

**LAPORAN TUGAS BESAR 3**  
**IF2211 STRATEGI ALGORITMA**

**PENERAPAN STRING MATCHING DAN REGULAR  
EXPRESSION DALAM DNA PATTERN MATCHING**



Dipersiapkan oleh:

**Kelompok DNAregeX (32)**

|                              |          |
|------------------------------|----------|
| Raka Wirabuana Ninagan       | 13520134 |
| Zayd Muhammad Kawakibi Zuhri | 13520144 |
| Mohamad Hilmi Rinaldi        | 13520149 |

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2022**

## DAFTAR ISI

|  |                 |
|--|-----------------|
| <b>DAFTAR ISI</b>  | <b><i>i</i></b> |
| <b>BAB I</b>   | <b>1</b>        |
| <b>BAB II</b>  | <b>4</b>        |
| Algoritma Knuth-Morris-Pratt (KMP)                         | 4               |
| Algoritma Boyer-Moore                                      | 5               |
| Regular Expression   | 7               |
| <b>BAB III</b>   | <b>8</b>        |
| Langkah-Langkah Pemecahan Masalah                          | 8               |
| Fitur Fungsional dan Arsitektur Aplikasi Web yang dibangun | 8               |
| <b>BAB IV</b>  | <b>10</b>       |
| Spesifikasi Teknis Program                                 | 10              |
| Struktur Data  | 10              |
| Fungsi dan Prosedur  | 10              |
| Tata Cara Penggunaan Program                               | 11              |
| Cara untuk menjalankan program                             | 11              |
| Cara untuk menggunakan program                             | 11              |
| Hasil Pengujian  | 12              |
| Analisis Hasil Pengujian                                   | 17              |
| <b>BAB V</b>   | <b>18</b>       |
| Kesimpulan   | 18              |
| Saran  | 18              |
| Refleksi   | 18              |
| <b>DAFTAR LINK</b>   | <b>19</b>       |
| <b>DAFTAR PUSTAKA</b>                                      | <b>20</b>       |

# BAB I

## DESKRIPSI TUGAS

Dalam tugas besar ini, kelompok 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, kelompok 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.

### Contoh Tampilan Program

Contoh masukan aplikasi:



The image shows a web form titled "Tambahkan Penyakit". It contains two input fields side-by-side. The first is labeled "Nama Penyakit:" and has a text input field with the placeholder "penyakit...". The second is labeled "Sequence DNA:" and has a file upload button labeled "upload file...". Below these two fields is a single green button labeled "Submit".

### Tes DNA

Nama Pengguna:  
<pengguna>

Sequence DNA:  
upload file...

Prediksi Penyakit:  
<penyakit>

Submit

---

Hasil Tes

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

Gambar halaman 1 masukan rantai DNA penyakit dan halaman 2 masukan inputan untuk tes DNA

Penyakit yang dimasukkan melalui halaman 1 akan disimpan ke database agar bisa dijadikan *pattern* yang akan dibandingkan ke DNA yang dimasukkan pengguna (halaman 2). Hasil tes akan menyatakan beberapa informasi seperti tanggal pengecekan, nama pengguna, penyakit yang diprediksi, dan keputusan apakah benar mengidap penyakit sesuai prediksi atau tidak.

Fitur riwayat pencarian:

13 April 2022 HIV

1. 13 April 2022 - Fulan - HIV - True.

2. 13 April 2022 - Kamal - HIV - False.

3. 13 April 2022 - Entah - HIV - False.

4. 13 April 2022 - Jamal - HIV - True.

5. 13 April 2022 - Yubai - HIV - True.

6. 13 April 2022 - Hika - HIV - False.

Pencarian akan menggunakan Regular Expression untuk mencari riwayat tes DNA yang pernah dilakukan di aplikasi ini.

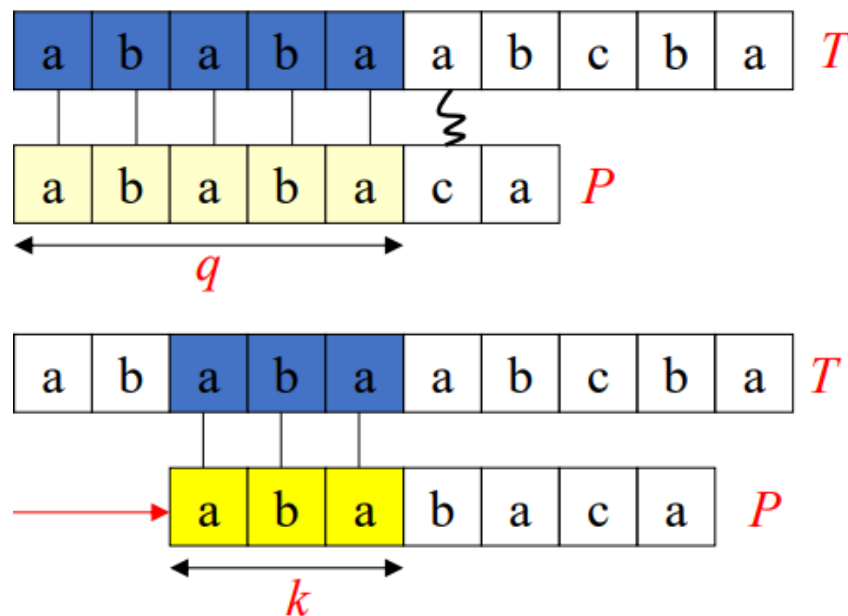
## BAB II

### LANDASAN TEORI

#### 2.1. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma KMP adalah algoritma untuk mencari *pattern* yang dicari pada sebuah teks dengan urutan pengecekan dari kiri ke kanan (*left-to-right order*). Sekilas, konsepnya mirip dengan algoritma pencocokan string *brute force*, karena kedua algoritma melakukan teknik pencocokan karakter yang sama. Perbedaannya, KMP jauh lebih cerdas dalam mengakali karakter mana saja yang tidak perlu diperiksa lagi (biasanya karena sudah pernah diperiksa di pengecekan sebelumnya), sehingga pergeseran yang dilakukan bisa lebih jauh (yang artinya jauh lebih sedikit melakukan pengecekan).

Misalkan, ada 2 buah string, yaitu *pattern* yang ingin dicari (ABABAABCBA) dan string yang diharapkan mengandung pattern (ABABACA). Langkah pencocokan diilustrasikan seperti ini:



Berdasarkan ilustrasi di atas, dapat dilihat bahwa ketika karakter di  $T$  dan  $P$  tidak sama, maka dilakukan pergeseran  $P$  ke kanan, tetapi ada 2 karakter yang dilewati. Hal ini dilakukan karena algoritma KMP menggeser pattern sejauh prefiks terbesar dari  $P[0..j-1]$  yang merupakan sufiks dari  $P[1..j-1]$  (pada kasus di atas,  $P[0..j-1] = ABABA$  dan  $P[1..j-1] = BABA$ , panjang

*pattern* yang sudah benar adalah 5, prefiks dan sufiks yang bersesuaian adalah ABA dengan panjang 3, maka  $5 - 3 = 2$  karakter pertama ditinggalkan tanpa pengecekan).

### Fungsi Pinggiran KMP (Border Function)

Algoritma KMP dapat mengetahui di mana pergeseran harus berhenti karena menggunakan informasi yang dihasilkan *border function* (disebut juga *failure function*, disingkat *fail*).

$(k = j-1)$

|        |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|
| $j$    | 0 | 1 | 2 | 3 | 4 | 5 |
| $P[j]$ | a | b | a | a | b | a |

|        |   |   |   |   |   |
|--------|---|---|---|---|---|
| $k$    | 0 | 1 | 2 | 3 | 4 |
| $b(k)$ | 0 | 0 | 1 | 1 | 2 |

$b(k)$  is the size of the largest border.

$j$  = indeks terjadinya ketidakcocokan karakter

$k = j - 1$

$b(k)$  = ukuran terbesar prefiks  $P[0..k]$  yang juga adalah sufiks  $P[1..k]$

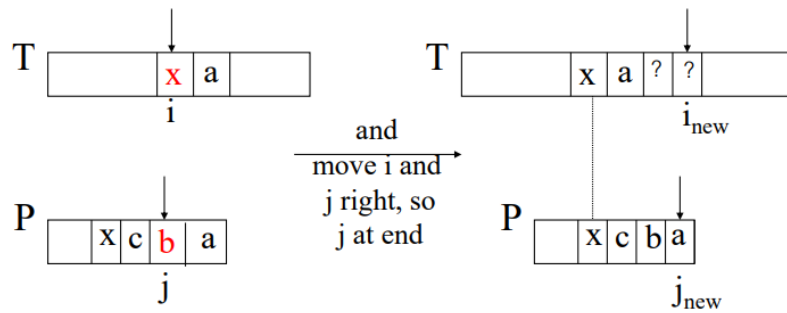
Nilai  $b(k)$  inilah yang dijadikan dasar algoritma KMP untuk mengetahui sejauh mana pergeseran harus dilakukan melalui pengurangan panjang *pattern* yang sudah valid dengan  $b(k)$ .

## 2.2. Algoritma Boyer-Moore

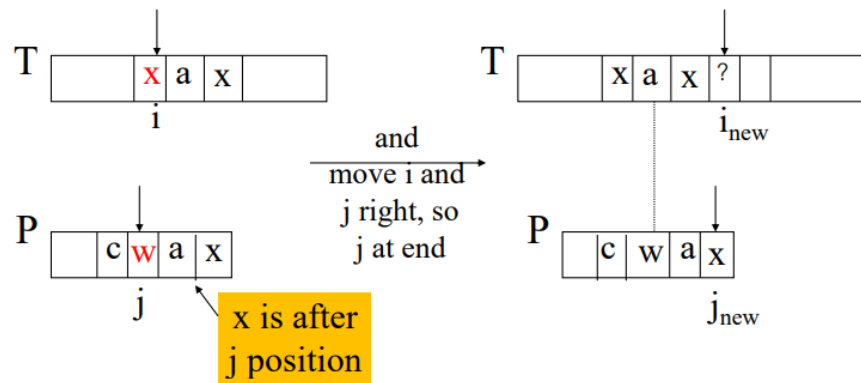
Algoritma Boyer-Moore melakukan pencocokan dengan 2 teknik, yaitu *looking glass technique* yang melakukan pencocokan dari bagian kanan *pattern* ke kiri (asumsikan huruf paling pertama tidak cocok, tentu akan dilakukan pergeseran *pattern* ke kanan, dan melakukan ulang *looking glass technique*), dan *character-jump technique*.

*Character-jump technique* dibagi menjadi 3:

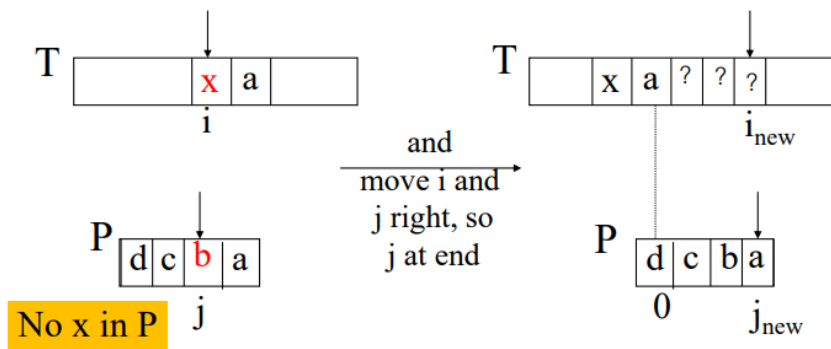
1. Kasus 1,  $P$  (*pattern*) mengandung  $x$ , maka dilakukan pergeseran sampai  $x$  pada  $P$  bertemu dengan  $x$  pada  $T$ .



2. Kasus 2,  $P$  (*pattern*) mengandung  $x$ , tetapi tidak mungkin melakukan shifting ke kiri (karena  $x$  tidak ada di kiri indeks  $j$ ), maka geser  $P$  ke kanan sebanyak 1 karakter.



3. Kasus 3, tidak memenuhi kasus 1 dan 2, maka langsung geser  $P[0]$  ke  $T[i+1]$ .



### Last Occurrence Function

Last Occurrence Function adalah fungsi yang memetakan setiap jenis karakter yang ada di *pattern* ke sebuah angka yang merupakan indeks paling akhir karakter tersebut muncul.



**P**

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| a | b | a | c | a | b |
| 0 | 1 | 2 | 3 | 4 | 5 |

|                    |                 |                 |                 |                 |
|--------------------|-----------------|-----------------|-----------------|-----------------|
| <b><i>x</i></b>    | <b><i>a</i></b> | <b><i>b</i></b> | <b><i>c</i></b> | <b><i>d</i></b> |
| <b><i>L(x)</i></b> | 4               | 5               | 3               | -1              |

Dapat dilihat bahwa a paling belakang berada di indeks 4, b pada indeks 5, c pada indeks 3 (berhubung hanya ada 1 c), dan d pada indeks -1 (tidak ada d pada *pattern*).

### 2.3. Regular Expression

Regular expression (regex) adalah notasi standar yang mendeskripsikan suatu pola (*pattern*) berupa urutan karakter atau string. Regex digunakan untuk pencocokan string dengan efisien karena memiliki *writing rule* tersendiri yang membantu pencarian sesuai keinginan.

Aturan penulisan pada Regex sudah sangat umum sehingga mayoritas bahasa pemrograman apapun akan mengadopsi cara ini. Misalkan dari sebuah teks, ingin dicari potongan string yang mengandung gabungan antara huruf a/b/c/d dengan e/f/g/h. Pada regex, permasalahan ini dapat diselesaikan dengan menulis `/[abcd][efgh]/g`.

## **BAB III**

### **ANALISIS PEMECAHAN MASALAH**

#### **3.1. Langkah-Langkah Pemecahan Masalah**

Dalam cara pandang sebuah teks, DNA terdiri dari 4 huruf kapital A, G, C, dan T. Karakter apapun selain keempat huruf ini tidak dapat dinyatakan sebagai DNA. Atas dasar ini, dibuat regex yang memeriksa apakah sebuah teks mengandung huruf selain keempat huruf DNA.

Program diharapkan mampu memeriksa apakah sebuah DNA yang dimasukkan pengguna mengidap penyakit tertentu. Langkah yang dilakukan untuk menyelesaikan poin persoalan ini dengan cara membuat basis data yang menyimpan data-data penyakit beserta rantai DNA penyakit tersebut, sehingga DNA yang dimasukkan pengguna akan dicocokkan ke rantai DNA penyakit yang diprediksi (prediksi dimasukkan oleh pengguna).

Seseorang dianggap memiliki penyakit tertentu ketika rantai DNA penyakit menjadi salah satu substring dari string DNA orang tersebut. Tahap ini adalah tahap pencocokan rantai DNA penyakit dengan DNA masukan. Diterapkan dua algoritma pencocokan string untuk menyelesaikan masalah ini, yaitu Algoritma Knuth-Morris-Pratt dan Algoritma Boyer-Moore.

Salah satu fitur yang harus tersedia di program adalah *test history*, yaitu halaman yang akan menampilkan riwayat hasil tes oleh semua pengguna yang pernah melakukan *testing*. Pengguna dapat mencari riwayat tertentu dengan masukan khusus dari pengguna. Masukkan hanya terdiri dari tanggal dan jenis penyakit. Dalam menangani permasalahan ini, program menerapkan regex agar semua data yang sesuai dengan kriteria pencarian pengguna dapat ditampilkan.

Seluruh fitur program tersebut dibentuk dalam aplikasi berbasis website.

#### **3.2. Fitur Fungsional dan Arsitektur Aplikasi Web yang dibangun**

Aplikasi Web yang dibangun memiliki beberapa fitur fungsional diantaranya yaitu aplikasi dapat menambahkan penyakit ke database berdasarkan masukan nama penyakit dan *sequence* DNA-nya, memprediksi seseorang menderita penyakit berdasarkan

*sequence* DNA-nya, menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit, dan mencari hasil prediksi dengan input pencarian dari pengguna.

*Stack* yang digunakan pada aplikasi web ini terdiri dari Angular 13 (Frontend), Node.js Express (Backend) dan MySQL (Database). *Stack* tersebut memanfaatkan suatu RESTful API melalui HTTPClient dengan format JSON (body-parser) sehingga frontend dapat meminta request seperti POST dan GET dari backend yang tersambung dengan database. Bahasa pemrograman yang digunakan terutama adalah Javascript untuk backend serta HTML, CSS, dan Typescript untuk frontend.

Frontend dibangun menggunakan Angular, suatu Web Application Framework open-source yang dikembangkan di antara lain oleh Google. Framework ini berbasis Typescript dan menggunakan sistem components untuk membangun suatu web app. Dalam implementasi kami, terdapat 5 komponen utama yang disatukan oleh routing module angular. Components tersebut merupakan representasi setiap elemen atau halaman dalam web app, seperti component home untuk homepage, test untuk halaman testing DNA, dan lain-lain. Setiap component terdiri atas file HTML untuk struktur dasar halaman, SCSS untuk styling, dan Typescript untuk script serta logika halaman tersebut. Dalam implementasi CSS digunakan framework Bulma untuk mempermudah styling. Agar tidak harus passing terlalu banyak argumen ke backend dan mempermudah implementasi, algoritma utama seperti Booyer-Moore dan KMP di-import ke dalam Typescript yang component yang sesuai, sehingga berjalan langsung di Angular.

Sehingga, backend hanya digunakan untuk mendapatkan data dari database serta regex search. Node.js pada implementasi kami memanfaatkan Express module yang terdiri dari index.js utama untuk inisialisasi server serta koneksi ke database, dan router.js yang berisi API yang dapat digunakan oleh web app ini.

Karena keterbatasan masing-masing platform, deployment web app ini dilakukan di tiga tempat berbeda, yaitu Vercel untuk frontend, Heroku untuk backend API, serta Planetscale untuk menyimpan database.

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1. Spesifikasi Teknis Program

##### 4.1.1. Struktur Data

Program menyimpan data DNA penyakit dan DNA input pengguna. Data disimpan pada DBMS SQL yang di-*hosting* melalui PlanetScale. Struktur dari basis data dibuat menjadi 2 tabel:

- `test_result` yang menyimpan riwayat pengguna yang melakukan tes pada fitur tes DNA. Detail dari metadata tabel :

```
CREATE TABLE test_results (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  date DATE,  
  user VARCHAR(100),  
  disease VARCHAR(100),  
  percentage INT CHECK ( percentage <= 100 AND percentage >= 0 ),  
  result VARCHAR(5)  
);
```

- `diseases` yang menyimpan nama dan string DNA dari penyakit. Detail dari metadata tabel :

```
CREATE TABLE diseases (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  name VARCHAR(100),  
  dna_sequence VARCHAR(1000)  
);
```

##### 4.1.2. Fungsi dan Prosedur

###### 1. Algoritma Backend

- Algoritma Boyer-Moore
  - boyermoore
  - lxcount
- Algoritma Knuth-Morris-Pratt
  - kmp
  - failure

- Algoritma sanitasi : input sanitation
- Algoritma hamming distance (Bonus)
  - hamming distance
  - hamming percentage
  - hamming process

## 2. Algoritma Frontend

- Fungsi untuk form Frontend di masing-masing component:
  - onFileSelected()
  - onSubmit()
  - onSearch()

## 4.2. Tata Cara Penggunaan Program

### 4.2.1. Cara untuk menjalankan program

Aplikasi Web dapat diakses melalui link berikut:

<https://dnaregex.vercel.app/>

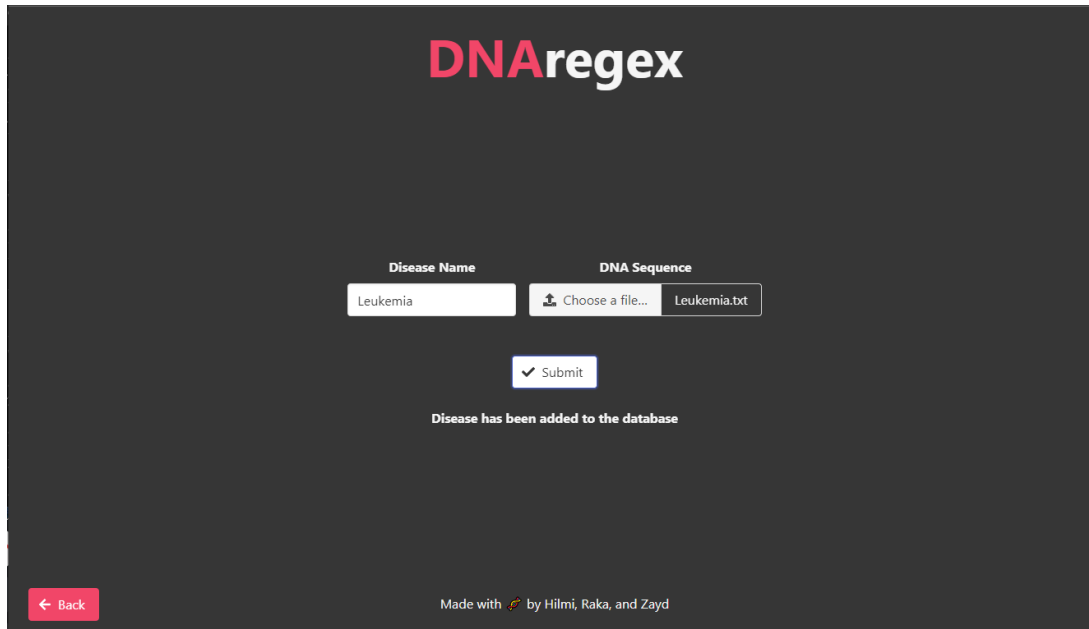
### 4.2.2. Cara untuk menggunakan program

3. Pilih fitur yang ingin digunakan  
(untuk menunjukkan cara menggunakan program, urutan yang akan ditunjukkan adalah menambah penyakit, DNA *testing*, kemudian search *test history*).
4. Tekan tombol bertuliskan “Add Disease”, masukkan nama penyakit dan file .txt yang mengandung rantai DNA penyakit. Apabila rantai sudah ada di basis data, penyakit tidak akan bisa ditambahkan ke basis data.
5. Keluar dan masuk ke fitur “Tes DNA”
6. Masukkan nama, file .txt yang mengandung rantai DNA, prediksi penyakit, dan algoritma yang ingin digunakan untuk melakukan pengetesan.
7. Apabila prediksi penyakit ada pada basis data, program akan melanjutkan ke tahap penerapan algoritma pencocokan string dan mengeluarkan hasil pencocokan. True apabila DNA input mengandung persis rantai DNA penyakit yang diprediksi / lebih dari 80% rantai DNA penyakit yang diprediksi, dan False apabila selain syarat True.
8. Keluar dan masuk ke fitur “Search Test Result”

9. Tulis tanggal (22 Februari 2022 / 22-02-2022) diikuti dengan nama penyakit pada kolom pencarian. Program akan menampilkan riwayat pencarian yang sesuai dengan kriteria pengguna.

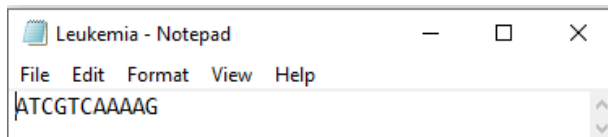
#### 4.3. Hasil Pengujian

1. Menambahkan penyakit ke database
  - a. Penyakit berhasil dimasukkan ke dalam database



The screenshot shows the DNAregex web application interface. At the top, the logo "DNAregex" is displayed in red and white. Below the logo, there are two input fields: "Disease Name" and "DNA Sequence". The "Disease Name" field contains the text "Leukemia". The "DNA Sequence" field has a "Choose a file..." button and a file named "Leukemia.txt" is selected. Below these fields is a "Submit" button with a checkmark icon. Under the "Submit" button, a message states "Disease has been added to the database". At the bottom left, there is a "Back" button with a left arrow icon. At the bottom right, it says "Made with by Hilmi, Raka, and Zayd".

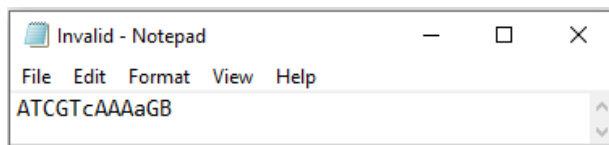
Isi DNA :



- b. Penyakit gagal dimasukkan ke dalam database

The screenshot shows the DNAregex web application interface. The 'Disease Name' field contains 'Covid'. The 'DNA Sequence' field has a file upload button and a file named 'Invalid.txt'. Below the fields is a 'Submit' button. A message below the submit button states: 'DNA sequence contains invalid characters'. At the bottom left is a 'Back' button, and at the bottom center is the text 'Made with by Hilmi, Raka, and Zayd'.

Isi DNA :



Gagal dimasukkan karena terdapat huruf yang tidak kapital dan mengandung karakter 'B'

## 2. Tes DNA

- a. Tes gagal dilakukan karena nama penyakit tidak ada di dalam database

The screenshot shows the DNAregex web application interface. The 'Name' field contains 'Joni'. The 'DNA Sequence' field has a file upload button and a file named 'Test2 - Leukemia.txt'. The 'Disease Prediction' field contains 'Begadang'. Below the fields is a 'Submit' button. Below the submit button, the text 'Test Result:' is displayed, followed by 'Disease not found'. At the bottom left is a 'Back' button, and at the bottom center is the text 'Made with by Hilmi, Raka, and Zayd'.

b. Tes gagal dilakukan karena masukan DNA bukan DNA yang valid

The screenshot shows the DNAregex web application interface. The 'Name' field contains 'Joni'. The 'DNA Sequence' field contains 'Test2 - Leukemia Invalid.txt'. The 'Disease Prediction' field contains 'Leukemia'. The 'Select Algorithm' section shows 'KMP' selected. A 'Submit' button is visible. Below the form, the 'Test Result:' section displays the message 'DNA sequence contains invalid characters'. At the bottom, there is a 'Back' button and a footer that reads 'Made with by Hilmi, Raka, and Zayd'.

Isi DNA :

The screenshot shows a Notepad window titled 'Test2 - Leukemia Invalid - Notepad'. The text inside the window is as follows:

```
TCAAGTGCCTCAGCTAAAGTCGTGTCGACGAGThGTTGCGACCAAGTCGAATCTGATGGTGGCTCGATT  
GGGTCTAAGGGTTTACCTCTGTGTTCAATTGGCATTAGATCGCCAGTTACTGCAGATCGAAGTTTCAA  
CTAGCTGCACAATCGCCTCCCGCCAGTTCGATCGTCAAAGCAAAACACCTCCAGGGATAAGTCAGTT  
GATGACCGAAC
```

Terdapat karakter yang bukan merupakan huruf kapital.

c. Tes berhasil dilakukan (true)

The screenshot shows the DNAregex web application interface. The 'Name' field contains 'Mamad'. The 'DNA Sequence' field contains 'HampirImba.txt'. The 'Disease Prediction' field contains 'Imba'. The 'Select Algorithm' section shows 'KMP' selected. A 'Submit' button is visible. Below the form, the 'Test Result:' section displays the message '28 April 2022 - Mamad - Imba - 96% - True'. At the bottom, there is a 'Back' button and a footer that reads 'Made with by Hilmi, Raka, and Zayd'.



d. Tes berhasil dilakukan (false)

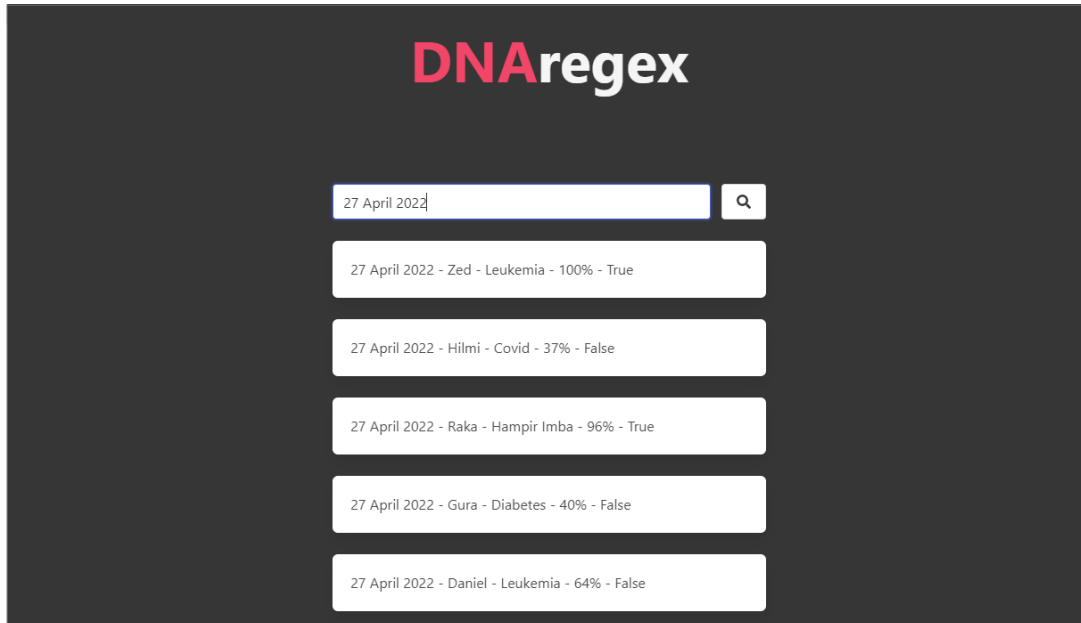
The screenshot shows the DNAregex application interface. At the top, the logo "DNAregex" is displayed in red and white. Below the logo, there are three input fields: "Name" with the value "Joni", "DNA Sequence" with a file upload button and the text "Test2 - Leukemia.txt", and "Disease Prediction" with the value "Leukemia". Below these fields, there is a "Select Algorithm:" section with two radio buttons: "KMP" (selected) and "Boyer-Moore". A "Submit" button is located below the algorithm selection. The "Test Result:" section displays the text "28 April 2022 - Joni - Leukemia - 48% - False". At the bottom left, there is a "Back" button. At the bottom right, there is a footer that says "Made with by Hilmi, Raka, and Zayd".

3. Pencarian hasil tes DNA

a. Pencarian berdasarkan tanggal dan penyakit

The screenshot shows the DNAregex application interface. At the top, the logo "DNAregex" is displayed in red and white. Below the logo, there is a search bar with the text "28 april 2022 Leukemia" and a search button. Below the search bar, there are three search results displayed in white boxes: "28 April 2022 - Joni - Leukemia - 48% - False", "28 April 2022 - Ucup - Leukemia - 57% - False", and "28 April 2022 - Agus - Leukemia - 57% - False". At the bottom left, there is a "Back" button. At the bottom right, there is a footer that says "Made with by Hilmi, Raka, and Zayd".

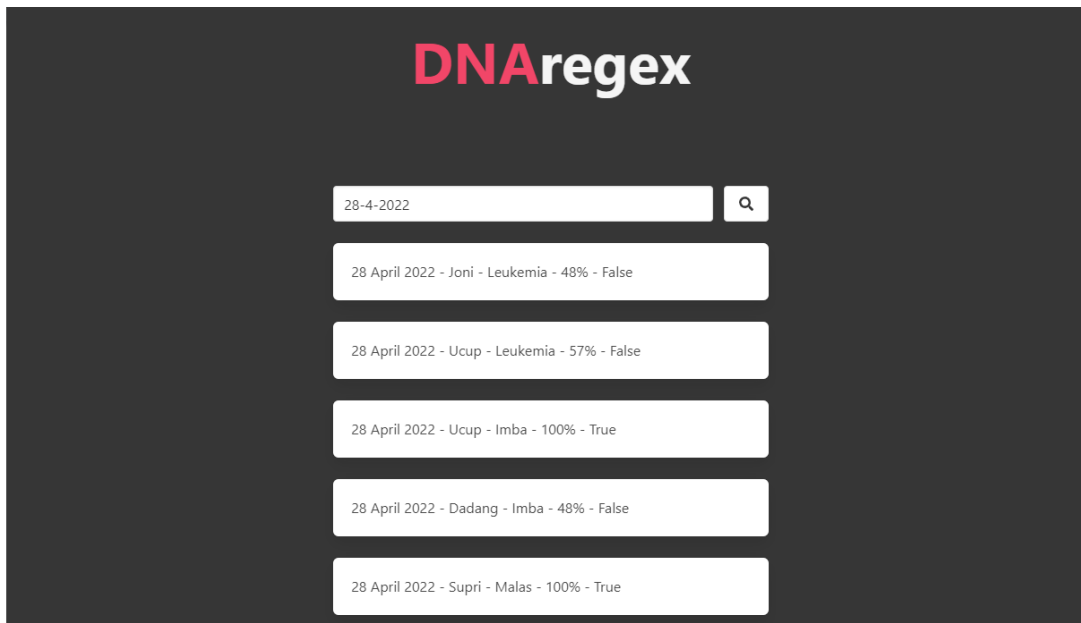
b. Pencarian berdasarkan tanggal (Format DD MM YEAR)



The screenshot shows the DNAregex search interface. The search bar contains the text "27 April 2022" and a search icon. Below the search bar, there are six result cards, each displaying a date, a name, a condition, a percentage, and a boolean value.

| Date          | Name   | Condition   | Percentage | Boolean |
|---------------|--------|-------------|------------|---------|
| 27 April 2022 | Zed    | Leukemia    | 100%       | True    |
| 27 April 2022 | Hilmi  | Covid       | 37%        | False   |
| 27 April 2022 | Raka   | Hampir Imba | 96%        | True    |
| 27 April 2022 | Gura   | Diabetes    | 40%        | False   |
| 27 April 2022 | Daniel | Leukemia    | 64%        | False   |

c. Pencarian berdasarkan tanggal (Format DD-MM-YEAR)



The screenshot shows the DNAregex search interface. The search bar contains the text "28-4-2022" and a search icon. Below the search bar, there are six result cards, each displaying a date, a name, a condition, a percentage, and a boolean value.

| Date          | Name   | Condition | Percentage | Boolean |
|---------------|--------|-----------|------------|---------|
| 28 April 2022 | Joni   | Leukemia  | 48%        | False   |
| 28 April 2022 | Ucup   | Leukemia  | 57%        | False   |
| 28 April 2022 | Ucup   | Imba      | 100%       | True    |
| 28 April 2022 | Dadang | Imba      | 48%        | False   |
| 28 April 2022 | Supri  | Malas     | 100%       | True    |

d. Pencarian berdasarkan penyakit

DNAregex

Imba

28 April 2022 - Ucup - Imba - 100% - True

28 April 2022 - Dadang - Imba - 48% - False

28 April 2022 - Mamad - Imba - 96% - True

← Back

Made with by Hilmi, Raka, and Zayd

#### 4.4. Analisis Hasil Pengujian

Berdasarkan hasil pengujian yang telah dilakukan, aplikasi dapat menjalankan fitur-fitur yang tersedia dengan baik. Algoritma *string matching* baik itu KMP dan Boyer-Moore yang dibuat dapat memeriksa apakah terdapat DNA penyakit dalam DNA penggunanya. Lalu, fungsi Hamming distance yang dibuat dapat memeriksa tingkat kemiripan antara DNA penyakit dengan DNA pengguna pada fitur tes DNA. Selain itu, fungsi sanitasi berhasil memeriksa jika inputan DNA bukan merupakan DNA yang valid.

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1. Kesimpulan**

Algoritma Knuth-Morris-Pratt dan Boyer-Moore yang tujuannya untuk melakukan pencocokan string ternyata dapat diterapkan ke berbagai macam kebutuhan di dunia nyata yang salah satu contohnya adalah aplikasi sederhana untuk pencocokan string DNA dengan rantai DNA penyakit.

#### **5.2. Saran**

Saran yang dapat kelompok kami sampaikan adalah perlunya *work planning* yang mengakibatkan pengimplementasian kode program bisa dilakukan secara modular, karena akan sangat mempersingkat waktu pengerjaan.

#### **5.3. Refleksi**

Tugas Besar 3 IF2211 Strategi Algoritma 2021/2022 merupakan tugas yang sangat menarik karena menggabungkan 3 ilmu dasar yang berbeda, yaitu strategi algoritma, basis data, dan *web development*. Tugas-tugas seperti ini akan sangat membantu mempercepat proses belajar apabila ditugaskan pada waktu yang sesuai (tidak terlalu cepat karena tidak ada dasar yang sudah pernah dipelajari atau tidak terlalu terlambat karena ilmu dasar yang digunakan sudah cukup lama dipelajari).

## DAFTAR LINK

Link Deploy :

<https://dnaregex.vercel.app/>

Link Github :

[https://github.com/rkvilena/Tubes3\\_13520134](https://github.com/rkvilena/Tubes3_13520134)

Link Video :

<https://youtu.be/sqlSjIahS5A>

## DAFTAR PUSTAKA

- Munir, Rinaldi. Pencocokan String (String/Pattern Matching). Institut Teknologi Bandung.  
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf> . Diakses pada 17 April 2022.
- Leylia Khodra, Masayu. String Matching dengan Regular Expression. Institut Teknologi Bandung.  
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf> . Diakses pada 17 April 2022.
- Schmitz, Holger. (2021). How to work with Angular and MySQL.  
<https://developer.okta.com/blog/2019/08/16/angular-mysql-express>. Diakses pada 23 April 2022.
- Potts, Tyler. (2020). Build Your First ANGULAR Web app ~ Beginner Angular Todo app.  
<https://www.youtube.com/watch?v=i7KaVFOXNUQ>. Diakses pada 23 April 2022.