

---

Tugas Besar IF2211 Strategi Algoritma

---

Pemanfaatan String Matching dalam Pengecekan Hubungan DNA  
dan Penyakit.

---

Semester I Tahun 2021/2022

---



Disusun oleh:

Jaya Mangalo Soegeng Rahardjo 13520015

Muhammad Gerald Akbar Gifferra 13520143

Aldwin Hardi Swastia 13520167

Institut Teknologi Bandung  
Sekolah Teknik Elektro dan Informatika

# DAFTAR ISI

<b>DAFTAR ISI .....</b>	<b>2</b>
<b>Bab I.....</b>	<b>3</b>
<b>Deskripsi Tugas.....</b>	<b>3</b>
<b>Bab II .....</b>	<b>9</b>
<b>Landasan Teori .....</b>	<b>9</b>
<b>2.1 Dasar Teori .....</b>	<b>9</b>
<b>Bab III.....</b>	<b>12</b>
<b>Analisis Pemecahan Masalah .....</b>	<b>12</b>
<b>3.1 Langkah Penyelesaian Masalah.....</b>	<b>12</b>
<b>3.2 Fitur Fungsional dan Arsitektur Aplikasi Web .....</b>	<b>12</b>
<b>3.2.1 Fitur fungsional.....</b>	<b>12</b>
<b>3.2.2 Arsitektur Aplikasi Web .....</b>	<b>13</b>
<b>Bab IV .....</b>	<b>14</b>
<b>Implementasi dan Pengujian .....</b>	<b>14</b>
<b>4.1 Spesifikasi Teknis Program .....</b>	<b>14</b>
<b>4.2 Penjelasan Tata Cara Penggunaan Program .....</b>	<b>19</b>
<b>4.3 Hasil Pengujian .....</b>	<b>20</b>
<b>4.4 Analisis Hasil Pengujian.....</b>	<b>22</b>
<b>Bab V .....</b>	<b>24</b>
<b>Kesimpulan dan Saran .....</b>	<b>24</b>
<b>Daftar Pustaka .....</b>	<b>25</b>
<b>Links .....</b>	<b>25</b>

# Bab I

## 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.

### 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" (Add Disease). It contains two input fields: "Nama Penyakit:" (Disease Name) with a placeholder text "penyakit..." and "Sequence DNA:" with a placeholder text "upload file...". Below these fields is a green "Submit" button.

Gambar 1. Ilustrasi Input Penyakit

2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.

- Tes DNA dilakukan dengan menerima input nama pengguna, sequence DNA pengguna, dan nama penyakit yang diuji. Asumsi sequence DNA pengguna > sequence DNA penyakit.
- 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).
- Pencocokan sequence DNA dilakukan dengan menggunakan algoritma string matching.
- 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
- Semua komponen hasil tes ini dapat ditampilkan pada halaman web (refer ke poin 3 pada “Fitur-Fitur Aplikasi”) dan disimpan pada sebuah tabel database.
- Contoh tampilan web:

**Tes DNA**

Nama Pengguna:

Sequence DNA:

Prediksi Penyakit:

---

**Hasil Tes**

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

Gambar 2. Ilustrasi Prediksi

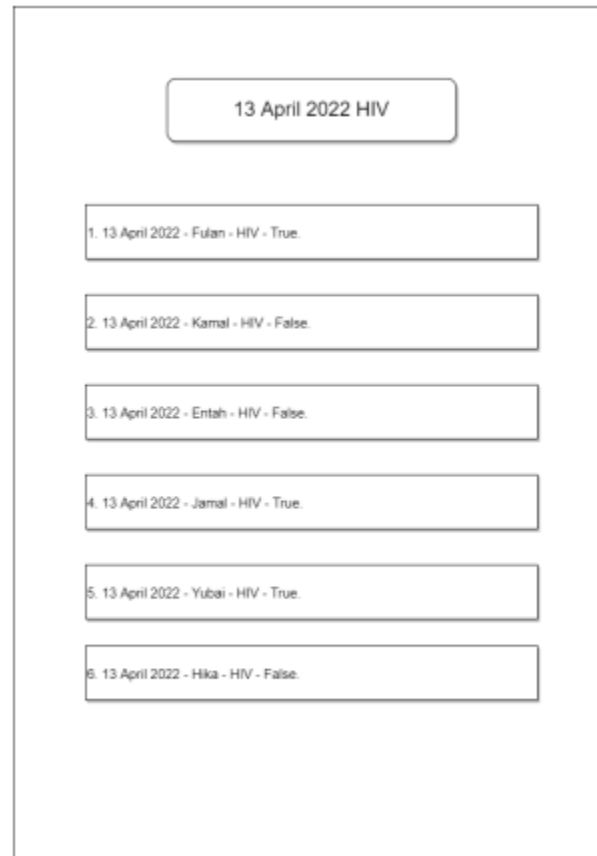
3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.

- 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.

c. Contoh ilustrasi:

i. Masukan tanggal dan nama penyakit



The image shows a search interface. At the top, there is a search bar containing the text "13 April 2022 HIV". Below the search bar, there is a list of 6 results, each displayed in a separate box. The results are as follows:

No.	Search Result
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.

Gambar 3. Ilustrasi Interaksi 1

ii. Masukan hanya tanggal

13 April 2022

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

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

3. 13 April 2022 - Entah - Down Syndrome - False.

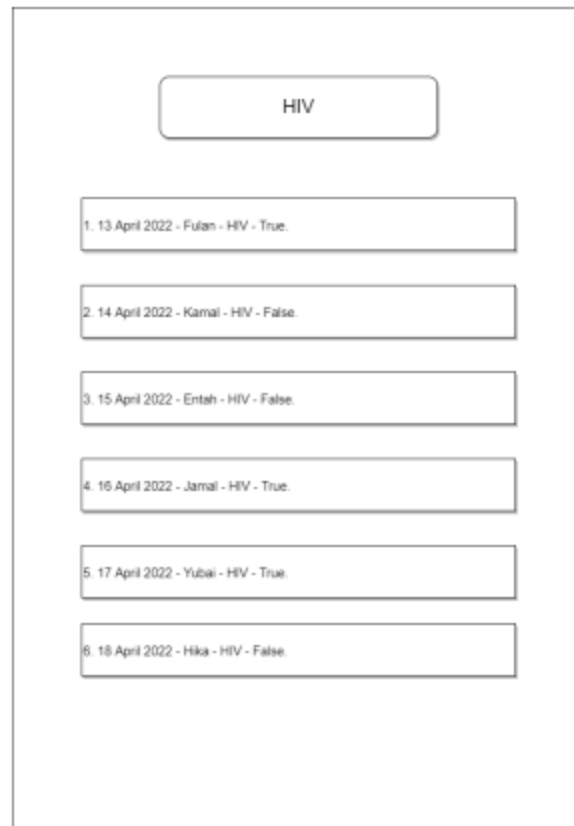
4. 13 April 2022 - Jamal - Polio - True.

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

6. 13 April 2022 - Hika - Hepatitis A - False.

Gambar 4. Ilustrasi Interaksi 2

iii. Masukan hanya nama penyakit



Gambar 5. Ilustrasi Interaksi 3

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.
  - d. Contoh tampilan:

### Tes DNA

Nama Pengguna:  
<pengguna>

Sequence DNA:  
upload file...

Prediksi Penyakit:  
<penyakit>

Submit

---

Hasil Tes

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



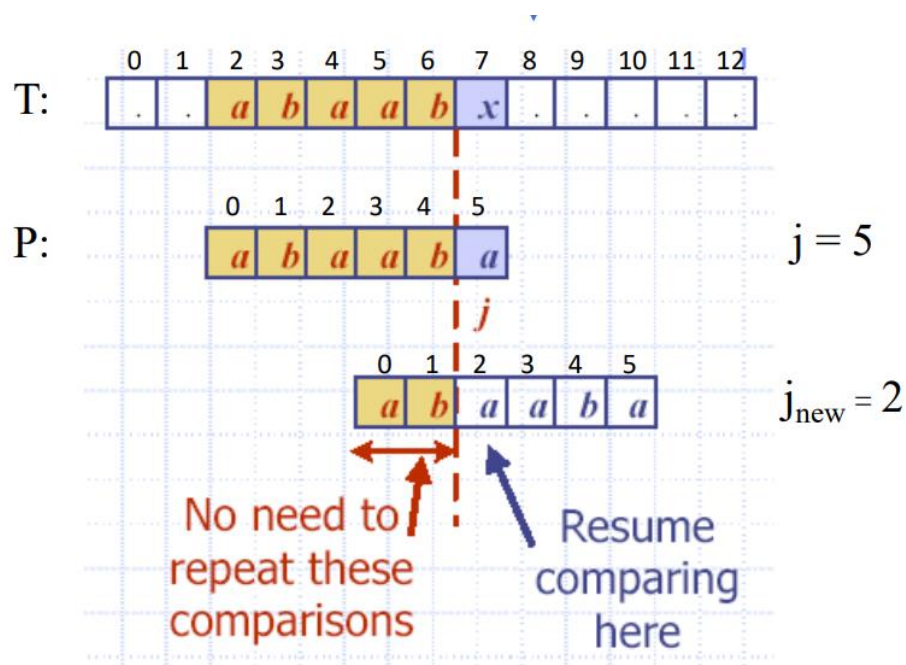
## Bab II

### Landasan Teori

#### 2.1 Dasar Teori

##### Knuth-Morris-Pratt Algorithm

Knuth-Morris-Pratt Algorithm atau lebih sering disebut KMP, adalah algoritma string matching yang mirip dengan algoritma brute force dalam string matching. Ia bekerja mirip brute force dengan bergerak dari kiri ke kanan, pengecekan juga dari kiri ke kanan. Bedanya terletak di lompatannya setiap gagal matching, program akan melompat secara lebih pintar jika dibandingkan dengan brute force.

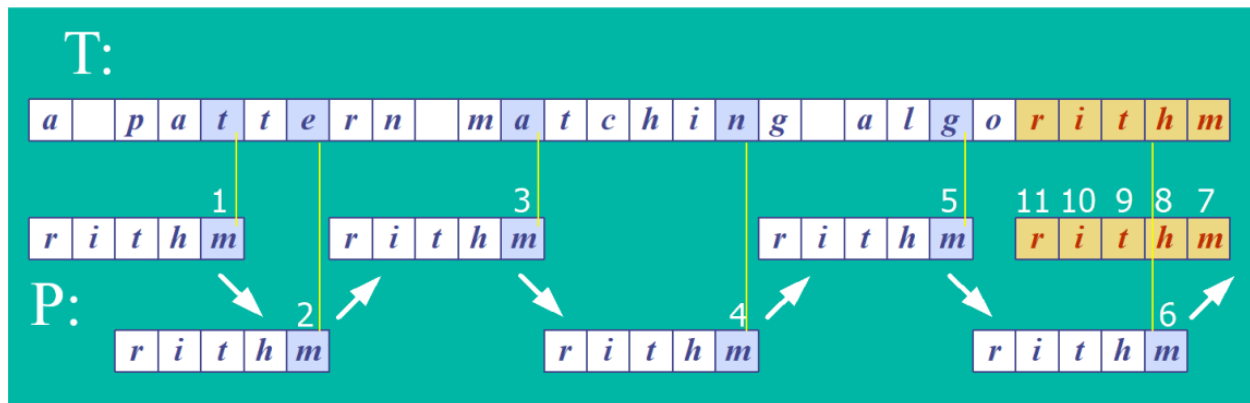


Pada contoh berikut, program melompat sebanyak 3 karakter karena pelompatan 1-2 karakter tidak akan menghasilkan match.

##### Boyer-Moore Algorithm

Boyer-Moore Algorithm adalah algoritma string matching yang bergerak dari kiri ke kanan, tetapi dengan pengecekan dari kanan ke kiri. Jika terjadi ketidak-cocokan dalam pengecekan

suatu karakter, maka program akan menempatkan karakter pattern yang cocok dengan karakter yang di cek (jika ada).



Pada contoh berikut, terlihat bahwa di no 1, terjadi mismatch antara t dan m, sehingga algoritma mengeser pattern sehingga “t” di pattern terletak tepat dengan “t” di text. Untuk pengecekan no2, karena terjadi mismatch dengan “e” dan pattern “rithm” tidak mengandung e, maka semua pattern digeser melewati “e”. Hal begitu di ulang terus sampai mencapai end state.

### Longest Common Substring

LCS adalah sebuah algoritma yang mirip seperti FPB, program mencari substring terbesar yang terdapat dalam dua string yang berbeda. Sebuah metode dasar tetapi tidak efisien dalam masalah yang terlalu besar adalah menggunakan rekursi. Program memecah string-string menjadi string yang lebih kecil lalu dibandingkan.

Untuk pengecekan DNA, LCS digunakan untuk mencari panjang Longest common substring yang kemudian dibagi dengan panjang pattern DNA penyakit untuk menghasilkan besar persentase kemiripan.

### Regular Expression

Regular Expression adalah sebuah notasi yang dapat digunakan untuk mencari pattern/string dengan kompleksitas yang lebih tinggi.

Regex masih bisa dipakai untuk mencari string secara exact, misalnya notasi “ITB”, akan mencari semua kemunculan “ITB” dalam sebuah string.

Regex tetapi juga mampu untuk mencari pattern lebih kompleks misalnya dengan notasi “d{4}”, notasi ini akan mencari semua kemunculan angka 4 digit (biasanya dipakai untuk mencari tahun).

Regex juga memiliki banyak kemampuan lain yang terdokumentasi dalam <https://yourbasic.org/golang/regexp-cheat-sheet/> untuk bahasa golang.

## 2.2 Aplikasi Web

Aplikasi web yang kami bangun memanfaatkan Bahasa pemrograman golang sebagai dasar pembangunan algoritma, dan juga sebagai dasar pembangunan servis *server-side*, dari websiter dengan memanfaatkan framework yang bernama *gin*. Untuk pembangunan tampilan website kami memanfaatkan framework React.js sehingga bisa dibuat tampilan yang lebih menari dan interaktif. Terdapat 3 fitur utama pada aplikasi web yaitu melakukan tes dna, melakukan searching hasil tes dna berdasarkan tanggal dan penyakit, dan yang terakhir menambahkan data penyakit kedalam database.

## **Bab III**

### **Analisis Pemecahan Masalah**

#### **3.1 Langkah Penyelesaian Masalah**

##### **Fitur Penambahan Penyakit**

Untuk Fitur ini, penyakit akan dibaca menggunakan file uploader yang kemudian isinya akan disanitasi menggunakan Regex. Jika sanitasi valid, maka penyakit akan dimasukkan ke database dan jika invalid maka akan ditolak.

##### **Fitur Tes DNA**

Untuk Fitur ini, DNA pengguna akan dibaca menggunakan file uploader yang kemudian isinya akan disanitasi menggunakan Regex. Jika sanitasi valid, maka penyakit akan dimasukkan ke database dan jika invalid maka akan ditolak.

Jika valid, maka DNA tersebut akan dicocokkan dengan DNA penyakit pilihan menggunakan algoritma antara KMP, BM, atau LCS yang sudah dibuat. Setelah pencocokan datanya disimpan.

##### **Fitur Pencarian Hasil Prediksi**

Program akan menerima fungsi string yang kemudian akan dicocokkan ke beberapa pattern regex. Ada beberapa pattern regex sehingga banyak input yang bisa diterima. Setelah pencocokan, maka string input akan dipecahkan menjadi tanggal, bulan, tahun, dan nama penyakit. Data tersebut kemudian akan digunakan sebagai bagian dari search query SQL.

#### **3.2 Fitur Fungsional dan Arsitektur Aplikasi Web**

##### **3.2.1 Fitur fungsional**

Terdapat 3 fitur fungsional utama pada aplikasi web yaitu:

- Fitur menambahkan data penyakit  
Pada fitur ini web menerima masukan berupa file .txt yang berisi sequence DNA sebuah penyakit, dan juga nama penyakit yang akan di input. Setelah menekan tombol submit maka data penyakit akan dimasukkan kedalam database sehingga bisa digunakan untuk tes penyakit selanjutnya.
- Fitur melakukan tes DNA

Pada fitur ini, web menerima masukan nama pengguna, file .txt yang berisi sequence DNA pengguna, dan juga nama penyakit yang ingin diperiksa. Setelah mengisi semua hal tersebut dan menekan tombol submit, maka hasil tes pengguna akan dimunculkan beserta tingkat kemiripan DNA-nya. Selain itu hasil tes ini akan langsung di input kedalam database sebagai riwayat diagnosis tes yang sudah dilakukan.

- Fitur searching hasil tes DNA berdasarkan bulan dan penyakit

Pada fitur ini, web menerima masukan berupa string yang diinput oleh pengguna, string ini dapat berupa tanggal lengkap/tidak lengkap, nama penyakit dan nama pengguna

### 3.2.2 Arsitektur Aplikasi Web

Aplikasi web ini kami kembangkan dengan terdiri dari 3 bagian yaitu bagian algoritma string matching menggunakan Bahasa pemrograman Golang, bagian pembuatan api yang juga menggunakan Golang, dan yang terakhir tampilan website yang memanfaatkan React.js. Bagian algoritma string matching bisa dilihat pada package *algorithm* yang kami kembangkan, sedangkan bagian api terdiri dari 3 bagian yaitu handler, connector dan main program. Connector berfungsi sebagai jalan koneksi kedalam database, handler berfungsi sebagai sarana handling dan routing untuk api yang kami buat sehingga dapat mengambil/menambah data dari database dan mengirimkannya ke front-end. Main program berfungsi untuk menjalankan server sehingga api dapat digunakan. Bagian front-end memanfaatkan React.js dan juga library *axios*, sehingga frontend dapat berinteraksi dengan database menggunakan api yang kami kembangkan sebelumnya.

## Bab IV

### Implementasi dan Pengujian

#### 4.1 Spesifikasi Teknis Program

##### Struktur Data

.struct.go : Struct yang digunakan untuk membantu fungsi-fungsi regex

```
type SearchRegex struct {  
    Day      string  
    Month    string  
    Year      string  
    Disease  string  
}  
type RegexFormat struct {  
    regex      string  
    splitter  string  
}
```

## Fungsi dan Prosedur Penting

Regex.go : digunakan untuk sanitasi input dan searching query

```
//Mengecek apakah input sesuai DNA
func IsSanitizedRegex(text string) bool {
    match1, err := regexp.MatchString("^[ATCG]+$", text)
    if err == nil {
        return match1
    }
    return false
}

//Mengecek text dengan format regex yang disediakan dan fungsi StringRegex
func ReadRegex(text string) SearchRegex {
    var regexformats [11]RegexFormat
    regexformats[0] = RegexFormat{"\\d{2}/\\d{2}/\\d{4}", "/" }
    regexformats[1] = RegexFormat{"\\d{2}-\\d{2}-\\d{4}", "-"}
    regexformats[2] = RegexFormat{"\\d{2}[ \\t]+\\d{2}[ \\t]+\\d{4}", " "}

    regexformats[3] = RegexFormat{"\\d{2}/.+\\d{4}", "/" }
    regexformats[4] = RegexFormat{"\\d{2}-.+\\d{4}", "-"}
    regexformats[5] = RegexFormat{"\\d{2}[ \\t]+.+\\d{4}", " "}

    regexformats[6] = RegexFormat{".+\\d{4}", "/" }
    regexformats[7] = RegexFormat{".+\\d{4}", "-"}
    regexformats[8] = RegexFormat{".+[ \\t]+\\d{4}", " "}

    regexformats[9] = RegexFormat{"^\\d{4}", "/$"}

    regexformats[10] = RegexFormat{".+", " "}

    var date SearchRegex
    var get bool
    for _, v := range regexformats {
        date, get = StringRegex(text, v.regex, v.splitter)
        fmt.Println(v.regex)
        if get {
            return date
        }
    }
    date = SearchRegex{"null", "null", "null", "null"}
    return date
}
```

```

//Memecah informasi text dengan regex kedalam SearchRegex yang lalu di-return bersama bool yang menyatakan apakah regex valid.
func StringRegex(text, regex, splitter string) (SearchRegex, bool) {
    regexcompiled := regexp.MustCompile(regex)

    if regexcompiled.MatchString(text) {
        fmt.Println("TEXT :",text )
        date := regexcompiled.FindString(text)
        res := strings.TrimSpace(regexcompiled.ReplaceAllString(text, "${1} $2"))

        datesplitted := strings.Split(date, splitter)
        fmt.Println("DATESPLITTER = ",len(datesplitted))
        if splitter == " " {
            datesplitted = strings.Fields(date)
        }
        var day string
        var month string
        var year string
        if len(datesplitted) == 3 {
            day = datesplitted[0]
            month = datesplitted[1]
            year = datesplitted[2]
        } else if len(datesplitted) == 2 {
            day = "null"
            month = datesplitted[0]
            year = datesplitted[1]
        } else {
            day = "null"
            regxyyear := regexp.MustCompile("\\d{4}")
            if regxyyear.MatchString(datesplitted[0]) {
                month = "null"
                year = datesplitted[0]
            } else if (checkIfDate(datesplitted[0])){
                month = datesplitted[0]
                year = "null"
            }else{
                month = "null"
                year = "null"
                res = strings.TrimSpace(datesplitted[0])
            }
        }
        if res == ""{
            res = "null"
        }
        fmt.Println("TEXT :",text )
        var x = SearchRegex{day, month, year, res}
        return x, true
    } else {
        var x = SearchRegex{"null", "null", "null", "null"}
        return x, false
    }
}

```



KMP.go : searching text dengan algoritma KMP

```
//fungsi text searching dengan algo KMP
func KMP(text, pattern string) int {
    var n int = len(text)
    var m int = len(pattern)

    var fail = computeFail(pattern)

    var i int = 0
    var j int = 0
    for i < n {
        if pattern[j] == text[i] {
            if j == m-1 {
                return i - m + 1
            }
            i++
            j++
        } else if j > 0 {
            j = fail[j-1]
        } else {
            i++
        }
    }
    return -1
}

//fungsi bantuan untuk KMP
func computeFail(pattern string) []int {
    var m int = len(pattern)

    var fail = make([]int, m)
    fail[0] = 0

    var j int = 0
    var i int = 1
    for i < m {
        if pattern[j] == pattern[i] {
            fail[i] = j + 1
            i++
            j++
        } else if j > 0 {
            j = fail[j-1]
        } else {
            fail[i] = 0
            i++
        }
    }
    return fail
}
```

BoyerMoore.go : searching text dengan algoritma BoyerMoore

```
//fungsi text searching dengan algo BoyerMoore
func BoyerMoore(text, pattern string) int {
    var last [128]int = buildLast(pattern)
    var n int = len(text)
    var m int = len(pattern)
    var i = m - 1

    if i > n-1 {
        return -1
    }

    var j = m - 1

    for {
        if pattern[j] == text[i] {
            if j == 0 {
                return i
            } else {
                i--
                j--
            }
        } else {
            var lo int = last[text[i]]
            i = i + m - min(j, i+lo)
            j = m - 1
        }
        if i > n-1 {
            break
        }
    }
    return -1
}

//fungsi bantuan untuk BoyerMoore
func buildLast(pattern string) [128]int {
    var last [128]int
    for i := 0; i < 128; i++ {
        last[i] = -1
    }
    for i := 0; i < len(pattern); i++ {
        last[pattern[i]] = i
    }
    return last
}
```

LCS.go: Mencari kemiripan antara pattern dan text dengan algoritma Longest Common Substring

```
//fungsi bantuan yang digunakan untuk mencari kemiripan pattern
//dengan text dalam bentuk integer persentase yang direturn
func LCSPercentage(text, pattern string) int {
    same := LCS(text, pattern, len(text), len(pattern), 0)
    percentage := same * 100 / len(pattern)
    return percentage
}

//fungsi rekursif Least Common Substring, me return panjang substring terbesar
func LCS(X, Y string, i, j, count int) int{
    if (i == 0 || j == 0){
        return count;
    }

    if (X[i - 1] == Y[j - 1]){
        count = LCS(X, Y, i - 1, j - 1, count + 1);
    }
    count = max(count, max(LCS(X, Y, i, j - 1, 0), LCS(X, Y, i - 1, j, 0)));
    return count;
}
```

## 4.2 Penjelasan Tata Cara Penggunaan Program

Program ini terdiri atas dua bagian yaitu *backend* dan *frontend*, sehingga untuk menjalankan program harus menjalankan kedua bagian ini, berikut tahap-tahapnya:

- Untuk menjalankan bagian *backend/server-side*, jalankan file `MainTester.go` untuk menjalankan server
- Untuk menjalankan bagian *frontend/client-side*, gunakan command `npm start` pada folder frontend
- Jika keduanya berhasil dijalankan, akan muncul output pada browser sebagai berikut:

### DNA Sequence Matcher

Cari Hasil Tes DNA

Tanggal>Nama

13 April 2022 - Alan - Diabetes - True - 50%

13 April 2022 - Bulan - Diabetes - True - 50%

13 April 2022 - Fulan - Diabetes - True - 50%

26 April 2022 - Gerald - HIV - True - 50%

26 April 2022 - Jaya - HIV - True - 50%

27 April 2022 - Aldwin - HIV - True - 96%

- d. Terdapat 3 menu pada aplikasi web yang masing-masing mempunyai fungsinya sendiri, yaitu menambah data penyakit kedalam database, fitur searching hasil diagnosis test yang sudah dilakukan, dan fitur melakukan tes dna terhadap pengguna.

## 4.3 Hasil Pengujian Upload Penyakit

Tambah Penyakit

Nama Penyakit

Sinusitis

Sequence DNA:

SINUSITIS.TXT

SUBMIT

## Hasil Upload Penyakit


**Before:**

nama	sequence_dna
HIV	CCGGTGCCCGGAATCTAGATCTGTGGCGCC
Polio	CGAGAATGGAGTTGGAGAACCGATGTAGAA

**After:**

nama	sequence_dna
HIV	CCGGTGCCCGGAATCTAGATCTGTGGCGCC
Polio	CGAGAATGGAGTTGGAGAACCGATGTAGAA
Sinusitis	TTTATCCCGGATTCCAGCGCTGGGATAAG

## Upload Penyakit

 Home   TambahPenyakit   TesDNA

### Tes DNA

Nama Pengguna

Jaya

Sequence DNA:

JAYA.TXT

Prediksi Penyakit

Covid

SUBMIT

29 April 2022 - Jaya - Covid - False - 23%

## Hasil Upload Penyakit

tanggal	nama	penyakit	diagnosis	similarity
13 April 2022	Alan	Diabetes	True	90
13 April 2022	Bulan	Diabetes	True	90
13 April 2022	Fulan	Diabetes	True	90
26 April 2022	Gerald	HIV	True	90
26 April 2022	Jaya	HIV	True	90
27 April 2022	Jaya	HIV	True	90
29 April 2022	Jaya	Covid	False	23

## Searching Tes DNA

### Cari Hasil Tes DNA

Tanggal>Nama

Covid

29 April 2022 - Jaya - Covid - False - 23%

### Cari Hasil Tes DNA

Tanggal>Nama

26 April 2022

26 April 2022 - Gerald - HIV - True - 90%

26 April 2022 - Jaya - HIV - True - 90%

## 4.4 Analisis Hasil Pengujian

Program bekerja dengan baik, Query pencarian hasil Tes DNA sedikit tidak elegan karena tidak

dapat menerima query secara full atau lengkap seperti search query aslinya pada sebuah website.  
(tetapi masih bisa query sesuai spesifikasi tugas besar)

Penambahan penyakit dan tes DNA memiliki bug kecil dimana jika terjadi Error di SQL user tidak diberitahu bahwa ada error.

## Bab V

### Kesimpulan dan Saran

Algoritma *String Matching* dan *Regular Expression* berhasil diimplementasikan untuk menyelesaikan persoalan *DNA Pattern Matching* dengan membuat sebuah aplikasi berbasis web. Aplikasi memiliki beberapa fitur, antara lain pencarian hasil tes, tambah penyakit, dan tes DNA. Pada pencarian hasil aplikasi akan melakukan pencarian dengan *regular expression* dan menampilkan semua datanya pada layar. Selanjutnya pada tambah penyakit aplikasi akan menambahkan penyakit pada *database*. Sedangkan pada tes DNA, DNA akan disamakan menggunakan algoritma string matching dengan target DNA penyakit.

Karena Tugas Besar ini masih tergolong “ringan”, sebuah saran untuk adik kelas tahun berikutnya adalah untuk dengan cepat menyelesaikan tugas ini.



## Daftar Pustaka

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

diakses pada 26 April 2022

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf> diakses pada 26 April 2022

<https://yourbasic.org/golang/regexp-cheat-sheet/> diakses pada 26 April 2022

<https://www.geeksforgeeks.org/longest-common-substring-dp-29/> diakses pada 29 April 2022

## Links

Repo Github : [https://github.com/JayaMangalo/Tubes3\\_13520015](https://github.com/JayaMangalo/Tubes3_13520015)

Video Demo : <https://youtu.be/wPTZJW3Qeyk>