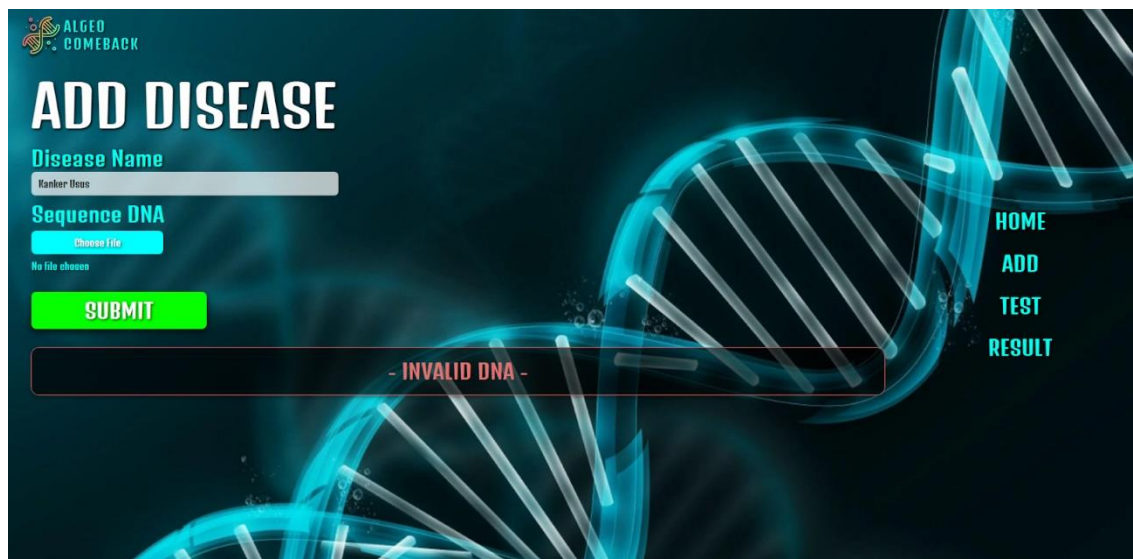
1. Buka command prompt dan masuk ke directory src/backend dari folder github
2. Ketik "go run main.go" untuk menjalankan bagian backend dari website
3. Buka command prompt yang baru dan masuk ke directory src/frontend dari folder github
4. Ketik "npm install" untuk mengunduh dependencies yang dibutuhkan oleh bagian frontend
5. Ketik "npm start" untuk menjalankan bagian frontend dari website
6. Ketika website terbuka, user akan berada di halaman home
7. Untuk menambahkan penyakit baru, pindah ke halaman Add, lalu masukkan nama penyakit beserta rantai DNA penyakit tersebut dalam bentuk text file, kemudian tekan tombol Submit
8. Untuk melakukan testing penyakit tertentu, pindah ke halaman Test, lalu masukkan nama pasien, nama prediksi penyakit, beserta rantai DNA pasien dalam bentuk file, kemudian pilih algoritma string matching yang ingin digunakan (antara KMP atau Boyer-Moore) dan tekan tombol Submit
9. Untuk melihat hasil prediksi apa saja yang pernah dilakukan, pindah ke halaman result. Apabila ingin mencari hasil prediksi secara spesifik (menurut tanggal atau nama penyakit), maka ketik tanggal atau nama penyakit di search field. Hasil prediksi akan muncul sesuai dengan input search field.

#### 4.3. Hasil Pengujian

Hasil pengujian berupa screenshot antarmuka dan skenario yang mungkin terjadi ketika website sedang digunakan



Add Penyakit ke Database Gagal (Invalid DNA)



The screenshot shows a web application interface for adding a disease. The title is "ADD DISEASE". There are two input fields: "Disease Name" with the value "Kanker Otak" and "Sequence DNA" with a "Choose File" button below it. A green "SUBMIT" button is visible. A red error message box at the bottom center displays "- INVALID DNA -". On the right side, there is a vertical navigation menu with links: "HOME", "ADD", "TEST", and "RESULT". The background features a glowing blue DNA double helix.

Add Penyakit ke Database Gagal (Penyakit sudah ada di database)



The screenshot shows the same "ADD DISEASE" form. The "Disease Name" field now contains the value "HIV". The "SUBMIT" button is still green. A red error message box at the bottom center displays "- DISEASE ALREADY IN DATABASE -". The navigation menu and background DNA helix are identical to the previous screenshot.

### Tes Penyakit Sukses

The screenshot shows the 'TEST DNA' interface with a dark blue background and a glowing DNA double helix. The 'Patient Name' field contains 'Yakobus Iryanto' and the 'Disease Prediction' field contains 'Kanker Usus'. The 'Sequence DNA' section has a 'Choose File' button and a 'No file chosen' message. The 'Boyer Moore' algorithm is selected with a green toggle. A green 'SUBMIT' button is visible. A green box at the bottom displays the result: '29 April 2022 - Yakobus Iryanto - Kanker Usus - 7.60% - False'. A sidebar on the right contains links: HOME, ADD, TEST, and RESULT.

ALGED COMEBACK

# TEST DNA

Patient Name: Yakobus Iryanto

Disease Prediction: Kanker Usus

Sequence DNA

Choose File

No file chosen

Boyer Moore ☒ KMP

SUBMIT

29 April 2022 - Yakobus Iryanto - Kanker Usus - 7.60% - False

HOME  
ADD  
TEST  
RESULT

### Tes Penyakit Gagal (Invalid DNA)

This screenshot shows the same 'TEST DNA' interface as above, but with a red box at the bottom displaying the message '- INVALID DNA -'. All other elements, including the patient name, disease prediction, and sidebar, remain the same.

ALGED COMEBACK

# TEST DNA

Patient Name: Yakobus Iryanto

Disease Prediction: Kanker Usus

Sequence DNA

Choose File

No file chosen

Boyer Moore ☒ KMP

SUBMIT

- INVALID DNA -

HOME  
ADD  
TEST  
RESULT



### Tes Penyakit Gagal (Penyakit tidak ditemukan di database)

The screenshot shows the 'TEST DNA' interface. The 'Patient Name' field contains 'Yakobus Iryanto'. The 'Sequence DNA' field is empty, with a 'Choose File' button and the text 'No file chosen'. The 'Boyer Moore' toggle is turned on, and 'KMP' is selected. A green 'SUBMIT' button is visible. The 'Disease Prediction' field shows 'Necrosis'. A red box at the bottom of the form contains the text '- DISEASE NOT FOUND -'. On the right side, there is a vertical menu with links: HOME, ADD, TEST, and RESULT.

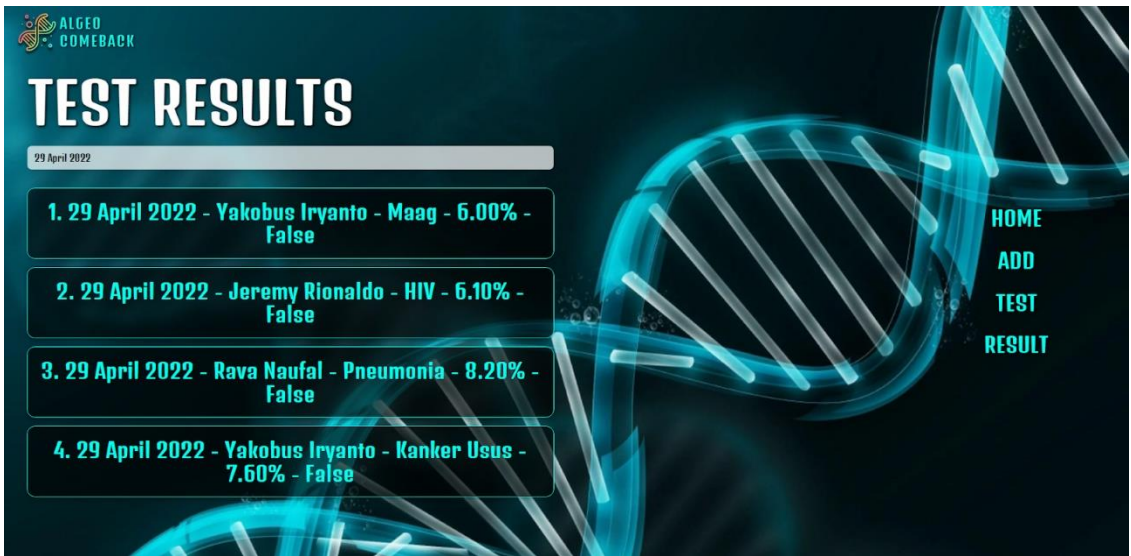
### Result (Layar awal)

The screenshot shows the 'TEST RESULTS' interface. It features a search bar at the top. Below it, there is a list of four test results, each in a separate box:

- 1. 29 April 2022 - Yakobus Iryanto - Maag - 6.00% - False
- 2. 29 April 2022 - Jeremy Ronaldo - HIV - 6.10% - False
- 3. 29 April 2022 - Rava Naufal - Pneumonia - 8.20% - False
- 4. 29 April 2022 - Yakobus Iryanto - Kanker Usus - 7.60% - False

On the right side, there is a vertical menu with links: HOME, ADD, TEST, and RESULT.


Result (Query tanggal)



The screenshot shows a web application interface with a dark blue background and a glowing DNA double helix. The header includes the logo 'ALGED COMEBACK' and the title 'TEST RESULTS'. Below the title is a search bar containing the date '29 April 2022'. The main content area displays a list of four test results, each in a rounded rectangle. On the right side, there is a vertical navigation menu with the options 'HOME', 'ADD', 'TEST', and 'RESULT'.

No.	Date	Name	Disease	Percentage	Result
1.	29 April 2022	Yakobus Iryanto	Maag	6.00%	False
2.	29 April 2022	Jeremy Ronaldo	HIV	6.10%	False
3.	29 April 2022	Rava Naufal	Pneumonia	8.20%	False
4.	29 April 2022	Yakobus Iryanto	Kanker Usus	7.60%	False

Result (Query nama penyakit)



The screenshot shows the same web application interface as the first one, but with the search bar containing the disease name 'Maag'. The list of test results now only contains one entry corresponding to 'Maag'. The navigation menu on the right remains the same.

No.	Date	Name	Disease	Percentage	Result
1.	29 April 2022	Yakobus Iryanto	Maag	6.00%	False



#### 4.4. Analisis Hasil Pengujian

Dari beberapa pengujian yang telah dilakukan, dapat dilihat bahwa website yang telah dibuat sudah bekerja sesuai dengan design dan keinginan dari kelompok kami. Website dapat menambahkan data penyakit beserta rantai DNA-nya ke dalam database. Website juga dapat melakukan testing terhadap penyakit yang mungkin dimiliki oleh seorang pasien. Website juga dapat menampilkan data hasil prediksi yang sudah pernah dites. Semua input dari user sudah dilakukan pembersihan dan sesuai dengan kebutuhan website. Apabila input dari user tidak sesuai dengan kebutuhan website, maka website dapat menampilkan pesan error.

Penggunaan algoritma string matching KMP dan Boyer-Moore digunakan untuk mendeteksi apakah string teks dan string pattern match atau tidak. Untuk permasalahan rantai DNA, algoritma KMP akan memproses permasalahan secara lebih cepat dibandingkan menggunakan algoritma Boyer-Moore. Hal ini dikarenakan untuk rantai DNA, tidak terdapat banyak variasi huruf yang mungkin ada (hanya A,C,G,T) dan kemungkinan menemukan prefix dan suffix yang lebih besar. Untuk kalkulasi distance matching, digunakan algoritma Damerau - Levenshtein. Nilai persentase yang dihasilkan adalah tingkat kemiripan antara string teks dengan string pattern.

## **BAB 5: Kesimpulan, Saran, dan Refleksi**

### **5.1. Kesimpulan**

Algoritma string matching (beserta regex) dan distance matching berhasil diimplementasikan dan digunakan dalam website pengujian penyakit. Regex berhasil digunakan untuk memvalidasi input dari user. Bagian front end dan back end berhasil digabungkan sehingga website pengujian penyakit yang memenuhi spesifikasi dapat terbentuk.

### **5.2. Saran**

Saran yang dapat kami berikan setelah mengerjakan tugas besar ini adalah:

- Melakukan eksplorasi library atau framework yang dapat digunakan untuk bagian frontend dan backend sebuah website
- Melakukan eksplorasi database services yang cocok digunakan untuk sebuah website
- Memahami spesifikasi tugas besar yang diberikan secara lebih mendalam
- Mulai mengerjakan tugas besar dari jauh hari agar tidak terganggu dengan tugas besar lainnya

### **5.3. Refleksi**

Pembuatan website ini membuat pemahaman mengenai algoritma string matching jauh lebih mudah. Bahasa Golang yang digunakan di bagian backend website cukup sulit untuk dimengerti, tetapi menjadi bekal pembelajaran yang baik untuk kami selanjutnya. Penggunaan framework React untuk bagian frontend dan framework Gin untuk bagian backend mempermudah proses penyatuan website. Database MongoDB juga memiliki API yang kaya, sehingga mempermudah akses keluar dan masuk database.



## **Daftar Pustaka**

“Pencocokan String (String/Pattern Matching)”. Rinaldi Munir.

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

“String matching dengan Regular Expression”. Rinaldi Munir.

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

## **Lampiran Link**

**Link Source Code (Github):**

[https://github.com/sivaren/Tubes3\\_13520077](https://github.com/sivaren/Tubes3_13520077)

**Link Video Program (Youtube):**

<https://bit.ly/AlgeoComebackVideoTubesStima>