

Laporan Tugas Besar 3
Penerapan *String Matching* dan *Regular Expression*
dalam *DNA Pattern Matching*
IF2211 Strategi Algoritma



Disusun Oleh:
Kelompok 10 - Not Ancestry Test
Muhammad Akmal Arifin (13520037)
Farnas Rozaan Iraquee (13520067)
Andika Naufal Hilmy (13520098)

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

2021/2022

DAFTAR ISI

DAFTAR ISI	2
BAB I DESKRIPSI TUGAS	3
BAB II LANDASAN TEORI	7
Deskripsi Algoritma KMP, BM, LCCS, dan Regex	7
Penjelasan Aplikasi Web	12
BAB III ANALISIS PEMECAHAN MASALAH	13
Langkah Penyelesaian Masalah Setiap Fitur	13
Fitur Fungsional Dan Arsitektur Aplikasi Web	13
BAB IV IMPLEMENTASI DAN PENGUJIAN	16
Spesifikasi Teknis Program	16
Tata Cara Penggunaan Program	17
Hasil Pengujian	20
BAB V KESIMPULAN, SARAN, DAN KOMENTAR/REFLEKSI	25
Kesimpulan	25
Saran	25
Komentar/Refleksi	25
DAFTAR PUSTAKA	26
LAMPIRAN	27

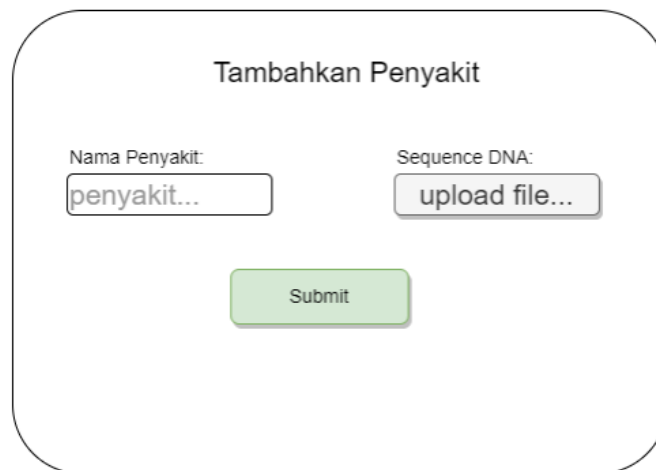
BAB I

DESKRIPSI TUGAS

Dalam tugas besar ini, kami diminta untuk membangun sebuah aplikasi *DNA Pattern Matching*. Dengan memanfaatkan algoritma *String Matching* dan *Regular Expression* yang telah kami pelajari di kelas IF2211 Strategi Algoritma, kami 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 1. 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*:

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.
 - a. Kolom pencarian dapat menerima masukan dengan struktur: `<tanggal_prediksi><spasi><nama_penyakit>` , 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:

- Masukan tanggal dan nama penyakit

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.

Gambar 3. Ilustrasi Interaksi 1

- 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

- 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:

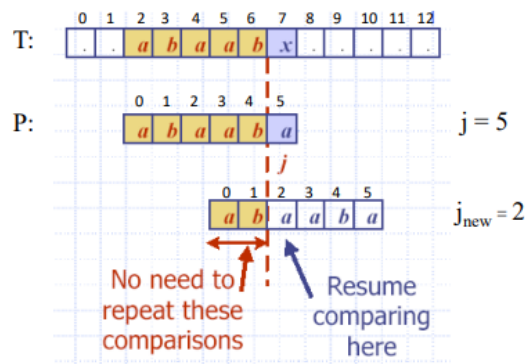
BAB II

LANDASAN TEORI

A. Deskripsi Algoritma KMP, BM, LCCS, dan Regex

1. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma KMP melakukan pencarian terhadap pola di dalam teks sama halnya dengan algoritma *brute force*, yaitu dimulai dari kiri ke kanan. Namun, algoritma KMP melakukannya dengan lebih cerdas daripada algoritma *brute force*. Misal jika terjadi ketidakcocokan antara teks (disimbolkan dengan T) dengan pola (disimbolkan dengan P) terjadi di $P[j]$, contohnya $T[i] \neq P[j]$, maka apa yang bisa kita lakukan agar tidak terjadi perbandingan yang sia-sia. Seperti yang sudah kita ketahui pada algoritma *brute force*, apabila terjadi ketidakcocokan maka proses pencocokan akan diulangi lagi dari awal dengan menggeser pola ke kanan sekali. Hal ini tentu saja akan menghasilkan perbandingan yang sia-sia dan hanya akan menghabiskan waktu saja. Namun, pada algoritma KMP proses penggeseran pola tersebut dilakukan sebesar prefiks terbesar dari P yang juga merupakan sufiks dari P. Berikut adalah ilustrasinya:



Gambar 7. Ilustrasi Penggeseran Saat Terjadi Ketidakcocokan (KMP)

Dari ilustrasi di atas dapat dilihat bahwa pergeseran dilakukan sebanyak 3 kali dan proses perbandingan terhadap 2 huruf awal pada pola tidak diulangi.

Untuk menghitung prefiks terbesar dari pola yang juga merupakan sufiks dari pola itu juga, maka sebelum melakukan proses pencocokan, algoritma KMP akan terlebih

dahulu melakukan praproses terhadap pola. Praproses ini sering disebut *border function* atau *failure function (fail)*. Berikut ini adalah contoh dari *border function*:

➤ P: abaaba
j: 012345

(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.

Gambar 8. Contoh Border Function

Jika *border function* dari suatu pola sudah dihitung, maka proses pencocokan sudah dapat dimulai. Berikut adalah contoh salah satu prosesnya:

T: a b a c a a b a c c a b a c a b a a b b

P: a b a c a b

7
a b a c a b

8 9 10 11 12
a b a c a b

13
a b a c a b

14 15 16 17 18 19
a b a c a b

j	0	1	2	3	4	5
P[j]	a	b	a	c	a	b
k	-	0	1	2	3	4
b(k)	-	0	0	1	0	1

Jumlah perbandingan karakter: 19 kali

Gambar 9. Contoh Proses String Matching dengan Algoritma KMP

Keuntungan dari algoritma KMP adalah algoritma ini tidak perlu untuk bergerak mundur pada masukan teks. Hal ini menyebabkan algoritma ini bagus untuk memproses *file* yang berukuran besar yang dibaca dari perangkat lain atau dari aliran jaringan. Sedangkan kerugiannya adalah algoritma KMP tidak bekerja dengan baik saat variasi alfabet yang ada pada teks meningkat. Hal tersebut menyebabkan kemungkinan terjadinya ketidakcocokan semakin besar dan ketidakcocokan tersebut cenderung terjadi di awal pola, sedangkan KMP akan bekerja lebih cepat saat ketidakcocokan terjadi di akhir.

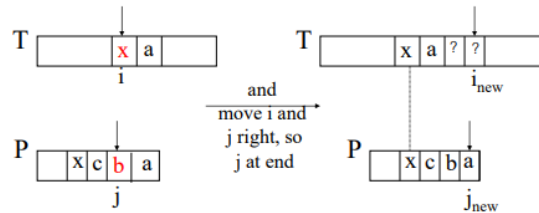
2. Algoritma Boyer-Moore (BM)

Algoritma BM bekerja berdasar pada 2 teknik, yaitu teknik *looking-glass* dan *character-jump*. Teknik *looking-glass* ini bekerja dengan mencari pola pada teks dengan

bergerak mundur dimulai dari akhir dari pola tersebut. Sedangkan, teknik *character-jump* merupakan teknik saat terjadi ketidakcocokan antara pola dengan teks. Terdapat 3 kasus yang mungkin terjadi pada teknik ini. Berikut adalah penjelasan dari masing-masing kasus tersebut, semisal jika ketidakcocokan terjadi di $T[i] = x$:

a. Kasus 1

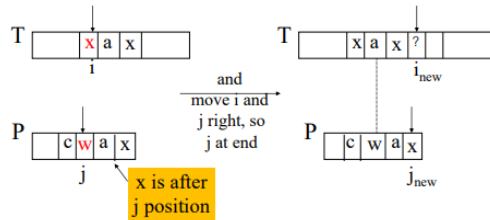
Jika x terdapat di suatu tempat pada P , maka coba geser P ke kanan untuk menyejajarkan kemunculan terakhir di x dengan $T[i]$.



Gambar 10. Kasus 1

b. Kasus 2

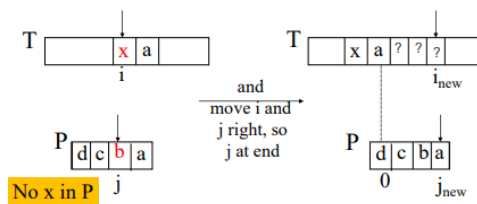
Jika x terdapat di suatu tempat pada P , tetapi pergeseran P ke kanan menuju kemunculan terakhir x tidak memungkinkan, maka geser P satu karakter ke kanan.



Gambar 11. Kasus 2

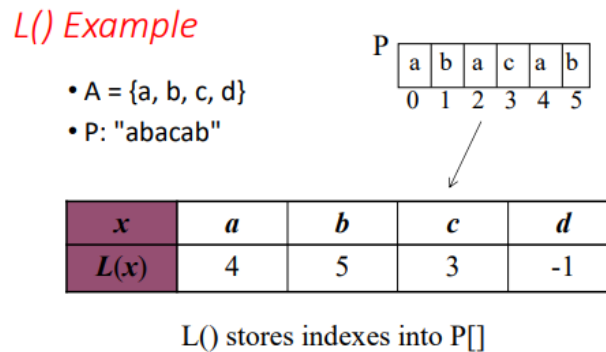
c. Kasus 3

Jika kondisi tidak memenuhi kasus 1 dan 2, maka geser P untuk menyejajarkan $P[0]$ dengan $T[i+1]$.



Gambar 12. Kasus 3

Apabila algoritma KMP memiliki praproses berupa *border function*, maka algoritma BM juga memiliki praproses yang biasa disebut *last occurrence function*. Algoritma BM melakukan praproses terhadap pola dan alfabet untuk membentuk *array* yang berisi indeks kemuculan terakhir suatu huruf pada pola. Berikut ini adalah contohnya:



Gambar 13. Contoh Last Occurrence Function

Jika praproses sudah dilakukan, maka proses pencocokan sudah dapat dilakukan. Berikut ini adalah contoh salah satu prosesnya:



Gambar 14. Contoh Proses String Matching dengan Algoritma BM

Beterbalikan dengan algoritma KMP yang akan bekerja dengan cepat saat ukuran alfabet kecil, maka algoritma BM bekerja cepat saat ukuran alfabet besar. Algoritma BM sangat baik untuk teks bahasa Inggris ataupun bahasa lainnya yang memiliki ukuran alfabet besar dan buruk untuk biner.

3. Algoritma *Longest Common Contiguous Subsequences* (LCCS)

Algoritma LCCS merupakan versi khusus dari algoritma *Longest Common Subsequences* (LCS). Algoritma LCS tidak memperhatikan kekontiguan dari *subsequence*, sedangkan algoritma LCCS memperhatikan kekontiguan tersebut karena dalam *string matching*, pola dikatakan cocok dengan teks apabila pola tersebut cocok

dengan *subsequence* dari teks secara kontigu. Berikut adalah salah satu contoh cara kerja algoritma tersebut:

P\T	C	C	C	A	G	T	A	T	A	G	A	T	T	T
A	0	0	0	1	0	0	1	0	1	0	1	0	0	0
T	0	0	0	0	0	1	0	2	0	0	0	2	1	1
G	0	0	0	0	1	0	0	0	0	1	0	0	0	0
C	1	1	1	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	1	0	0	0	0	1	0	0	0	0
A	0	0	0	1	0	0	1	0	1	0	2	0	0	0
T	0	0	0	0	0	1	0	2	0	0	0	3	1	1
A	0	0	0	1	0	0	2	0	3	0	1	0	0	0
G	0	0	0	0	2	0	0	0	0	4	0	0	0	0
A	0	0	0	1	0	0	1	0	1	0	5	0	0	0
G	0	0	0	0	2	0	0	0	0	2	0	0	0	0
T	0	0	0	0	0	3	0	1	0	0	0	1	1	1

Gambar 1. Tabel Algoritma LCCS

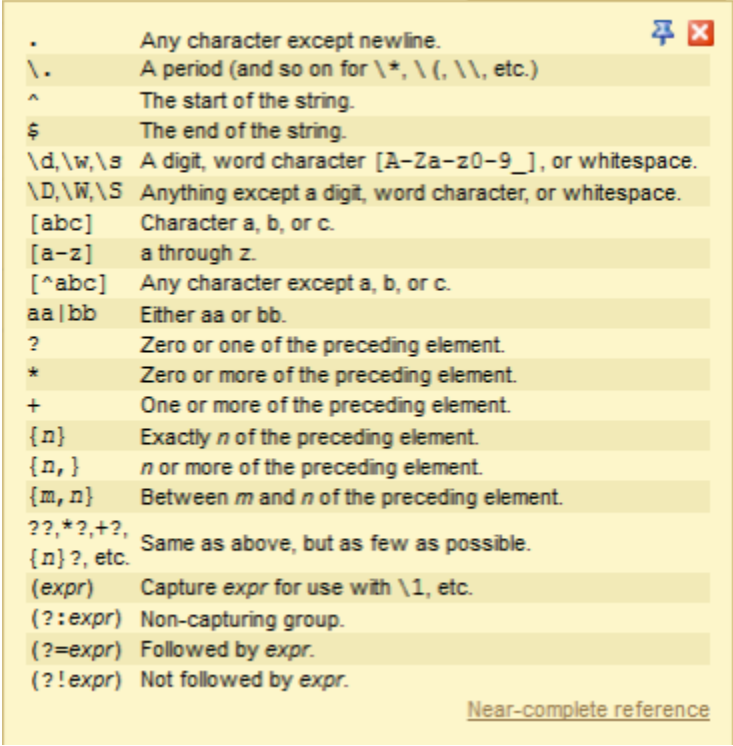
Cara pengisian tabel:

- Apabila karakter sama, maka isi sel tersebut dengan formula $1 + \text{isi dari sel sebelah kiri atas sel tersebut}$.
- Apabila karakter berbeda, maka isi sel tersebut dengan 0.

Hasil akhir dari algoritma tersebut adalah sel dengan nilai terbesar. Konstruksi *subsequence* yang dihasilkan dengan cara merunut-balik sel dengan nilai terbesar ke sel sebelah kiri atas sampai ditemukan sel dengan nilai 1. Pada contoh di atas, sel dengan nilai terbesar adalah 5 dan hasil konstruksi *subsequence*-nya adalah **ATAGA**.

4. Algoritma Regex

Berikut ini adalah notasi umum dari regex:



.	Any character except newline.
\.	A period (and so on for *, \ (, \\, etc.)
^	The start of the string.
\$	The end of the string.
\d,\w,\s	A digit, word character [A-Za-z0-9_], or whitespace.
\D,\W,\S	Anything except a digit, word character, or whitespace.
[abc]	Character a, b, or c.
[a-z]	a through z.
[^abc]	Any character except a, b, or c.
aa bb	Either aa or bb.
?	Zero or one of the preceding element.
*	Zero or more of the preceding element.
+	One or more of the preceding element.
{n}	Exactly n of the preceding element.
{n, }	n or more of the preceding element.
{m, n}	Between m and n of the preceding element.
??,*?,+?, {n}?, etc.	Same as above, but as few as possible.
(expr)	Capture expr for use with \1, etc.
(?:expr)	Non-capturing group.
(?=expr)	Followed by expr.
(?!expr)	Not followed by expr.

[Near-complete reference](#)

Gambar 15. Notasi Regular Expression

B. Penjelasan Aplikasi Web

Aplikasi *web* ini dibuat dengan bahasa Javascript. Untuk bagian *frontend* dari aplikasi *web* ini dibuat dengan menggunakan Next.js serta Chakra UI dan untuk bagian *backend*-nya dibuat dengan menggunakan Node.js. Kemudian *database* yang digunakan dalam aplikasi *web* ini adalah MongoDB. Aplikasi *web* ini juga di-*deploy* ke dalam *website* dengan bantuan Vercel.

BAB III

ANALISIS PEMECAHAN MASALAH

A. Langkah Penyelesaian Masalah Setiap Fitur

1. Input Penyakit

Pada fitur Input Penyakit, program hanya menambahkan penyakit baru beserta dengan DNA dari penyakit tersebut. Data terkait penyakit tersebut akan ditambahkan ke dalam *database* menggunakan MongoDB.

2. Prediksi Penyakit

Pada fitur Prediksi Penyakit, program melakukan prediksi apakah seorang pasien mengidap penyakit tertentu berdasarkan DNA yang dimilikinya. Program akan meminta input berupa nama pengguna, nama penyakit yang ingin dilakukan prediksi terhadap pengguna, input file berupa DNA dari pengguna, dan metode untuk melakukan string matching, pada program kami terdapat dua jenis metode yang dapat digunakan, yaitu Algoritma Knuth-Morris-Pratt (KMP atau Algoritma Boyer-Moore (BM).

Langkah penyelesaian untuk menentukan apakah pasien mengidap penyakit tertentu adalah

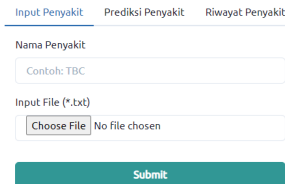
- 1) Program mengambil data DNA penyakit berdasarkan input nama penyakit.
- 2) Program melakukan algoritma String Matching berdasarkan metode yang dipilih oleh pengguna, algoritma String Matching mencari DNA penyakit pada DNA pasien, kemudian mengeluarkan tingkat kemiripan tertinggi yang dimiliki DNA pasien.
- 3) Program menampilkan hasil yang didapatkan dengan format “<Tanggal Pengecekan> - <Nama Pengguna> - <Nama Penyakit> - <Tingkat Kemiripan> - <Hasil Prediksi>”
- 4) Hasil prediksi akan bernilai true apabila tingkat kemiripan lebih atau sama dengan 80%
- 5) Program menyimpan hasil yang didapatkan ke dalam database.

3. Riwayat Penyakit

Pada fitur Riwayat Penyakit, program melakukan query data dari database berdasarkan input yang diberi pengguna. Pengguna dapat melakukan pencarian menggunakan tanggal dan nama penyakit, hanya tanggal, dan hanya nama penyakit. Dalam penentuan query data digunakan algoritma Regular Expression (Regex).

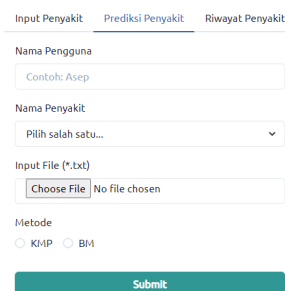
B. Fitur Fungsional Dan Arsitektur Aplikasi Web

Pada aplikasi web kami, ketiga fitur diatas tersedia pada laman utama, dan dibedakan berdasarkan tab yang dipilih. Terdapat tiga tab, yaitu Input Penyakit, Prediksi Penyakit, dan Riwayat Penyakit. Input penyakit digunakan untuk menambahkan data terkait penyakit. Prediksi penyakit digunakan untuk melakukan prediksi suatu penyakit terhadap pengguna. Riwayat penyakit untuk melakukan pencarian riwayat penyakit yang pernah dilakukan sebelumnya.



29 April 2022
Dibuat oleh:
13520025 - M. Akmal Arifin
13520067 - Farnas Rozaan I.
13520098 - Andika Naufal Hilmy
Powered by  Vercel

Gambar 16. Tampilan Aplikasi Web Tab Input Penyakit



29 April 2022
Dibuat oleh:
13520025 - M. Akmal Arifin
13520067 - Farnas Rozaan I.
13520098 - Andika Naufal Hilmy
Powered by  Vercel

Gambar 17. Tampilan Aplikasi Web Tab Prediksi Penyakit

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Kata Kunci

Ketikkan tanggal dan/atau penyakit...

Submit

29 April 2022
Dibuat oleh:
13520025 - M. Akmal Arifin
13520067 - Farnas Rozaan I.
13520098 - Andika Naufal Hilmy
Powered by  **Vercel**

Gambar 18. Tampilan Aplikasi Web Tab Riwayat Penyakit

BAB IV

IMPLEMENTASI DAN PENGUJIAN

A. Spesifikasi Teknis Program

Pada program ini terdapat beberapa fungsi dan prosedur yang dibangun untuk menjalankan program. Fungsi dan prosedur tersebut adalah:

1. **checkDNA**

Fungsi ini melakukan pengecekan apakah file input yang dimasukan merupakan rangkaian DNA atau bukan. Fungsi ini digunakan ketika melakukan input file oleh pengguna pada fitur Input Penyakit dan Prediksi Penyakit

2. **dateToString**

Fungsi ini mengubah format tanggal yang awalnya berupa angka menjadi tulisan dengan format "<Tanggal> <Bulan> <Tahun>". Fungsi ini digunakan pada fitur riwayat penyakit, ketika ingin menampilkan riwayat penyakit ke layar.

3. **dnaSimilarity**

Fungsi ini mengembalikan tingkat kesamaan kesamaan paling besar yang ada diantara dua string. Fungsi ini digunakan pada fungsi stringmatchingBM dan stringmatchingKMP.

4. **getFileContent**

Fungsi ini mengembalikan isi dari suatu file. Fungsi ini digunakan pada fitur Input Penyakit dan Prediksi Penyakit.

5. **getFileExtension**

Fungsi ini mengembalikan extension dari suatu file. Fungsi ini digunakan pada fitur Input Penyakit dan Prediksi Penyakit.

6. **searchTermsRegex**

Fungsi ini mengembalikan status, dateBegin, dateEnd, disease dalam bentuk JSON yang nantinya akan digunakan dalam proses query pada API. Fungsi ini digunakan pada fitur Riwayat Penyakit.

7. **stringmatchingBM**

Prosedur ini merupakan algoritma string matching menggunakan Algoritma Boyer-Moore (BM). Prosedur ini digunakan pada fitur Prediksi Penyakit.

8. stringmatchingKMP

Prosedur ini merupakan algoritma string matching menggunakan Algoritma Knuth-Morris-Pratt (KMP). Prosedur ini digunakan pada fitur Prediksi Penyakit.

9. stringSanitizer

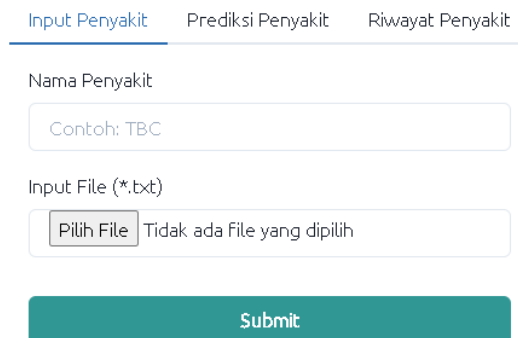
Fungsi ini mengembalikan string yang telah dilakukan pembersihan format. Fungsi ini menghilangkan newline, spaces, dan tabs pada string. Fungsi ini digunakan pada saat input file, untuk mengembalikan isi file yang telah terformat dengan baik.

B. Tata Cara Penggunaan Program

Terdapat 3 fitur yang terdapat dalam program ini, yaitu fitur input penyakit, prediksi penyakit, dan riwayat prediksi penyakit. Berikut ini adalah *interface* dari masing-masing fitur serta cara penggunaannya:

1. Input Penyakit

Interface:



Gambar 19. Interface Input Penyakit

Cara penggunaan:

- i. Masukkan nama penyakit yang hendak ditambahkan ke *database*. Apabila nama penyakit yang hendak ditambahkan ke *database* ternyata sudah ada dalam *database* maka akan terjadi *error*.

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Nama Penyakit

TBC

Input File (*.txt)

Pilih File dna.txt

Submit

Error E11000 duplicate key error collection:
myFirstDatabase.diseases index: name_1 dup
key: { name: "TBC" }

Gambar 20. Error Nama Penyakit Duplikat

- ii. Unggah *file sequence* DNA penyakit yang hendak ditambahkan ke *database*. Pastikan *sequence* DNA valid, yaitu tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, dan tidak ada spasi. Jika *sequence* DNA tidak valid, misalnya adalah sebagai berikut: ACCTGGATGCCAGCTTCACGTCCAATGCATt aD, maka akan terjadi *error*.

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Nama Penyakit

Sinus

Input File (*.txt)

Pilih File dna.txt

Submit

Error

Gambar 21. Error Invalid DNA

- iii. Tekan “Submit” untuk menambahkan data penyakit ke *database*.

2. Prediksi Penyakit

Interface:

Input Penyakit **Prediksi Penyakit** Riwayat Penyakit

Nama Pengguna
Contoh: Asep

Nama Penyakit
Pilih salah satu...

Input File (*.txt)
Pilih File Tidak ada file yang dipilih

Metode
☐ KMP ☐ BM

Submit

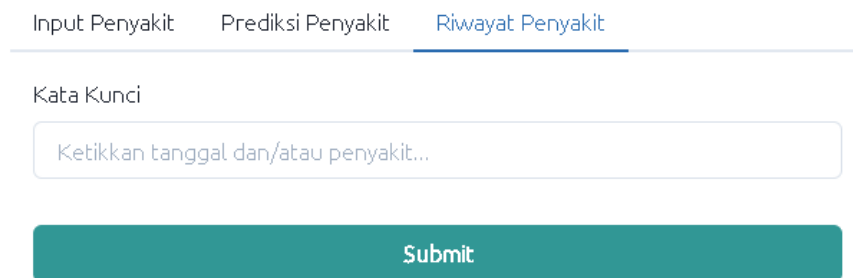
Gambar 22. Interface Prediksi Penyakit

Cara penggunaan:

- Masukkan nama pengguna yang hendak diprediksi apakah pengguna tersebut mengidap penyakit tertentu.
- Pilih nama penyakit untuk dicek apakah pengguna tadi mengidap penyakit yang dipilih.
- Unggah *file sequence* DNA pengguna yang hendak diprediksi. Asumsi panjang *sequence* DNA pengguna > panjang *sequence* DNA penyakit, sehingga di bagian ini *error* tersebut tidak ditangani. Pastikan *sequence* DNA valid, yaitu tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, dan tidak ada spasi. Jika *sequence* DNA tidak valid, maka akan terjadi *error* sama seperti di fitur Input Penyakit.
- Pilih metode *string matching* yang hendak digunakan, KMP atau BM.
- Tekan “Submit” untuk memprediksi apakah pengguna mengidap penyakit yang sudah dipilih tadi. Hasil prediksi juga akan ditambahkan ke *database*.

3. Riwayat Penyakit

Interface:



Gambar 23. Interface Riwayat Penyakit

Cara penggunaan:

- i. Ketikkan kata kunci untuk mencari riwayat hasil prediksi penyakit. Format pencarian yang diterima pada kolom pencarian ini adalah kata kunci berupa tanggal beserta penyakit, hanya tanggal, atau hanya penyakit. Untuk format tanggal sendiri, program dapat menerima 3 jenis penulisan. Berikut adalah ketiga jenis penulisan tersebut:
 - a. dd <bulan> yyyy
Contoh: 7 April 2022, 29 Januari 2022
 - b. dd/mm/yyyy
Contoh: 7/4/2022, 29/01/2022
 - c. dd-mm-yyyy
Contoh: 7-4-2022, 29-01-2022
- ii. Tekan “Submit” untuk mencari riwayat prediksi penyakit sesuai kata kunci yang sudah diketikkan.

C. Hasil Pengujian

1. *Input* penyakit berhasil


Input Penyakit Prediksi Penyakit Riwayat Penyakit

Nama Penyakit

Input File (*.txt)

Pilih File ValidDiseaseDNA3.txt

Submit


Success Penyakit berhasil ditambahkan

2. *Input* penyakit gagal
Duplikat nama penyakit:


Input Penyakit Prediksi Penyakit Riwayat Penyakit

Nama Penyakit

Input File (*.txt)

Pilih File ValidDiseaseDNA1.txt

Submit


Error E11000 duplicate key error collection:
myFirstDatabase.diseases index: name_1 dup
key: { name: "Alzheimer" }

Invalid *sequence* DNA:


Input Penyakit Prediksi Penyakit Riwayat Penyakit

Nama Penyakit

Input File (*.txt)

Pilih File InvalidDiseaseDNA4.txt

Submit


Error Invalid DNA sequence

3. Prediksi penyakit berhasil dengan hasil True

Input Penyakit **Prediksi Penyakit** Riwayat Penyakit

Nama Pengguna

Endah

Nama Penyakit

Alzheimer


Input File (*.txt)

Pilih File ValidTestDNA3.txt

Metode

☒ KMP ☐ BM

Submit

Berhasil menambahkan data: 29 April 2022 -
 **Success** Endah - Alzheimer - 100.00% - true, Waktu eksekusi: 1 ms

4. Prediksi penyakit berhasil dengan hasil False

Input Penyakit **Prediksi Penyakit** Riwayat Penyakit

Nama Pengguna

Endah

Nama Penyakit

TBC


Input File (*.txt)

Pilih File ValidTestDNA3.txt

Metode

☐ KMP ☒ BM

Submit

Berhasil menambahkan data: 29 April 2022 -
 **Success** Endah - TBC - 13.16% - false, Waktu eksekusi: 23 ms

5. Prediksi penyakit gagal

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Nama Pengguna

Ucup

Nama Penyakit

TBC


Input File (*.txt)

Pilih File InvalidTestDNA4.txt

Metode

☒ KMP ☐ BM

Submit

 **Error** Invalid DNA sequence


6. Riwayat penyakit menggunakan format tanggal + penyakit

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Kata Kunci

29 April 2022 Alzheimer

Submit

 **Success** Berhasil mengambil data

NO	TANGGAL	NAMA PASIEN	NAMA PENYAKIT	KEMIRIPAN	HASIL
1	29 April 2022	Endah	Alzheimer	100.00%	True

7. Riwayat penyakit menggunakan format tanggal

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Kata Kunci

Submit

✓ **Success** Berhasil mengambil data

NO	TANGGAL	NAMA PASIEN	NAMA PENYAKIT	KEMIRIPAN	HASIL
1	29 April 2022	aaaaaa	COVID-19	34.62%	False
2	29 April 2022	asede	COVID-19	34.62%	False
3	29 April 2022	Endah	Alzheimer	100.00%	True
4	29 April 2022	Endah	TBC	13.16%	False

8. Riwayat penyakit menggunakan format penyakit

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Kata Kunci

Submit

✓ **Success** Berhasil mengambil data

NO	TANGGAL	NAMA PASIEN	NAMA PENYAKIT	KEMIRIPAN	HASIL
1	29 April 2022	Endah	TBC	13.16%	False


9. Riwayat penyakit menggunakan format yang salah Penyakit + tanggal:

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Kata Kunci

TBC 29/04/2022

Submit

 **Error** Kata kunci pencarian Anda kurang tepat


Format tanggal salah:

Input Penyakit Prediksi Penyakit Riwayat Penyakit

Kata Kunci

29 02 2022

Submit

 **Error** Kata kunci pencarian Anda kurang tepat

BAB V

KESIMPULAN, SARAN, DAN KOMENTAR/REFLEKSI

A. Kesimpulan

Algoritma Knuth-Morris-Pratt dan/atau Boyer-Moore adalah algoritma yang tepat untuk menyelesaikan permasalahan *string matching*. Kedua algoritma ini lebih cepat jika dibandingkan dengan Brute Force. Algoritma KMP tidak pernah bergerak mundur, hal ini menyebabkan algoritma ini lebih baik dalam memproses suatu string yang ukurannya besar. Akan tetapi, algoritma KMP tidak dapat bekerja dengan baik apabila alfabet yang muncul pada string bervariasi. Hal ini dikarenakan akan meningkatkan kemungkinan terjadinya *mismatch* ketika melakukan pencarian. Sebaliknya, algoritma BM akan bekerja dengan baik apabila string memiliki variasi alfabet yang banyak, namun bekerja kurang baik jika variasi alfabetnya sedikit dan ukurannya besar.

B. Saran

Aplikasi web yang kami buat masih bisa untuk dikembangkan lebih lanjut. Salah satu pengembangan fitur yang dapat ditambahkan antara lain yaitu fitur drag and drop file untuk memasukkan file DNA penyakit maupun pasien. Menambahkan lebih banyak fitur pada search riwayat penyakit. Menambahkan opsi untuk menampilkan banyaknya perbandingan yang dilakukan ketika melakukan prediksi penyakit pada pasien. Selain itu, dalam menentukan algoritma string matching, dapat ditentukan berdasarkan banyaknya variasi huruf yang muncul pada text. Apabila variasi yang muncul hanya sedikit, maka lebih baik menggunakan algoritma Knuth-Morris-Pratt. Sedangkan apabila variasi yang muncul banyak, lebih baik menggunakan algoritma Boyer-Moore.

C. Komentar/Refleksi

Secara garis besar, tugas besar 3 strategi algoritma ini sangat menantang, memaksa kami untuk mempelajari banyak hal dengan sendirinya. Mengeksplor banyak hal baru yang tidak diajari diperkuliahan. Meskipun kami berhasil menyelesaikan tugas besar ini, kami yakin masih banyak hal yang perlu dipelajari untuk dapat mengembangkan suatu web aplikasi dengan baik. Kode yang kami buat pun masih belum tertata dengan rapi. Kami berharap kedepannya kami bisa belajar dengan baik, dan suatu saat nanti, kami dapat mengembangkan suatu aplikasi yang dapat bermanfaat bagi banyak orang.

DAFTAR PUSTAKA

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>
https://www.academia.edu/12374486/String_Matching_Evaluation_Methods_for_DNA_Comparison
<https://www.mongodb.com/blog/post/quick-start-nodejs-mongodb-how-to-get-connected-to-your-database>

LAMPIRAN

Repository GitHub : https://github.com/dawetmaster/Tubes3_13520025
Link webapp : <https://tubes3-stima-five.vercel.app/>