

LAPORAN TUGAS BESAR 3

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana

Ditujukan untuk memenuhi salah satu tugas besar mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2022/2023



Disusun oleh:

Ahmad Ghulam Ilham 13521118

Ferindya Aulia Berlianty 13521161

Muhammad Habibi Husni 13521169

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023**

DAFTAR ISI

DAFTAR ISI

BAB I DESKRIPSI TUGAS

BAB II LANDASAN TEORI

BAB III ANALISIS PEMECAHAN MASALAH

BAB IV IMPLEMENTASI DAN PENGUJIAN

BAB V KESIMPULAN DAN SARAN

DAFTAR PUSTAKA

LAMPIRAN

BAB I

DESKRIPSI TUGAS

Dalam tugas besar ini, anda diminta untuk membangun sebuah aplikasi ChatGPT sederhana dengan mengaplikasikan pendekatan QA yang paling sederhana tersebut. Pencarian pertanyaan yang paling mirip dengan pertanyaan yang diberikan pengguna dilakukan dengan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Regex digunakan untuk menentukan format dari pertanyaan (akan dijelaskan lebih lanjut pada bagian fitur aplikasi). Jika tidak ada satupun pertanyaan pada database yang exact match dengan pertanyaan pengguna melalui algoritma KMP ataupun BM, maka gunakan pertanyaan termirip dengan kesamaan setidaknya 90%. Apabila tidak ada pertanyaan yang kemiripannya di atas 90%, maka chatbot akan memberikan maksimum 3 pilihan pertanyaan yang paling mirip untuk dipilih oleh pengguna.

Perhitungan tingkat kemiripan dibebaskan kepada anda asalkan dijelaskan di laporan, namun disarankan menggunakan salah satu dari algoritma *Hamming Distance*, *Levenshtein Distance*, ataupun *Longest Common Subsequence*.

Fitur-Fitur Aplikasi:

ChatGPT sederhana yang anda membuat wajib dapat melakukan beberapa fitur / klasifikasi query seperti berikut:

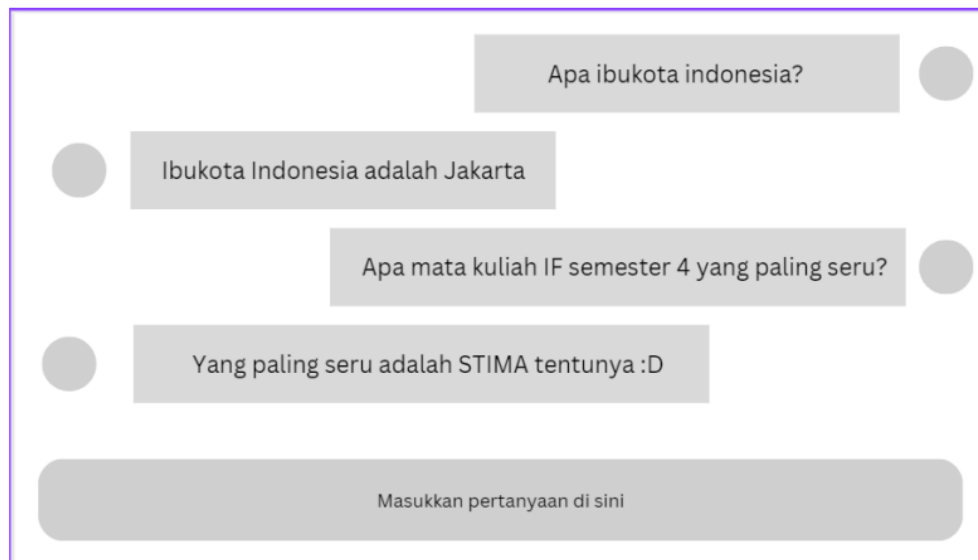
1. Fitur pertanyaan teks (didapat dari database)
Mencocokkan pertanyaan dari input pengguna ke pertanyaan di database menggunakan algoritma KMP atau BM.
2. Fitur Kalkulator
Pengguna memasukkan input query berupa persamaan matematika. Contohnya adalah $2*5$ atau $5+9*(2+4)$. Operasi cukup Tambah, kurang, kali, bagi, pangkat, kurung.
3. Fitur Tanggal
Pengguna memasukkan input berupa tanggal, lalu chatbot akan merespon dengan hari apa di tanggal tersebut. Contohnya adalah 25/08/2023 maka chatbot akan menjawab dengan hari senin.
4. Tambah pertanyaan dan jawaban ke database
Pengguna dapat menambahkan pertanyaan dan jawabannya sendiri ke database dengan query contoh “Tambahkan pertanyaan xxx dengan jawaban yyy”. Menggunakan algoritma

string matching untuk mencari tahu apakah pertanyaan sudah ada. Apabila sudah, maka jawaban akan diperbaharui.

5. Hapus pertanyaan dari database

Pengguna dapat menghapus sebuah pertanyaan dari database dengan query contoh “Hapus pertanyaan xxx”. Menggunakan string algoritma string matching untuk mencari pertanyaan xxx tersebut pada database.

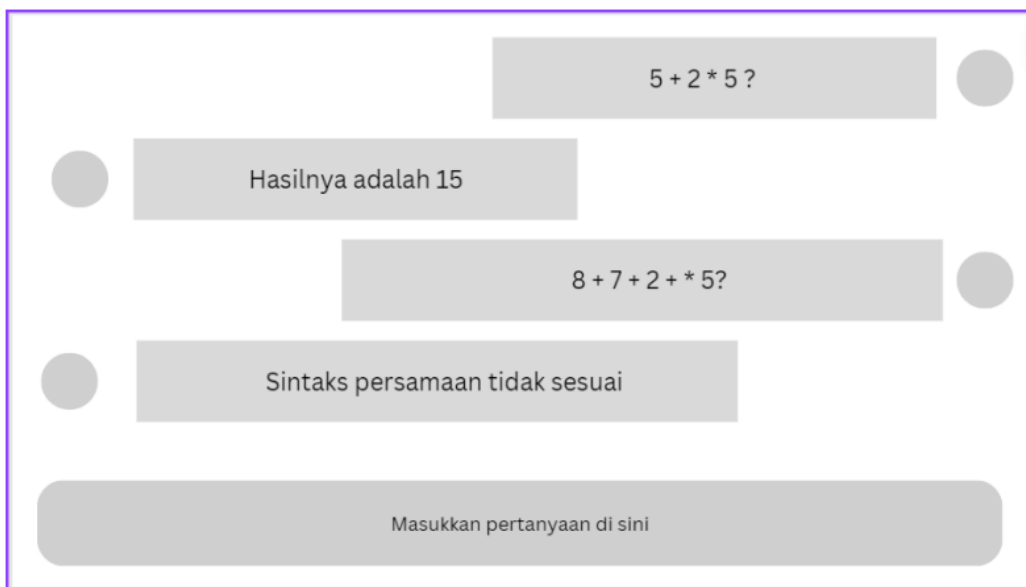
Klasifikasi dilakukan menggunakan regex dan terklasifikasi layaknya bahasa sehari-hari. Algoritma string matching KMP dan BM digunakan untuk klasifikasi query teks. Tersedia toggle untuk memilih algoritma KMP atau BM. Semua pemrosesan respons dilakukan pada sisi backend. Jika ada pertanyaan yang sesuai dengan fitur, maka tampilkan saja “Pertanyaan tidak dapat diproses”. Berikut adalah beberapa contoh ilustrasi sederhana untuk tiap pertanyaannya. (Note: Tidak wajib mengikuti ilustrasi ini, tampilan disamakan dengan chatGPT juga boleh)



Gambar 1.1 Ilustrasi Fitur Pertanyaan teks kasus exact



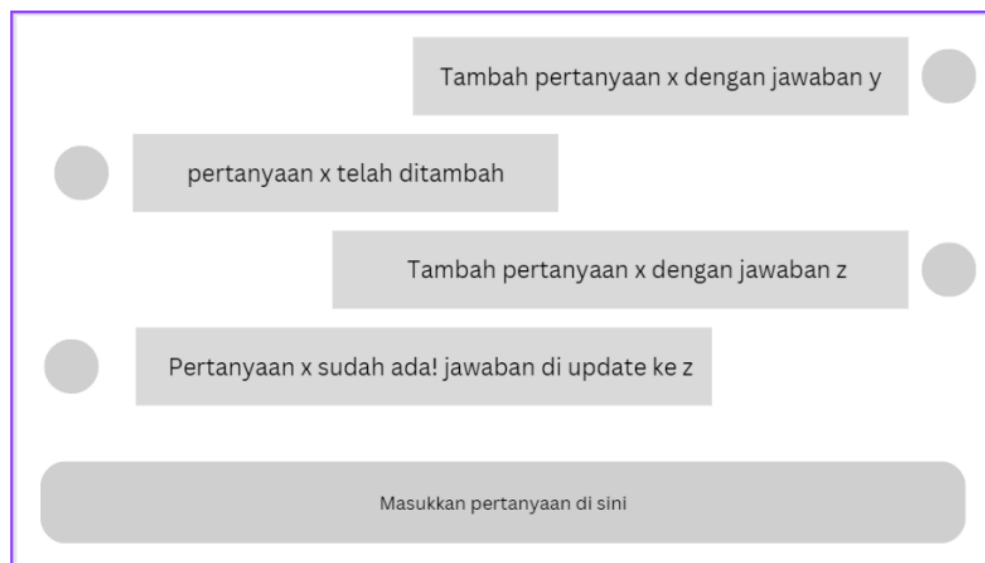
Gambar 1.2 Ilustrasi Fitur Pertanyaan teks kasus tidak exact



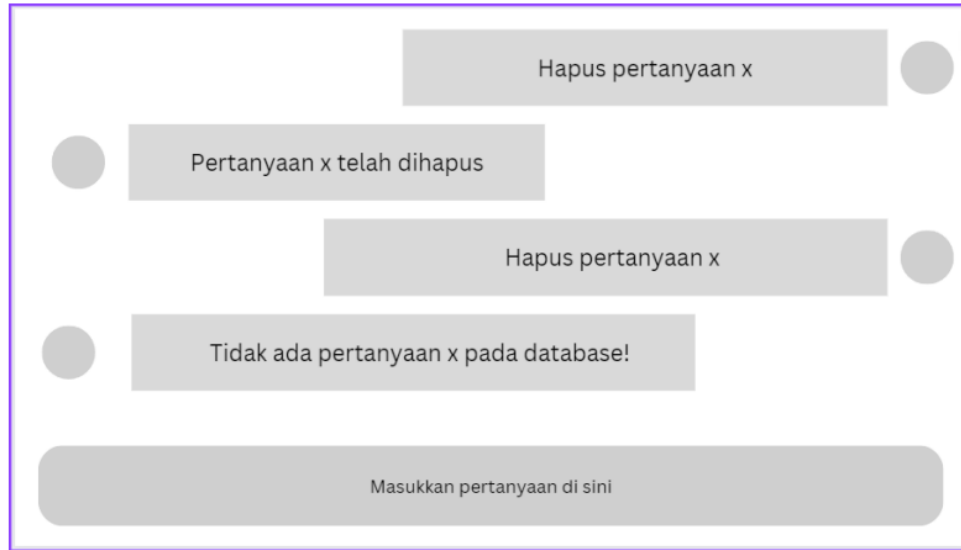
Gambar 1.3 Ilustrasi Fitur Kalkulator



Gambar 1.4 Ilustrasi Fitur Tanggal

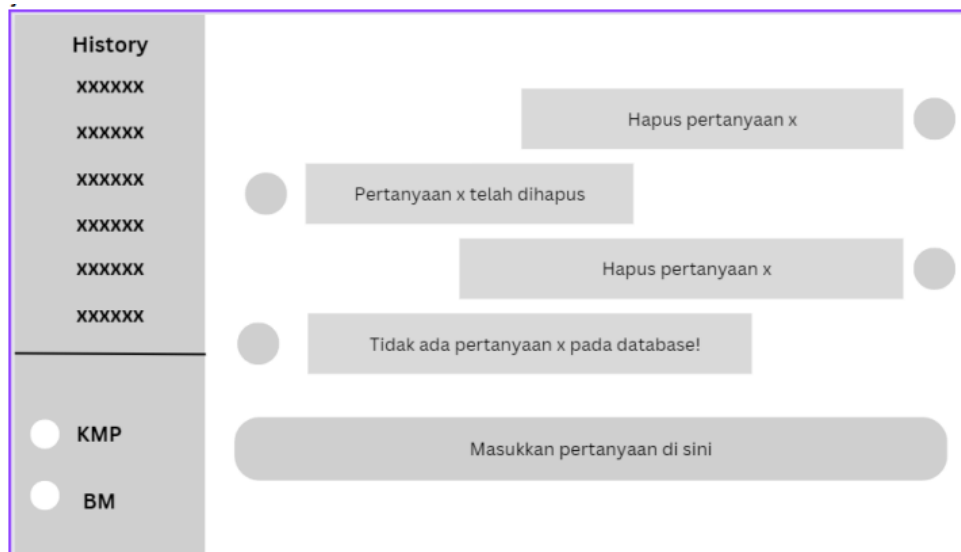


Gambar 1.5 Ilustrasi Fitur Tambah Pertanyaan



Gambar 1.6 Ilustrasi Fitur Hapus Pertanyaan

Layaknya ChatGPT, di sebelah kiri disediakan history dari hasil pertanyaan anda. Cukup tampilkan 5-10 pertanyaan terbaru di toolbar kiri. Perhatikan bahwa sistem history disini disamakan dengan chatGPT, sehingga satu history yang diklik menyimpan seluruh IF2211 Strategi Algoritma - Tugas Besar 3 7 pertanyaan pada sesi itu. Apabila history diclick, maka akan merestore seluruh pertanyaan dan jawaban di halaman utama. Contoh ilustrasi keseluruhan:



Gambar 1.7 Ilustrasi Keseluruhan

Spesifikasi Program:

1. Aplikasi berbasis website dengan pembagian Frontend dan Backend yang jelas.
2. Implementasi Backend **wajib** menggunakan Node.js / Golang, sedangkan Frontend dibebaskan tetapi **disarankan** untuk menggunakan React / Next.js / Vue / Angular. Lihat referensi untuk selengkapnya.
3. Penyimpanan data **wajib** menggunakan basis data (MySQL / PostgreSQL / MongoDB).
4. Algoritma pencocokan string (KMP dan Boyer-Moore) dan Regex wajib diimplementasikan pada sisi Backend aplikasi.
5. Informasi yang **wajib** disimpan pada basis data:
 - a. Tabel pasangan pertanyaan dan Jawaban
 - b. Tabel history
6. Skema basis data dibebaskan asalkan mencakup setidaknya kedua informasi di atas.
7. Proses string matching pada tugas ini **Tidak case sensitive**.
8. Pencocokan yang dilakukan adalah dalam satu kesatuan string pertanyaan utuh (misal “Apa ibukota Filipina?”), bukan kata per kata (“apa”, “ibukota”, “Filipina”).

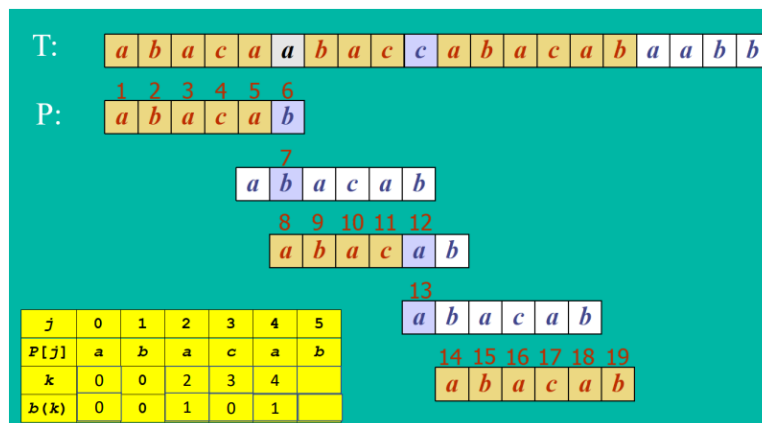
BAB II

LANDASAN TEORI

2.1 Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan string yang ditemukan oleh D.E. Knuth bersama J.H. Morris dan V.R. Pratt. Algoritma ini merupakan pengembangan dari algoritma pencarian string dengan pendekatan secara Brute Force. Algoritma Knuth-Morris-Pratt (KMP) akan melakukan pencarian pola dalam sebuah string secara urut dari kiri ke kanan seperti algoritma Brute Force, namun memiliki pola pencarian yang lebih optimal sehingga meningkatkan efisiensi dari pencarian dengan mengurangi banyaknya perbandingan huruf yang diperiksa.

Algoritma ini memanfaatkan pencocokan prefix dan suffix dari sebuah string. Apabila saat pencarian ditemukan ketidakcocokan sebuah huruf antara string S pada indeks ke i dengan pola P pada indeks ke j , maka indeks i pada S dapat digeser sebanyak prefix terbesar dari $P[0..j-1]$ yang juga merupakan suffix dari $P[1..j-1]$. Hal ini bertujuan untuk mengurangi pemborosan pengecekan dengan menghindari memeriksa huruf yang tidak perlu. Berikut adalah ilustrasinya:



Gambar 2.1.1 Ilustrasi Pergeseran Prefix dengan algoritma KMP

Agar memudahkan untuk mengetahui prefix terbesar yang cocok dengan suffix apabila ditemukan ketidakcocokan huruf pada suatu indeks x , algoritma KMP menggunakan suatu border function atau dengan nama lain failure function yang menyimpan nilai dari panjang prefix terbesar yang sesuai. Hal tersebut bertujuan agar algoritma KMP dapat melakukan pergeseran

berdasarkan informasi yang disimpan sehingga membuat waktu pencarian panjang prefix terbesar menurun drastis. Berikut adalah contoh failure function untuk ilustrasi sebelumnya:

x	0	1	2	3	4	5
$P[x]$	a	b	a	c	a	b
$f(x)$	0	0	1	0	1	2

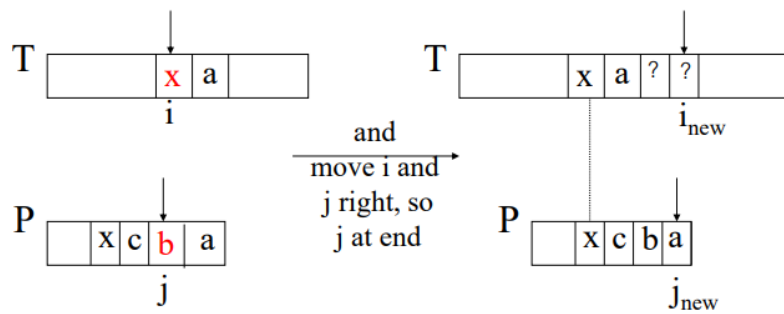
Gambar 2.1.2 Contoh failure function untuk Ilustrasi Sebelumnya

2.2 Algoritma Boyer-Moore (BM)

Algoritma Boyer-Moore (BM) merupakan algoritma pencocokan string yang ditemukan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1977. Berbeda dengan algoritma Knuth-Morris-Pratt (KMP) dan Brute Force yang melakukan pencocokan *pattern* dari kiri ke kanan, algoritma Boyer-Moore melakukan pencocokan *pattern* dari kanan ke kiri. Hal ini dilakukan agar lebih banyak informasi yang didapatkan jika dimulai dari kanan. Pemeriksaan terhadap suatu string S dimulai dari awal, namun terhadap suatu *pattern* P dimulai dari indeks terakhirnya yaitu panjang *pattern* - 1.

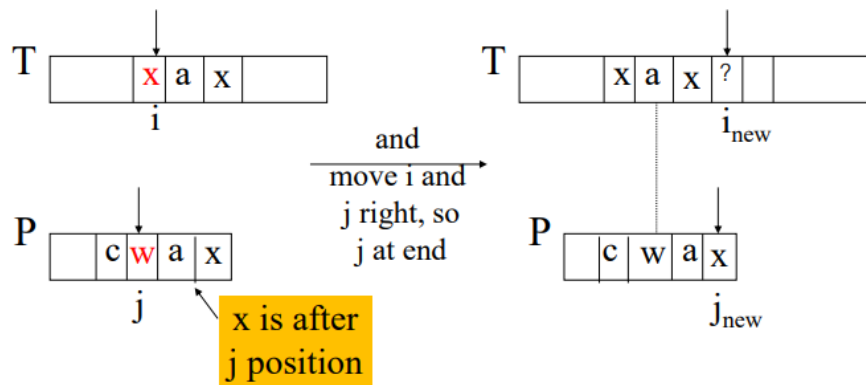
Terdapat dua teknik yang digunakan dalam algoritma Boyer-Moore yaitu *the looking-glass technique* dan *character-jump technique*. *The looking-glass technique* diartikan jika kita memeriksa suatu *pattern* P dalam string S dengan bergerak mundur terhadap P , dimulai dari akhir. Apabila huruf yang diperiksa cocok, maka akan lanjut bergeser ke kiri hingga semua huruf sudah diperiksa cocok atau bisa juga ditemukan huruf yang tidak cocok. Apabila huruf yang diperiksa tidak cocok, akan memanfaatkan *character-jump technique*. *Character-jump technique* merupakan teknik pencarian string dengan melakukan pergeseran indeks ke- i pada S apabila ditemukan huruf yang tidak cocok dan akan dikembalikan ke indeks terakhir pada P . Terdapat tiga kasus yang sering ditemui dalam pergeseran indeks i untuk ketidakcocokan huruf x di S yaitu:

- Pattern* P mengandung x , perlu menggeser *pattern* P agar kemunculan huruf x terakhir pada *pattern* P sejajar dengan $S[i]$. Dengan kata lain, mengembalikan j menjadi panjang *pattern* P - 1 dan menambah i hingga sejajar dengan kemunculan huruf x terakhir pada *pattern* P .



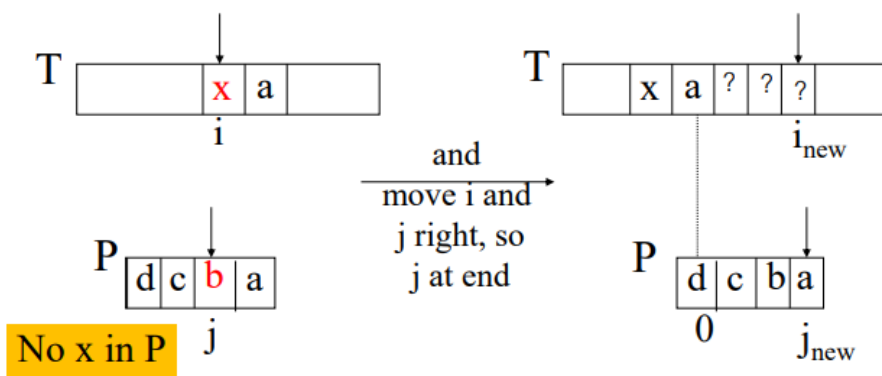
Gambar 2.2.1 Ilustrasi pergeseran pada Kasus 1

- b. *Pattern P* mengandung *x* tetapi tidak memungkinkan untuk menggeser *pattern P* hingga kemunculan huruf *x* terakhir dalam *P* sejajar dengan *x* dalam *S*. Dengan demikian, kita menggeser *P* sebanyak 1 karakter di kanan $S[i + 1]$.



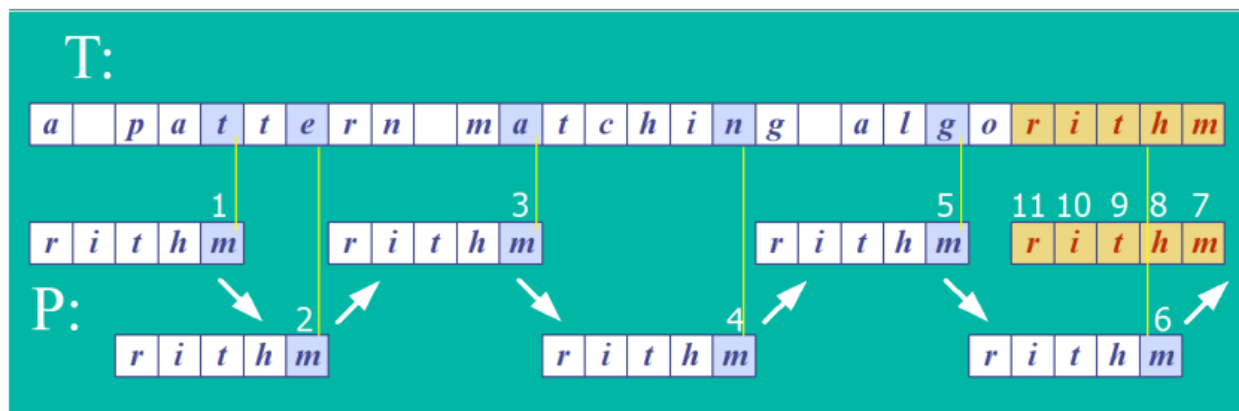
Gambar 2.2.2 Ilustrasi pergeseran pada Kasus 2

- c. Apabila terjadi kasus selain kasus diatas, maka perlu menggeser *pattern P* agar $P[0]$ sejajar dengan $T[i + 1]$.



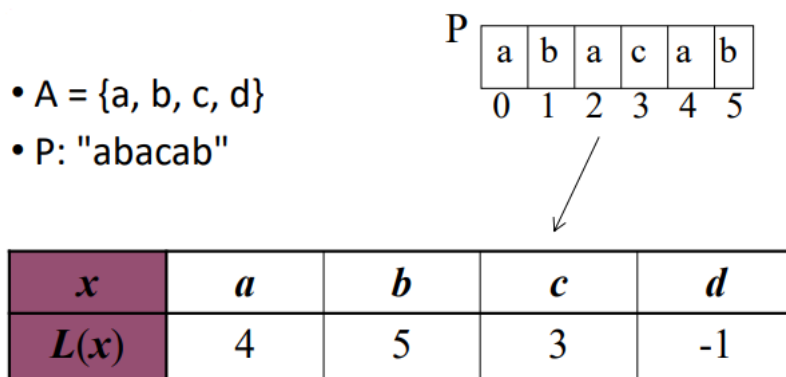
Gambar 2.2.3 Ilustrasi pergeseran selain kasus 1 dan 2

Di bawah ini merupakan ilustrasi pergeseran menggunakan algoritma Boyer-Moore:



Gambar 2.2.4 Ilustrasi pergeseran dengan algoritma Boyer-Moore

Algoritma Boyer-Moore memanfaatkan sebuah *Last Occurrence Function* yang menyimpan indeks terakhir kemunculan huruf x dalam *pattern* P . Hal ini bertujuan untuk mempermudah mengetahui kemunculan terakhir x dalam sebuah *pattern* P serta agar algoritma Boyer-Moore dapat melakukan pergeseran berdasarkan informasi yang disimpan sehingga membuat waktu pencarian indeks kemunculan terakhir menurun. Di bawah ini merupakan contoh Last Occurrence Function untuk ilustrasi sebelumnya.



Gambar 2.2.5 Ilustrasi Contoh Last Occurrence Function

2.3 Regular Expression (Regex)

Regular Expression (Regex) merupakan sebuah konsep yang ditemukan oleh seorang ilmuwan matematika bernama Stephen Cole Kleene. Regular Expression adalah sebuah teks atau string yang mendefinisikan pola pencarian agar dapat melakukan pencocokan string (*string*

matching), pencarian string (*string locate*), dan manipulasi string. Konsep ini kemudian digunakan dalam beberapa bahasa pemrograman menjadi sebuah library pemroses string.

Pencocokan string pada Regex berbeda dengan pencocokan string pada algoritma Brute Force, KMP, dan BM yang melakukan exact matching. Regex melakukan pencocokan string berdasarkan notasi dan *pattern* yang dimilikinya. Perlu diperhatikan bahwa *pattern* dan notasi dalam Regex termasuk *case sensitive*. Tanda yang terdapat pada Regex diklasifikasikan menjadi berikut:

a. Wildcard

Merupakan tanda titik “.” untuk menandakan cocok dengan huruf apapun selain newline.

b. Optionality

Merupakan tanda tanya “?” untuk menandakan bahwa regular expression yang diberikan sebelum tanda tersebut bersifat optional.

c. Repeatability

Merupakan tanda plus “+” untuk menandakan bahwa regular expression dapat diulang lebih dari sekali namun minimal diulang sekali serta tanda asterisk “*” untuk menandakan boleh diulang namun boleh tidak disertakan sama sekali.

d. Choice

Merupakan tanda kurung siku “[]” yang menandakan pola pattern yang cocok terbatas berdasarkan regular expression yang diberikan dalam choice tersebut.

e. Range

Merupakan tanda kurang “-” yang menunjukkan range, misalkan A-Z yang berarti menandakan huruf A kapital hingga Z kapital.

f. Complementation

Merupakan tanda panah ke arah atas “^” yang menandakan kebalikan atau negasi sehingga tanda yang berada setelah tanda ^ tidak boleh dipakai.

g. Common Special Symbol

Merupakan tanda “^” dan “\$” digunakan untuk mencocokkan awalan dan akhiran dari sebuah baris di dalam file. Tanda “^” memiliki dua arti. Bila tanda tersebut digunakan menjadi sebuah awalan pada class character maka tanda tersebut berarti negasi, selain itu tanda tersebut berarti menandakan awal dari sebuah baris.

h. Alternation

Merupakan tanda garis tegak “ | “ yang berfungsi seperti if-else,

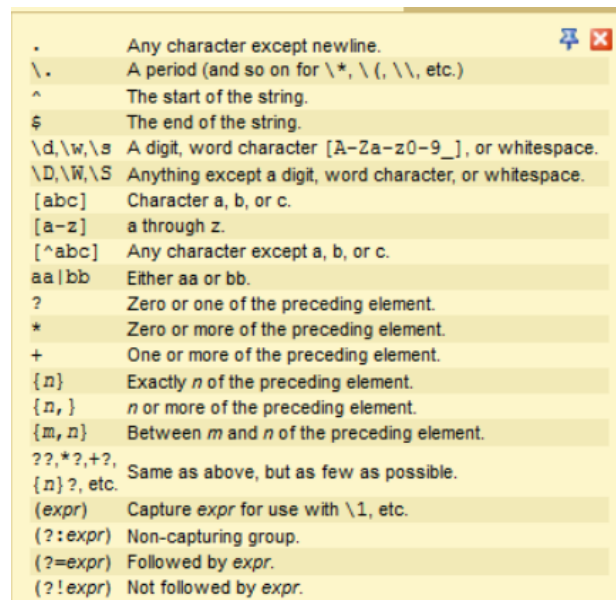
i. Token

Merupakan beberapa command khusus yang memiliki makna tertentu. Di bawah ini merupakan daftar token yang ada dalam regular expression:

Special Sequences

\b	Word boundary (zero width)
\d	Any decimal digit (equivalent to [0-9])
\D	Any non-digit character (equivalent to [^0-9])
\s	Any whitespace character (equivalent to [\t\n\r\f\v])
\S	Any non-whitespace character (equivalent to [^ \t\n\r\f\v])
\w	Any alphanumeric character (equivalent to [a-zA-Z0-9_])
\W	Any non-alphanumeric character (equivalent to [^ a-zA-Z0-9_])

Dengan demikian, berbagai tanda yang telah dijelaskan diatas dapat digunakan dalam regex untuk mendapatkan notasi-notasi umum dengan contoh sebagai berikut :

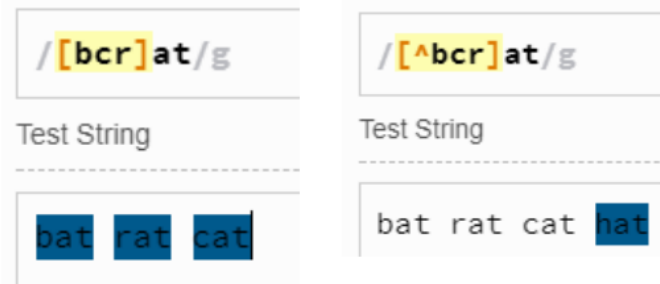


.	Any character except newline.
\.	A period (and so on for *, \ (, \ \, etc.)
^	The start of the string.
\$	The end of the string.
\d,\w,\s	A digit, word character [A-Za-z0-9_], or whitespace.
\D,\W,\S	Anything except a digit, word character, or whitespace.
[abc]	Character a, b, or c.
[a-z]	a through 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.

Gambar 2.2.6 Notasi umum dalam regex

Notasi regex tersebut dapat dimanfaatkan dalam bahasa pemrograman untuk melakukan pengecekan string. Biasanya bahasa pemrograman memiliki library masing-masing untuk pengecekan regex. Fungsi pencocokan string akan menerima parameter string yang akan

diperiksa serta pattern yang akan diperiksa dalam string tersebut serta mengembalikan boolean berupa True or False berdasarkan ada atau tidaknya pattern. Atau bisa juga untuk mengembalikan indeks saat ditemukan pattern tersebut, bergantung fungsi yang digunakan. Di bawah ini merupakan contoh pencocokan string menggunakan pattern dalam regex:



Gambar 2.2.7 Contoh pertama pencocokan string menggunakan regex



Gambar 2.2.8 Contoh kedua pencocokan string menggunakan regex

2.4 Penjelasan Singkat mengenai Aplikasi Web yang Dibangun

Aplikasi web yang dibangun terdiri atas bagian *front-end*, *back-end*, dan basis data. Pada sisi back-end, bahasa pemrograman yang kami gunakan adalah JavaScript dengan framework Express. Kami menggunakan JavaScript dikarenakan kemudahannya dalam menginisialisasi sebuah server. Data kami disimpan menggunakan database MongoDB. Pada sisi back-end terdapat juga algoritma yang digunakan untuk pencocokan string yaitu algoritma Knuth-Morris-Pratt (KMP) dan algoritma Boyer-Moore (BM).

Sementara itu, pada sisi front-end, kami menggunakan bahasa pemrograman JavaScript dengan framework React untuk mempermudah proses penggunaan komponen. Tampilan aplikasi web yang kami tulis pada sisi front-end, terdiri dari halaman yaitu halaman untuk

(Disini nanti ditaruh SS tampilan web)

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah Penyelesaian Masalah Setiap Fitur

a. Fitur pertanyaan teks (didapat dari database)

Pada tampilan aplikasi web, pengguna dapat mengakses halaman utama yang berguna untuk mencocokkan pertanyaan dari input pengguna ke pertanyaan di database menggunakan algoritma KMP atau BM. Langkah penyelesaian pencocokan string dengan algoritma KMP adalah sebagai berikut:

1. Buat fungsi `computeBorder(pattern)` yang menghitung nilai border (fail) pada setiap indeks dari `pattern`. Border atau fail adalah panjang maksimum suffix yang juga merupakan prefix dari sub-string `pattern[0..i]`. Algoritma `computeBorder(pattern)` menggunakan teknik pembuatan border dengan memanfaatkan informasi border pada karakter sebelumnya.
2. Buat fungsi `kmpMatch(text, pattern)` yang melakukan pencocokan `pattern` pada teks `text` menggunakan nilai border yang sudah dihitung sebelumnya. Algoritma `kmpMatch(text, pattern)` menggeser indeks `i` pada teks `text` dan indeks `j` pada `pattern`.
3. Panggil fungsi `kmpMatch(text, pattern)` untuk mencari kemunculan pertama `pattern` pada teks `text`. Jika ditemukan, kembalikan indeks awal kemunculan `pattern` pada teks `text`. Jika tidak ditemukan, mengembalikan nilai -1.

Sedangkan langkah penyelesaian pencocokan string dengan algoritma BM adalah sebagai berikut:

1. Buat fungsi `buildLast(pattern)` untuk membangun tabel `last`, yang berisi indeks terakhir dari setiap karakter dalam pola. Fungsi ini digunakan untuk membantu menghitung pergeseran dalam algoritma BM.
2. Buat fungsi `bmMatch(text, pattern)` yang mengambil teks dan pola sebagai argumen. Fungsi ini akan mengembalikan indeks pertama dari pola dalam teks, atau -1 jika tidak ditemukan.
3. Hitung panjang teks dan pola dengan variabel `n` dan `m`.
4. Buat tabel `last` menggunakan fungsi `buildLast(pattern)`.

5. Tentukan indeks awal untuk pencocokan dengan mengatur i dan j sama dengan $m-1$. Variabel i akan menunjukkan indeks saat ini dalam teks, dan j akan menunjukkan indeks saat ini dalam pola.
6. Jika indeks awal i lebih besar dari $n-1$, maka pola tidak dapat ditemukan dalam teks. Kembalikan -1.
7. Jika tidak, lakukan loop:
 - a. Jika karakter saat ini di teks (`text.charAt(i)`) cocok dengan karakter saat ini di pola (`pattern.charAt(j)`), maka periksa karakter berikutnya di teks dan pola dengan menurunkan i dan j sebanyak satu.
 - b. Jika karakter saat ini di teks tidak cocok dengan karakter saat ini di pola, maka tentukan pergeseran berikutnya menggunakan tabel last. Temukan indeks terakhir karakter saat ini di teks dalam tabel last (`last[text.charAt(i)]`). Jika karakter ini tidak ada dalam pola, maka pergeseran sejauh m . Jika karakter ini ada dalam pola, maka pergeseran sejauh $m - \text{Math.min}(j, 1 + \text{lo})$. Setelah menghitung pergeseran, atur i dan j ke posisi awal sebelum pergeseran.
8. Ulangi loop selama i kurang dari atau sama dengan $n-1$.
9. Jika pola tidak ditemukan, kembalikan -1.

Pada tampilan aplikasi web, terdapat kolom chat yang dapat diisi pertanyaan oleh pengguna. Aplikasi kemudian akan melakukan validasi terlebih dahulu apakah input pertanyaan sudah benar. Validasi dilakukan dengan menggunakan Regex dan akan bernilai benar apabila pola Regex pertanyaan sama seperti pertanyaan yang terdapat pada database. Apabila pertanyaan valid, maka program akan menampilkan jawaban yang sesuai dengan pertanyaan dari pengguna dan menampilkan history chat pada tampilan web. Apabila pola Regex pertanyaan tidak valid, maka akan menampilkan pesan “Pertanyaan tidak terdapat dalam database”.

b. Fitur Kalkulator

Pada tampilan aplikasi web, pengguna dapat mengakses halaman utama dapat menerima berbagai input pertanyaan oleh pengguna. Terdapat kolom chat yang dapat diisi pertanyaan oleh pengguna. Pengguna juga dapat memasukkan input query berupa persamaan matematika. Operasi matematika yang dapat dilakukan yaitu operasi tambah, kurang, kali, bagi, pangkat, dan kurung. Aplikasi kemudian akan melakukan validasi terlebih dahulu apakah input query

persamaan matematika sudah benar. Validasi dilakukan dengan menggunakan Regex dan akan akan bernilai benar apabila pola Regex persamaan matematika sama seperti persamaan matematika yang terdapat pada database. Sebagai contoh, jika pengguna memasukkan query berupa $2*5$ maka chatbot akan menjawab dengan 10.

c. Fitur Tanggal

Pada tampilan aplikasi web, terdapat kolom chat yang dapat diisi pertanyaan oleh pengguna. Fitur ini memanfaatkan Regex dalam melakukan pencocokan string. Pengguna dapat memasukkan berbagai input query salah satunya mengenai tanggal. Sebagai contohnya, apabila pengguna memasukkan input query 04/05/2023 hari apa, maka aplikasi akan menjawab dengan hari Kamis.

d. Tambah pertanyaan dan jawaban ke database

Pengguna juga dapat menambahkan pertanyaan dan jawabannya sendiri ke database dengan query tertentu. Proses ini menggunakan algoritma pencocokan string yang telah dibuat yaitu meliputi algoritma KMP dan BM untuk mencari tahu apakah pertanyaan sudah ada dalam database. Apabila sudah, maka jawaban akan diperbaharui. Fitur ini memanfaatkan algoritma KMP dan BM yang terdapat dalam file KMPSearch.js dan BMSearch.js. Masing-masing file ini melakukan extends terhadap getLevenDist. getLevenDist disini merupakan function dari Levenshtein Distance yang digunakan untuk membandingkan seberapa cocok string yang terdapat pada input dan string yang terdapat pada database. Fungsi getLevenDist mengembalikan jarak Levenshtein Distance yaitu jumlah minimum operasi yang diperlukan untuk mengubah string s1 menjadi s2, dimana semakin besar nilainya maka semakin berbeda stringnya.

e. Hapus pertanyaan dari database

Aplikasi yang dibangun memungkinkan pengguna dapat menghapus sebuah pertanyaan dari database dengan query “Hapus pertanyaan xxx”. Fitur ini dibangun dengan memanfaatkan algoritma pencocokan string yaitu algoritma KMP dan BM yang masing-masing implementasinya terdapat dalam file KMPSearch.js dan BMSearch.js. Masing-masing file ini melakukan extends terhadap getLevenDist. getLevenDist disini merupakan function dari Levenshtein Distance yang digunakan untuk membandingkan seberapa cocok string yang

terdapat pada input dan string yang terdapat pada database. Fungsi getLevenDist mengembalikan jarak Levenshtein Distance yaitu jumlah minimum operasi yang diperlukan untuk mengubah string s1 menjadi s2, dimana semakin besar nilainya maka semakin berbeda stringnya.

3.2 Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

Fitur Fungsional yang terdapat dalam aplikasi web yang dibangun adalah sebagai berikut:

- Aplikasi dapat menerima input pertanyaan baru berbentuk teks mengenai hal umum, operasi matematika, serta tanggal (yang dapat dimasukkan ke dalam database atau dihapus dari database)
- Aplikasi dapat menjawab input pertanyaan menggunakan algoritma KMP atau BM.
- (Bonus) Aplikasi yang dibangun dapat di-deploy menggunakan hosting provider ...
- Aplikasi memiliki halaman yang dapat menampilkan history pertanyaan dan kolom chat yang berisi input query dari pengguna dan jawaban dari chatbot

Sedangkan, arsitektur dari aplikasi web yang dibangun yaitu:

- Pada bagian front-end menggunakan framework React.js, sedangkan bagian back-end menggunakan framework express.js
- Database yang digunakan yaitu MongoDB
- Vercel untuk deployment

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

Spesifikasi teknis program yang kami buat meliputi struktur data, fungsi, dan prosedur yang dibangun. Berikut penjelasan dari masing-masing komponen tersebut:

a. Struktur Data

Data disimpan dengan menggunakan database mongoDB. Terdapat tiga tabel yang digunakan yaitu QnA, *message*, dan *session*. QnA digunakan untuk menampung pertanyaan-pertanyaan yang nantinya akan diproses dan dihitung kemiripannya dengan algoritma-algoritma string matching, *message* digunakan untuk menyimpan *history* suatu *chat*, dan *session* digunakan untuk menyimpan *history* pertanyaan user dan jawaban yang diberikan bot.

b. Fungsi

1. `function buildLast(pattern)`

Fungsi ini memungkinkan algoritma pencocokan pola untuk dengan efisien menghitung pergeseran yang diperlukan dalam setiap iterasi pencocokan dengan cara menyimpan indeks terakhir dari setiap karakter dalam pola dalam array last.

2. `function bmMatch(text, pattern)`

Fungsi ini menerima dua parameter: text yang merupakan teks yang akan dicocokkan dengan pola (pattern), dan pattern yang merupakan pola yang akan dicari di dalam teks. Fungsi ini akan mengembalikan indeks dari awal kemunculan pertama dari pola di dalam teks, atau -1 jika pola tidak ditemukan di dalam teks.

3. `function evaluateExpression(tokens)`

Fungsi ini digunakan untuk mengevaluasi ekspresi matematika dalam bentuk token yang diberikan dalam sebuah array tokens.

4. `function applyOperator(operator)`

Fungsi ini digunakan untuk menerapkan operator yang diberikan ke dua operan teratas pada tumpukan operan.

5. `function precedence(operator)`

Fungsi ini digunakan untuk menentukan urutan operasi (prioritas) saat mengevaluasi ekspresi matematika.

6. `function getDayOfWeek(dateString)`

Fungsi ini digunakan untuk menghitung dan mengembalikan hari dalam seminggu berdasarkan tanggal yang diberikan dalam format "DD/MM/YYYY".

7. `function computeBorder(pattern)`

Fungsi ini digunakan untuk menghitung border dari suatu pola (pattern) pada algoritma *Knuth-Morris-Pratt* (KMP) untuk pencarian string.

8. `function kmpMatch(text, pattern)`

Fungsi `kmpMatch()` merupakan implementasi dari algoritma *Knuth-Morris-Pratt* (KMP) untuk mencari pola tertentu dalam sebuah teks.

9. `function getLevenDist(s1, s2)`

Fungsi untuk mengembalikan jarak *levenshtein distance*, yaitu jumlah minimum operasi yang diperlukan untuk mengubah string `s1` menjadi `s2`. Semakin besar nilainya, maka semakin berbeda stringnya.

10. `function classifyInput(input)`

Fungsi untuk mengklasifikasikan input ke dalam kategori yang berbeda yang terdiri atas *date*, *calculator*, *insert question*, *delete question*, dan *general question*.

c. Prosedur yang dibangun

Pertama, query pertanyaan akan dicek menggunakan regex untuk menentukan pertanyaan tersebut masuk ke fitur mana. Prioritas klasifikasi fitur yang kami buat yaitu tanggal -> kalkulator -> tambah pertanyaan -> hapus pertanyaan -> string matching teks. Kemudian pertanyaan akan diproses di fitur yang masuk pada salah satu klasifikasi tersebut. Apabila pertanyaan tidak ada yang cocok dengan algoritma KMP atau BM, maka ia akan otomatis diperiksa oleh algoritma *Levenshtein Distance*. Namun, apabila pertanyaan tidak ada yang mirip dengan yang ada di database, maka ia akan mengeluarkan pesan "Pertanyaan tidak terdapat dalam database."

4.2 Penjelasan Tata Cara Penggunaan Program

Berikut instalasi dan persiapan program:

- Clone repository dengan link github sebagai berikut
https://github.com/ferindya/Tubes3_13521118.git
- Buka directory folder hasil clone github
- Kemudian install requirements dengan cara buka Command Prompt lalu ketik 'npm install', lalu jalankan dengan mengetik 'npm run dev' pada terminal
- Setelah berhasil diinstal, pastikan halaman yang dibuka adalah <https://localhost:3000>
- Program sudah dapat digunakan dengan mengetikkan pertanyaan pada chat box sesuai dengan fitur-fitur yang dijelaskan pada bagian spesifikasi
- Selain meng-clone repository ini, program juga dapat dijalankan pada link berikut

4.3 Hasil Pengujian

```
D:\GitHub\Tubes3_13521118>node src/algorithm/Main.js
input: Hari apa 12/12/2012
output: Hari Rabu

D:\GitHub\Tubes3_13521118>
```

```
D:\GitHub\Tubes3_13521118>node src/algorithm/Main.js
input: Hitung 1.5 - 2 ^ 3
output: -6.5
```

```
D:\GitHub\Tubes3_13521118>node src/algorithm/Main.js
input: 1 ^ 2 + 3
output: 4
```

```
D:\GitHub\Tubes3_13521118>node src/algorithm/Main.js
input: tambah pertanyaan apa kabar dengan jawaban baik baik saja
output: Pertanyaan apa kabar berhasil ditambahkan dengan jawaban baik baik saja
```

```
D:\GitHub\Tubes3_13521118>node src/algorithm/Main.js  
input: hapus pertanyaan apa kabar  
output: Pertanyaan apa kabar berhasil dihapus
```

4.4 Analisis Hasil Pengujian

Berdasarkan hasil pengujian yang telah dilakukan, dapat dilihat bahwa aplikasi web yang kami buat dapat melakukan beberapa hal, yaitu:

- Pada semua pengujian, hasil sudah sesuai seperti yang diharapkan.
- Pengujian pada backend menggunakan dua algoritma yaitu Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) serta menggunakan algoritma untuk mengukur tingkat kemiripan perbandingan string yaitu Levenshtein Distance sehingga diperoleh hasil yang konsisten, bernilai True jika menghasilkan kemiripan 100% yang menandakan alur algoritma sudah benar.
- Untuk menghasilkan hasil pengujian yang valid, input query pengguna harus terdapat dalam database.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang didapatkan dari hasil penerapan string matching dan regular expression dalam pembuatan chatGPT sederhana adalah sebagai berikut:

1. Aplikasi yang dihasilkan mampu mengimplementasikan algoritma Knuth-Morris-Pratt (KMP) untuk melakukan pencocokan string dengan baik.
2. Aplikasi yang dihasilkan mampu mengimplementasikan algoritma Boyer-Moore (BM) untuk melakukan pencocokan string dengan baik.
3. Aplikasi yang telah dibuat mampu menampilkan history chat pertanyaan dan jawaban.
4. Aplikasi yang telah dibuat mampu menampilkan jawaban yang valid untuk input query dari pengguna berupa persamaan matematika.
5. Aplikasi yang telah dibuat mampu menampilkan jawaban yang valid untuk input query dari pengguna berupa tanggal.
6. Aplikasi yang telah dibuat dapat melakukan aksi menambahkan pertanyaan dan jawaban ke database otomatis untuk input query menambahkan pertanyaan dan jawaban.
7. Aplikasi yang telah dibuat dapat melakukan aksi menghapus pertanyaan dari database otomatis untuk input query menghapus pertanyaan.
8. Aplikasi yang telah dibuat dapat di deploy menggunakan hosting.

5.2 Saran

Berdasarkan tugas besar yang telah kami kerjakan, kami menyadari bahwa program kami baik sisi front-end, back-end, ataupun algoritma masih jauh dari kata sempurna sehingga kami berharap bahwa program kami masih dapat dikembangkan menjadi semakin fungsional, sangkil, dan ringkas. Berikut adalah beberapa bagian yang masih dapat dikembangkan dari program kelompok kami:

- Optimasi efisiensi algoritma dengan menggabungkan Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM) sekaligus dengan Levenshtein Distance agar sekali kerja dan lebih sangkil.
- Mempelajari lebih lanjut mengenai penungguan data dan sifat tipe data dalam JavaScript.

- Meningkatkan fungsionalitas dari halaman agar dapat menampilkan waktu pengguna input query dan waktu chatBot saat menjawab

5.3 Refleksi

Setelah menyelesaikan tugas besar terakhir IF2211 Strategi Algoritma, kami dapat merefleksikan bahwa komunikasi antar anggota kelompok berjalan cukup baik sehingga tidak terjadi miskomunikasi dan kesalahpahaman dalam pengerjaan tugas besar ini, sebelum dimulainya pengerjaan tugas besar ini, sudah dilakukan diskusi untuk membahas pembagian kerja untuk setiap anggota kelompok, dan terakhir kami menyadari perlunya mempelajari algoritma *Knuth-Morris-Pratt* (KMP), *Boyer-Moore* (BM), dan algoritma untuk mengukur tingkat kemiripan perbandingan string seperti algoritma *Hamming Distance*, *Levenshtein Distance*, ataupun *Longest Common Subsequence* dalam menyelesaikan suatu permasalahan.

LAMPIRAN

Github Repository: https://github.com/ferindya/Tubes3_13521118

Link to Video:

Link to Deployed Website:

DAFTAR PUSTAKA

<https://legacy.reactjs.org/tutorial/tutorial.html> diakses pada 27 April 2023

<https://expressjs.com/> diakses pada 27 April 2023

<https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0> diakses pada 28 April 2023

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf> diakses pada 29 April 2023

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf> diakses pada 29 April 2023

<https://www.baeldung.com/cs/string-similarity-edit-distance> diakses pada 30 April 2023

<https://mongoosejs.com/docs/guide.html#timestamps> diakses pada 5 Mei 2023