

**TUGAS BESAR 3**  
**IF2211 STRATEGI ALGORITMA**  
**Pemanfaatan Pattern Matching dalam Membangun Sistem**  
**Deteksi Individu Berbasis Biometrik Melalui Citra Sidik Jari**



**DISUSUN OLEH:**

Aland Mulia Pratama	13522124
Muhammad Zaki	13522136
Muhammad Rasheed Qais Tandjung	13522158

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2024**

## DAFTAR ISI

<b>DESKRIPSI TUGAS</b>	<b>2</b>
1.1 Latar Belakang	2
1.2 Antarmuka Pengguna	2
1.3 Spesifikasi Tugas	3
1.4 Spesifikasi Bonus	4
<b>LANDASAN TEORI</b>	<b>5</b>
2.1 Algoritma Knuth-Morris-Pratt (KMP)	5
2.2 Algoritma Boyer-Moore (BM)	6
2.3 Regular Expression (REGEX)	7
2.4 Teknik Pengukuran Persentase Kemiripan	7
2.5 Penjelasan Singkat Aplikasi Desktop yang Dibangun	8
<b>BAB III</b>	<b>9</b>
<b>ANALISIS PEMECAHAN MASALAH</b>	<b>9</b>
3.1 Langkah-langkah Pemecahan Masalah	9
3.2 Proses penyelesaian solusi dengan Algoritma KMP dan BM	9
3.2.1 Algoritma KMP	9
3.2.2 Algoritma BM	9
3.3 Proses Pencocokan String untuk nama pada sidik_jari dan biodata dengan Regex	10
3.4 Proses Mencari Persentase Kemiripan dengan Hamming Distance	11
3.5 Fitur Fungsional dan Arsitektur Aplikasi Desktop	11
3.6 Contoh Ilustrasi Kasus	14
<b>BAB IV</b>	<b>17</b>
<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>17</b>
4.1 Spesifikasi Teknis Program	17
4.1.1 Struktur Data Program	17
4.1.2 Fungsi & Prosedur yang Dibangun	19
4.2 Penjelasan Tata Cara Penggunaan Program	23
4.3 Hasil Pengujian	25
4.4 Analisis Pengujian	45
<b>BAB V</b>	<b>47</b>
<b>KESIMPULAN</b>	<b>47</b>
5.1 Kesimpulan	47
5.2 Saran	47
5.3 Tanggapan	47
5.4 Refleksi	47
<b>DAFTAR PUSTAKA</b>	<b>49</b>

# BAB I

## DESKRIPSI TUGAS

### 1.1 Latar Belakang

Di era digital ini, pentingnya keamanan data dan akses semakin meningkat. Dalam menghadapi tantangan tersebut, teknologi biometrik, khususnya identifikasi sidik jari, telah menjadi solusi yang semakin populer. Sidik jari merupakan identitas unik yang tidak dapat ditiru, membuatnya menjadi metode akses yang aman dan andal. Teknik penting dalam identifikasi sidik jari adalah pattern matching, yang memungkinkan sistem untuk mencocokkan pola sidik jari dengan cepat dan akurat. Algoritma seperti Knuth-Morris-Pratt dan Boyer-Moore menjadi solusi yang umum digunakan dalam proses ini. Dengan menggabungkan teknologi identifikasi sidik jari dan pattern matching, kami bertujuan untuk membangun sistem identifikasi biometrik yang aman, handal, dan mudah digunakan. Sistem ini memiliki aplikasi potensial dalam berbagai bidang, seperti kontrol akses dan verifikasi identitas dalam transaksi keuangan. Dalam Tugas Besar 3 ini, kami akan mengimplementasikan sistem yang menggunakan deteksi sidik jari berbasis Boyer-Moore dan Knuth-Morris-Pratt, serta menghubungkannya dengan basis data untuk mengenali identitas individu secara lengkap hanya dengan menggunakan sidik jari.

### 1.2 Antarmuka Pengguna

Dalam proyek Tugas Besar ini, kami akan mengembangkan sebuah aplikasi desktop menggunakan bahasa pemrograman C# dengan bantuan kakas WinForm atau WPF untuk mengatur tampilan antarmuka pengguna. Tujuan dari aplikasi ini adalah untuk melakukan identifikasi individu berbasis biometrik menggunakan citra sidik jari yang diberikan oleh pengguna.

Berikut adalah tampilan layout yang akan dibuat dalam aplikasi:

- Judul Aplikasi:** Bagian atas aplikasi akan memiliki judul yang jelas dan menarik perhatian pengguna, memberikan informasi tentang tujuan aplikasi.
- Tombol Insert Citra Sidik Jari:** Terdapat tombol yang memungkinkan pengguna untuk memasukkan citra sidik jari yang ingin dicari biodatanya. Setelah pengguna menekan tombol ini, aplikasi akan membuka jendela atau dialog untuk memilih citra sidik jari dari sistem pengguna.
- Display Citra Sidik Jari yang Dicari:** Di sebelah tombol "Insert Citra Sidik Jari", akan ada display atau panel yang menampilkan citra sidik jari yang telah dimasukkan oleh pengguna.
- Toggle Button untuk Memilih Algoritma:** Terdapat toggle button yang memungkinkan pengguna untuk memilih algoritma pencarian yang ingin digunakan, yaitu Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM).

Tombol Search: Setelah pengguna memasukkan citra sidik jari dan memilih algoritma, mereka dapat menekan tombol "Search" untuk memulai proses pencarian. Program akan

mulai memproses citra sidik jari dan mencocokkannya dengan citra sidik jari yang ada dalam basis data.

5. **Display Biodata dari Sidik Jari yang Paling Mirip:** Setelah proses pencarian selesai, aplikasi akan menampilkan citra sidik jari yang paling mirip dengan citra sidik jari yang dicari oleh pengguna.
6. **Informasi Mengenai Waktu Eksekusi:** Aplikasi akan menampilkan informasi mengenai waktu eksekusi proses pencarian, sehingga pengguna dapat mengetahui seberapa cepat atau lambat aplikasi bekerja.
7. **Informasi Mengenai Tingkat Kemiripan Sidik Jari:** Aplikasi akan memberikan informasi mengenai tingkat kemiripan citra sidik jari yang ditemukan dengan citra sidik jari yang dicari, dalam bentuk persentase (%).
8. **List Biodata Hasil Pencarian:** Aplikasi akan menampilkan daftar biodata dari individu yang memiliki sidik jari paling mirip dengan citra sidik jari yang dicari oleh pengguna. Daftar ini akan mencakup semua nilai atribut yang terkait dengan individu tersebut.



Gambar 2. Tampilan GUI Aplikasi Desktop

### 1.3 Spesifikasi Tugas

Pada Tugas Besar ini, kami akan membuat sebuah sistem yang dapat melakukan identifikasi individu berbasis biometrik dengan menggunakan sidik jari dengan detail sebagai berikut.

1. Sistem dibangun dalam **bahasa C#** dengan kakas Visual Studio .NET yang mengimplementasikan **algoritma KMP, BM, dan Regular Expression** dalam mencocokkan sidik jari dengan biodata yang berpotensi rusak. Pelajarilah C# desktop development, **disarankan untuk menggunakan [Visual Studio](#)** untuk mempermudah penggerjaan.

2. Program dapat memiliki basis data **SQL** yang telah mencocokkan berkas citra sidik jari yang telah ada dengan seorang pribadi. Basis data yang digunakan dibebaskan asalkan **bukan No-SQL** (sebagai contoh, MySQL, PostgreSQL, SQLite).
3. Program dapat menerima masukan sebuah citra sidik jari yang ingin dicocokkan. Apabila citra tersebut memiliki kecocokan di atas batas tertentu (silakan lakukan *tuning* nilai yang tepat) dengan citra yang sudah ada, maka tunjukkan biodata orang tersebut. Apabila di bawah nilai yang telah ditentukan tersebut, memunculkan pesan bahwa sidik jari tidak dikenali.
4. Program memiliki keluaran yang **minimal** mengandung seluruh data yang terdapat pada contoh antarmuka pada bagian [penggunaan program](#).
5. Pengguna dapat memilih algoritma yang ingin digunakan antara KMP atau BM.
6. Biodata yang ditampilkan harus biodata yang memiliki nama yang benar (gunakan Regex untuk memperbaiki nama yang rusak dan gunakan KMP atau BM untuk mencari orang yang paling sesuai).
7. Program memiliki antarmuka yang *user-friendly*. Anda juga dapat menambahkan fitur lain untuk menunjang program yang Anda buat (unsur kreativitas).

#### 1.4 Spesifikasi Bonus

Spesifikasi Bonus bagian pertama meminta untuk melakukan enkripsi terhadap semua data yang tersimpan dalam basis data. Karena data KTP merupakan informasi pribadi yang sensitif, perlu ada langkah tambahan untuk melindungi keamanannya. Kami akan mengimplementasikan skema enkripsi-dekripsi sendiri untuk semua data yang tersimpan dalam basis data. Dengan demikian, jika ada pihak yang mencoba mengakses data secara langsung melalui query SQL, mereka tidak akan dapat membaca data yang sebenarnya. Kami akan memastikan bahwa enkripsi yang kami terapkan cukup kuat untuk melindungi privasi data dengan baik.

Untuk bonus lainnya, terdapat juga membuat video presentasi tentang program yang kami buat dan algoritma yang digunakan. Video ini akan mencakup penjelasan tentang cara kerja program, tampilan aplikasi, serta demonstrasi algoritma yang kami implementasikan. Video ini harus memiliki audio yang jelas dan menampilkan wajah dari setiap anggota tim kami.

## BAB II

### LANDASAN TEORI

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



Gambar 1. Penemu Algoritma Knuth-Morris-Pratt (Donald Knuth, James Morris, Vaughan Pratt)

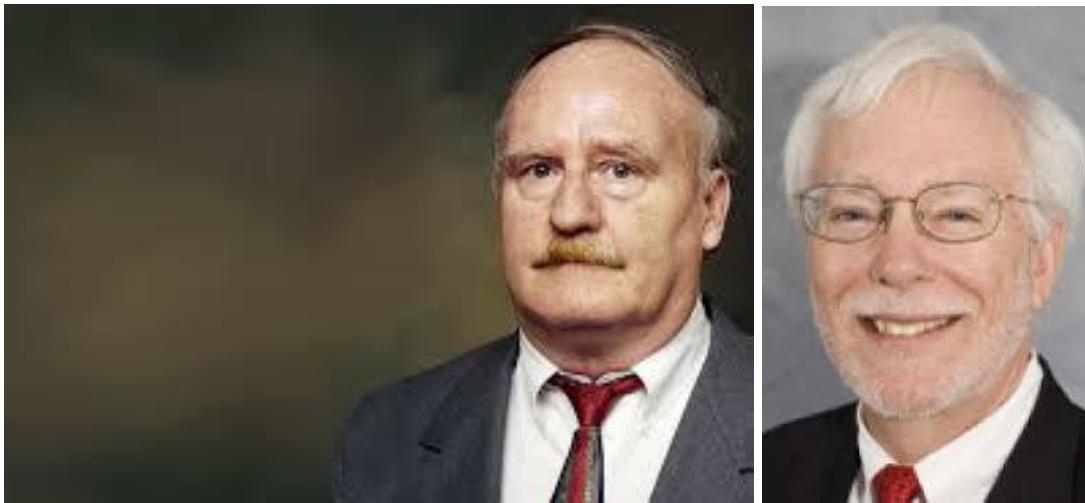
Algoritma Knuth-Morris-Pratt (KMP) adalah salah satu algoritma pencocokan pola yang efisien untuk mencari keberadaan sebuah pola dalam teks. Algoritma ini dinamai sesuai dengan penemunya, Donald Knuth, Vaughan Pratt, dan James H. Morris. Keunggulan utama dari KMP adalah kemampuannya untuk menghindari perbandingan ulang yang tidak perlu dengan memanfaatkan informasi yang terkandung dalam pola itu sendiri. Ini membuatnya jauh lebih cepat dibandingkan metode pencocokan pola yang lebih sederhana, seperti brute force.

Algoritma KMP bekerja dalam dua tahap utama. Tahap pertama adalah pra-pemrosesan, di mana algoritma membangun tabel LPS (Longest Prefix which is also Suffix). Tabel LPS digunakan untuk menyimpan panjang dari prefix terpanjang dari pola yang juga merupakan suffix untuk setiap posisi dalam pola. Informasi ini sangat berguna untuk menentukan seberapa banyak pola dapat digeser saat terjadi ketidakcocokan selama pencarian, sehingga menghindari perbandingan ulang dari awal.

Tahap kedua adalah pencarian pola dalam teks. Dengan bantuan tabel LPS, algoritma KMP dapat menggeser pola dengan cara yang efisien setiap kali terjadi ketidakcocokan. Ini berarti algoritma tidak perlu membandingkan karakter yang sudah diketahui cocok, sehingga menghemat waktu. Selama proses pencarian, jika pola ditemukan cocok dengan bagian teks, posisi pencocokan dicatat, dan pencarian dilanjutkan untuk menemukan semua kemungkinan kemunculan pola dalam teks.

Secara keseluruhan, algoritma KMP memiliki kompleksitas waktu  $O(n+m)$ , di mana  $n$  adalah panjang teks dan  $m$  adalah panjang pola. Kompleksitas ini menjadikannya lebih efisien dibandingkan metode pencocokan pola sederhana yang memiliki kompleksitas  $O(n \times m)$ . Dengan demikian, KMP adalah pilihan yang sangat baik untuk aplikasi yang membutuhkan pencocokan pola cepat dan efisien, seperti dalam pengolahan teks.

## 2.2 Algoritma Boyer-Moore (BM)



**Gambar 2.** Penemu Algoritma Boyer-Moore (Robert Boyer & Strother Moore)

Algoritma Boyer-Moore, yang dikembangkan oleh Robert S. Boyer dan J Strother Moore pada tahun 1977, adalah metode pencarian string yang efisien dalam pemrosesan teks. Algoritma ini dirancang untuk meningkatkan kecepatan pencarian dengan memanfaatkan karakteristik unik dari string yang dicari dan teks sasaran, sehingga mengurangi jumlah perbandingan yang perlu dilakukan selama proses pencarian.

Salah satu komponen utama dari algoritma Boyer-Moore adalah heuristik 'bad character'. Ketika karakter pada string pencarian tidak cocok dengan karakter pada teks, algoritma ini akan mencari keberadaan karakter tersebut di dalam string pencarian. Jika karakter tersebut tidak ditemukan, maka string pencarian bisa melompat sejauh panjang string tersebut. Jika karakter ditemukan dalam string pencarian, string tersebut akan dilompatkan sehingga karakter terakhir yang cocok pada string pencarian sejajar dengan posisi terakhir karakter yang sama di teks.

Heuristik kedua yang digunakan oleh algoritma Boyer-Moore adalah 'good suffix'. Heuristik ini digunakan ketika ada bagian akhir dari string pencarian yang cocok dengan teks, tetapi karakter berikutnya tidak cocok. Algoritma ini akan mencari di dalam string pencarian untuk menemukan kemunculan lain dari sufiks yang cocok ini. Jika sufiks tersebut ditemukan, algoritma akan melompati teks sehingga bagian akhir string pencarian yang cocok tersebut dapat sejajar dengan kemunculan terakhir sufiks tersebut dalam string pencarian.

Proses pencarian menggunakan algoritma Boyer-Moore dimulai dari ujung kanan string pencarian dan teks, membandingkan karakter secara mundur. Ketidakcocokan karakter memicu salah satu dari dua heuristik untuk menentukan jumlah lompatan yang akan dilakukan, meminimalkan jumlah perbandingan yang harus dilakukan. Efisiensi algoritma ini membuatnya sangat populer, terutama untuk pencarian dalam teks yang panjang, karena sering kali jauh lebih cepat daripada algoritma pencarian string tradisional.

### 2.3 Regular Expression (REGEX)

Regular Expression (REGEX) merupakan suatu metode yang sangat bermanfaat dalam pencarian dan manipulasi string berdasarkan pola tertentu. Regex memungkinkan pengguna untuk menentukan suatu pola yang kompleks dalam teks, yang dapat digunakan untuk melakukan pencarian yang sangat spesifik, validasi, serta operasi pemisahan dan penggantian pada string.

Tabel 1. Notasi Regex beserta deskripsi

Regex	Deskripsi
.	Semua karakter kecuali <i>newline</i>
^	Mencocokkan Awal String
\$	Mencocokkan Akhir String
\d,\w,\s	Digit,karakter [A-Za-z0-9],spasi
\D,\W,\S	Kecuali digit, karakter, dan spasi
[abc]	a atau b atau c
[a-z]	Dari karakter a sampai z
\.,\ ,\*	Karakteristik, backslash, dan bintang
aa   bb	aa atau bb
*	Pengulangan karakter sebelumnya sebanyak 0 atau lebih kali
+	Mencocokkan karakter sebelumnya setidaknya satu kali atau lebih.
?	Mencocokkan karakter sebelumnya setidaknya satu kali atau tidak sama sekali.

Regex sering digunakan dalam pemrograman dan pemrosesan teks untuk tugas-tugas seperti validasi input, pencarian dan penggantian teks, serta pemisahan string berdasarkan pola yang ditentukan. Dalam pemrograman, sebuah pola ekspresi reguler yang telah ditentukan dapat digunakan untuk mengidentifikasi dan mengekstrak subteks dari sebuah string menggunakan fungsi bawaan yang tersedia di berbagai bahasa pemrograman.

### 2.4 Teknik Pengukuran Persentase Kemiripan

Hamming Distance adalah suatu metode yang digunakan untuk mengukur perbedaan antara dua string dengan panjang yang sama. Pengukuran ini dilakukan dengan menghitung jumlah posisi di mana karakter-karakter dalam kedua string tersebut berbeda. Hamming Distance bisa digunakan sebagai dasar untuk mengukur persentase kemiripan antara dua string.

Langkah pertama dalam menghitung Hamming Distance adalah memastikan bahwa kedua string memiliki panjang yang identik. Setelah itu, perbandingan karakter-karakter pada

posisi yang sama dilakukan, dimulai dari karakter pertama hingga karakter terakhir pada kedua string. Setiap kali ditemukan perbedaan antara dua karakter pada posisi yang sama, nilai Hamming Distance akan bertambah satu. Setelah seluruh karakter pada kedua string dibandingkan, total jumlah perbedaan akan memberikan nilai Hamming Distance.

## 2.5 Penjelasan Singkat Aplikasi Desktop yang Dibangun

Aplikasi desktop yang dibangun menggunakan bahasa pemrograman C# yang berbasis Windows Forms. Aplikasi ini dirancang untuk memuat citra sidik jari, menganalisisnya, dan mencocokkannya dengan data sidik jari yang tersimpan dalam database. Basis data yang dikembangkan menggunakan DBMS MariaDB dan schema dari database yang digunakan dapat dilihat pada gambar 3. Pada gambar 3, dapat dilihat bahwa relasi biodata dan relasi sidik\_jari tidak terhubung dengan suatu foreign key references. Hal ini disebabkan karena atribut yang mendefinisikan kedua relasi tersebut terdapat *corrupted text* yaitu pada relasi biodata.

'biodata'	
PK	'NIK' varchar(16) NOT NULL
	'nama' varchar(100) DEFAULT NULL
	'tempat_lahir' varchar(50) DEFAULT NULL
	'tanggal_lahir' date DEFAULT NULL
	'jenis_kelamin' enum('Laki-Laki','Perempuan') DEFAULT NULL
	'golongan_darah' varchar(5) DEFAULT NULL
	'alamat' varchar(255) DEFAULT NULL
	'agama' varchar(50) DEFAULT NULL
	'status_perkawinan' enum('Belum Menikah','Menikah','Cerai') DEFAULT NULL
	'pekerjaan' varchar(100) DEFAULT NULL
	'kewarganegaraan' varchar(50) DEFAULT NULL

'sidik_jari'	
	'berkas_citra' text
	'nama' varchar(100) DEFAULT NULL

Gambar 3. Schema Database

Aplikasi Desktop yang dibangun menggunakan bahasa pemrograman C# berbasis Windows Forms menggunakan pendekatan OOP dengan *frontend* dan *backend* yang terintegrasi dalam satu file program. Aplikasi Desktop terdiri dari 1 halaman utama yang terdiri dari beberapa elemen seperti tombol untuk memilih gambar, label untuk menampilkan informasi, dan PictureBox untuk menampilkan citra yang dimuat dan hasil pencocokan. Terdapat juga toggle button dalam aplikasi ini yang memungkinkan pengguna untuk beralih antara algoritma KMP dan BM (Boyer-Moore) untuk pencocokan citra. Saat pengguna mengklik tombol "Search", aplikasi memulai proses pencocokan dengan algoritma yang dipilih. Waktu yang dibutuhkan untuk proses pencocokan dicatat dan ditampilkan kepada pengguna. Jika ditemukan kecocokan, informasi biodata yang sesuai dengan sidik jari tersebut ditampilkan pada label di antarmuka pengguna.

## **BAB III**

### **ANALISIS PEMECAHAN MASALAH**

#### **3.1 Langkah-langkah Pemecahan Masalah**

Pemecahan masalah dimulai dengan mengolah citra yang diberikan user, citra tersebut diambil sebanyak 30 pixel. Setelah diambil 30 pixel gambar tersebut diolah atau dikonversi menjadi string *binary* yang selanjutnya akan diubah lagi menjadi string ascii 8 bit yang akan dijadikan sebagai pattern atau acuan. Lalu dengan algoritma KMP atau BM akan ditelusuri apakah ada pattern tersebut ada di gambar fingerprint lainnya yang ada pada basis data, jika ada maka pencarian dihentikan dan jika tidak ada maka akan digunakan algoritma Hamming Distance untuk mencari kemiripan antara satu string dengan yang lain, dari algoritma itulah kita mendapatkan string yang berkemungkinan memiliki kemiripan yang serupa. Setelah mendapat pattern yang match atau yang mirip dari database , selanjutnya dilakukan pencarian di tabel basis data, karena tabel basis data itu corrupt dilakukanlah regex untuk mencari nama yang sesuai dengan nama yang ada pada tabel sidik\_jari.

#### **3.2 Proses penyelesaian solusi dengan Algoritma KMP dan BM**

##### **3.2.1 Algoritma KMP**

Algoritma Knuth-Morris-Pratt (KMP) untuk pencocokan string dirancang untuk mencari kemunculan suatu pola (pattern) dalam sebuah teks. Metode kmpMatch berfungsi untuk mencari indeks di mana pola tersebut mulai muncul di teks, atau mengembalikan -1 jika pola tidak ditemukan. Algoritma ini dimulai dengan menghitung panjang teks dan pola, kemudian menggunakan metode computeBorder untuk membuat array border ( $b[]$ ) yang menyimpan panjang dari proper prefix yang juga merupakan suffix untuk setiap posisi dalam pola. Proper prefix adalah awalan dari string yang bukan keseluruhan string itu sendiri.

Array border ini membantu dalam menghindari perulangan pencocokan yang tidak perlu. Dalam kmpMatch, pencocokan dilakukan karakter demi karakter antara pola dan teks. Jika ada kecocokan, pencocokan berlanjut hingga seluruh pola ditemukan dalam teks, dan indeks awal kecocokan dikembalikan. Jika ada ketidakcocokan, algoritma menggunakan array border untuk menentukan posisi pencocokan berikutnya, sehingga menghindari pencocokan ulang karakter yang sudah diketahui cocok.

##### **3.2.2 Algoritma BM**

Metode bmMatch memulai dengan membangun array last melalui metode buildLast, yang menyimpan indeks kemunculan terakhir dari setiap karakter ASCII dalam pola. Hal ini membantu menentukan seberapa jauh pola harus digeser jika terjadi ketidakcocokan. Algoritma BM bekerja dari kanan ke kiri dalam pola. Jika pola lebih panjang dari teks, langsung dikembalikan -1 karena tidak mungkin ada kecocokan.

Dalam bmMatch, pencocokan dimulai dari karakter terakhir pola dan teks. Jika ada kecocokan, pencocokan berlanjut mundur hingga seluruh pola cocok, dan indeks awal kecocokan dikembalikan. Jika terjadi ketidakcocokan, algoritma menggunakan dua teknik: "looking-glass" dan "character jump". "Looking-glass" melanjutkan pencocokan mundur, sementara "character jump" menggunakan informasi dari array last untuk melompatkan pola sehingga menghindari pencocokan karakter yang sudah pasti tidak akan cocok.

### 3.3 Proses Pencocokan String untuk nama pada sidik\_jari dan biodata dengan Regex

Fungsi NameToRegex dalam kelas AlayConverter mengubah nama yang diberikan menjadi ekspresi reguler (regex) yang memungkinkan variasi karakter. Ini berguna untuk mencocokkan teks dengan berbagai ejaan atau format. Contoh variasi yang diizinkan termasuk huruf besar dan kecil, angka yang mirip dengan huruf yang dapat dilihat lebih lengkap pada Tabel 2.

**Tabel 2.** Replacement character menjadi Regex

Regex	Deskripsi
.	Semua karakter kecuali newline
^	Mencocokkan Awal String
\$	Mencocokkan Akhir String
\d,\w,\s	Digit,karakter [A-Za-z0-9],spasi
\D,\W,\S	Kecuali digit, karakter, dan spasi
[abc]	a atau b atau c
[a-z]	Dari karakter a sampai z
\.,\\,\\*	Karakteristik, backslash, dan bintang
aa   bb	aa atau bb
*	Pengulangan karakter sebelumnya sebanyak 0 atau lebih kali
+	Mencocokkan karakter sebelumnya setidaknya satu kali atau lebih.
?	Mencocokkan karakter sebelumnya setidaknya satu kali atau tidak sama sekali.

Saat tombol diklik (button2\_Click), aplikasi memulai proses pencocokan citra sidik jari menggunakan algoritma KMP. Jika ditemukan kecocokan pada citra sidik jari, aplikasi kemudian mencocokkan nama yang terkait dengan sidik jari tersebut dengan nama-nama dalam data biodata menggunakan regex. Aplikasi desktop akan menampilkan informasi biodata yang sesuai jika nama dari biodata cocok dengan regex yang telah dikonversi dari nama pada sidik\_jari yang cocok.

### 3.4 Proses Mencari Persentase Kemiripan dengan Hamming Distance

Untuk setiap proses pengecekan antara sidik jari masukan dengan sebuah sidik jari pada basis data, akan dihitung juga persentase kemiripan antara kedua sidik jari. Kalkulasi kemiripan ini dilakukan sebagai mitigasi ketika terjadi mutasi data pada sidik jari masukan. Algoritma KMP dan Boyer-Moore mengharuskan sidik jari masukan berupa *exact match* dengan sebuah sidik jari yang ada di basis data, sehingga walaupun terdapat sidik jari yang cocok, jika sidik jari masukan berbeda sedikit saja, maka algoritma akan menganggap kedua sidik jari tersebut berbeda.

Sehingga agar program dapat tetap menemukan sidik jari yang cocok walaupun terdapat perbedaan data, program dapat dibuat agar menganggap dua buah sidik jari cocok jika kemiripan antara dua sidik jari melewati sebuah batas persentase tertentu. Kemiripan ini dapat dikuantifikasi dengan memanfaatkan algoritma *Hamming distance*.

Untuk mencari *Hamming distance* antara sidik jari A dengan sidik jari B, harus dipenuhi prekondisi bahwa telah terdapat string ASCII untuk A dan B. Karena pengecekan dengan algoritma KMP dan Boyer-Moore juga membutuhkan string ASCII tersebut, maka seharusnya prekondisi ini sudah terpenuhi ketika ingin dihitung *Hamming distance* A dan B.

Selanjutnya, string ASCII dari A dan B akan digunakan sebagai input ke fungsi *Hamming distance*. Karena syarat dari fungsi *Hamming distance* adalah kedua string memiliki panjang yang sama, maka string yang lebih panjang akan dipotong agar panjangnya sama dengan string yang lebih pendek.

$$\text{Similarity}(A, B) = 1 - \frac{\text{HammingDistance}(A, B)}{\min(\text{length}(A), \text{length}(B))}$$

Setelah didapat *Hamming distance* antara A dan B, nilai tersebut dibagi dengan panjang string yang lebih pendek. Langkah ini tujuannya untuk menormalisasikan nilai *Hamming distance* di antara 0 dan 1. Nilai normalisasi ini dapat dianggap sebagai persentase perbedaan A dan B. Maka untuk mendapatkan kemiripan A dan B, cukup menghitung  $1 - \text{persentase perbedaan tersebut}$ .

Setelah didapatkan persentase kemiripan kedua sidik jari, memutuskan kecocokan hanya sekedar memilih sebuah batas bawah kemiripan agar kedua sidik jari dianggap cocok. Melalui proses *tuning* dengan melakukan berkali-kali *testing*, didapatkan bahwa dua sidik jari yang berbeda umumnya memiliki kemiripan dalam rentang  $20\% \leq s \leq 50\%$ , sedangkan sidik jari yang terkorupsi umumnya memiliki rentang kemiripan dalam rentang  $60\% \leq s \leq 95\%$ , sehingga program akan menggunakan batas bawah  $B = 65\%$ , dan seluruh nilai  $s \geq 65\%$  akan dianggap sebagai dua buah sidik jari yang cocok.

### 3.5 Fitur Fungsional dan Arsitektur Aplikasi Desktop

#### 1) Upload Gambar Sidik Jari

- Komponen: PictureBox imageUploader, Button button1
- Deskripsi: Mengunggah gambar sidik jari dari file lokal ke aplikasi. Format gambar yang didukung termasuk BMP, JPG, dan PNG.
- Fungsi: button1\_Click

- 2) Upload Gambar Sidik Jari
  - Menampilkan Gambar yang Dipilih
  - Komponen: PictureBox imageUploader
  - Deskripsi: Menampilkan gambar sidik jari yang dipilih oleh pengguna.
  - Fungsi: button1\_Click
- 3) Toggle Pemilihan Algoritma Pencarian
  - Komponen: ToggleButton toggleButton1
  - Deskripsi: Mengaktifkan atau menonaktifkan penggunaan algoritma KMP atau BM (Boyer-Moore) untuk pencarian gambar sidik jari yang cocok.
  - Fungsi: toggleButton1\_CheckedChanged
- 4) Toggle Pemilihan Algoritma Pencarian
  - Pencarian Sidik Jari yang Cocok
  - Komponen: Button button2, Label label9, Label label10
  - Deskripsi: Melakukan pencarian gambar sidik jari yang cocok dari database menggunakan algoritma yang dipilih (KMP atau BM). Menampilkan waktu pencarian dan persentase kecocokan.
  - Fungsi: button2\_Click
- 5) Toggle Pemilihan Algoritma Pencarian
  - Menampilkan Gambar Sidik Jari yang Cocok
  - Komponen: PictureBox pictureBox1
  - Deskripsi: Menampilkan gambar sidik jari yang ditemukan cocok dengan gambar yang diunggah.
  - Fungsi: button2\_Click
- 6) Menampilkan Data Biodata yang Cocok
  - Komponen: Label NIK, Label Nama, Label TempatLahir, Label TanggalLahir, Label GolonganDarah, Label Alamat, Label Agama, Label StatusPerkawinan, Label Pekerjaan, Label Kewarganegaraan, Label JenisKelamin
  - Deskripsi: Menampilkan informasi biodata yang sesuai dengan sidik jari yang ditemukan cocok.
  - Fungsi: button2\_Click
- 7) Menampilkan Informasi Aplikasi
  - Komponen: Label label7, Label label8
  - Deskripsi: Menampilkan judul aplikasi dan keterangan tentang proyek tugas besar.

- Fungsi: InitializeComponent
- 8) Menampilkan dan Mengubah Status Tombol Toggle
- Komponen: Label label1, Label label2
  - Deskripsi: Menampilkan status dari toggle antara algoritma KMP dan BM.
  - Fungsi: label1\_Click, label2\_Click
- 9) Waktu pencarian dan persentase kecocokan sidik jari
- Komponen: Label label9, Label label10
  - Deskripsi: Menampilkan waktu pencarian dan persentase kecocokan sidik jari.
  - Fungsi: button2\_Click
- 10) Pengaturan Batas Bawah Kecocokan
- Komponen: comboBox1\_SelectedIndexChanged, Label17
  - Deskripsi: Mengatur batas bawah kecocokan hamming distance melalui interface aplikasi desktop
  - Fungsi: button2\_Click



(a)



(b)



(c)

**Gambar 4.** (a) Tangkapan layar program saat sedang melakukan pencarian, (b) Tangkapan layar program setelah menemukan sidik jari yang cocok, (c) pengaturan batas bawah kecocokan hamming distance

### 3.6 Contoh Ilustrasi Kasus



Gambar 5. Sebuah sidik jari yang terkena mutasi data

Misalkan sebuah perangkat pembaca sidik jari membaca sebuah sidik jari yang dikonversi ke dalam format .BMP seperti pada Gambar 5. Ingin dilakukan pengecekan apakah sidik jari tersebut tersimpan pada basis data. Pada proses pembacaan, terjadi sebuah mutasi data yang menyebabkan data sidik jari terkorupsi. Karena proses pencocokan akan melihat kemiripan antara kedua sidik jari, korupsi data ini seharusnya tidak menjadi masalah.



Gambar 6. Proses pencarian sidik jari pada program

Pertama, file bitmap sidik jari tersebut akan diunggah ke program. Untuk setiap sidik jari yang tersimpan di basis data, program akan membandingkan sidik jari tersebut secara *exact match* (dengan menggunakan algoritma KMP/Boyer-Moore), dan secara kemiripan (dengan menggunakan algoritma *Hamming distance*).

Karena pengecekan menggunakan KMP/Boyer-Moore membutuhkan sebuah substring, maka akan diambil 30 *pixel* kontigu di tengah gambar, yang akan diperlakukan sebagai *pattern* yang ingin dicari di setiap sidik jari pada basis data. Untuk perhitungan kemiripan dengan *Hamming distance* akan tetap menggunakan keseluruhan gambar awal.

```
Using KMP
Path gambar: test\17_M_Right_index_finger.BMP
ASCII similarity: 68.04207%
[ Rr ][ Oo0 ] ? [ Bb8 ][ Bb8 ][ Ii1 ] ? [ ] [ Gg6 ][ Rr ][ Ee3 ] ? [ Vv ][ Ee3 ] ? [ Ss5 ][ Oo0 ] ? [ Nn ]
Match found in biodata:
Nama: Robbi Grevsn
```

**Gambar 7.** Ditemukan sidik jari yang cocok pada basis data, dengan kemiripan 68.04%

Karena gambar sidik jari yang dimasukkan ke program mengalami korupsi data, maka algoritma KMP/Boyer-Moore tidak akan menemukan sidik jari yang cocok. Namun menggunakan algoritma *Hamming distance* didapatkan sebuah sidik jari dengan kecocokan 68.04%, dan karena kecocokannya melewati batas bawah 55%, maka pencarian akan dihentikan, dan program akan memutuskan bahwa sidik jari yang cocok berhasil ditemukan.

Sayangnya pada tabel sidik jari didapatkan korupsi terhadap seluruh nama yang terhubungkan pada setiap sidik jari, sehingga nama yang tertera tidak dapat digunakan untuk mencari biodata pemilik sidik jari yang bersesuaian. Namun diketahui bahwa korupsi nama tidak terlalu signifikan, sehingga menggunakan *Regex* dapat dibuat sebuah format *regular expression* untuk mencari nama pemilik sidik jari yang sesuai.



**Gambar 8.** Program mengeluarkan biodata yang bersesuaian dengan sidik jari yang diberikan

Sehingga walaupun terdapat korupsi data sidik jari masukan serta korupsi nama pada basis data, program dapat tetap menemukan biodata pemilik sidik jari yang sesuai. Ditampilkan gambar sidik jari asli serta biodata pemilik sidik jari tersebut seperti yang terlihat pada Gambar 8, dan didapatkan sidik jari masukan dimiliki oleh seorang perempuan di China bernama Robbi Greveson.

## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

#### **4.1 Spesifikasi Teknis Program**

##### **4.1.1 Struktur Data Program**

Struktur data terdiri dari frontend dan backend. Frontend bertanggung jawab untuk tampilan situs termasuk layout, input, tombol, dll. Backend bertanggung jawab untuk pemrosesan masukan, mengambil dan menyimpan data dari dan ke basis data, serta pemrosesan pencocokan kata, dll. Berikut adalah struktur data yang digunakan pada program / aplikasi desktop kami berdasarkan nama file program:

- 1) Database.cs
  - Class: Database
    - ❖ Static Fields:
      - connectionString (string): Menyimpan string koneksi ke basis data MySQL.
      - connection (MySqlConnection): Menyimpan koneksi ke basis data.
    - ❖ Static Methods:
      - OpenConnection(): Membuka koneksi ke basis data.
      - CloseConnection(): Menutup koneksi ke basis data.
      - GetConnection(): Mengembalikan objek koneksi.
  - Class: Biodata
    - ❖ Fields:
      - NIK, Nama, Tempat\_Lahir, Tanggal\_Lahir, Jenis\_Kelamin, Golongan\_Darah, Alamat, Agama, Status\_Perkawinan, Pekerjaan, Kewarganegaraan (string): Menyimpan informasi biodata.
  - Class: BiodataLoader
    - ❖ Static Fields:
      - biodataList (List<Biodata>): Menyimpan daftar biodata.
    - ❖ Static Methods:
      - LoadDataFromSql(): Memuat data dari basis data ke biodataList.
      - GetBiodataList(): Mengembalikan biodataList.
  - Class: SidikJari
    - ❖ Fields:
      - BerkasCitra, Nama (string): Menyimpan informasi sidik jari.

- Class: SidikJariLoader
  - ❖ Fields:
    - sidikJariList (List<SidikJari>): Menyimpan daftar sidik jari.
  - ❖ Methods
    - LoadDataFromSql(): Memuat data dari basis data ke sidikJariList.
    - GetSidikJariList(): Mengembalikan sidikJariList.

## 2) Logic.cs

- Class: Manipulation
  - ❖ Static Methods
    - ImageToBinary(Bitmap img): Mengonversi gambar ke string biner.
    - ConvertToGrayscale(Bitmap img): Mengonversi gambar menjadi grayscale.
    - BinaryToAscii(string binaryString): Mengonversi string biner menjadi ASCII.
    - CropImage(string imagePath): Memotong gambar dari tengah dengan ukuran tertentu.
    - CropImageContiguous(Bitmap bitmap): Memotong gambar secara kontigu dari tengah.
    - loadImage(string path): Memuat gambar dari path yang diberikan.
- Class: BoyerMoore
  - ❖ Static Methods
    - BuildLast(string pattern): Membangun array last untuk algoritma Boyer-Moore.
    - BmMatch(string text, string pattern): Mencocokkan pattern dalam text menggunakan algoritma Boyer-Moore.
- Class: KMP
  - ❖ Static Methods
    - KMPMatch(string text, string pattern): Mencocokkan pattern dalam text menggunakan algoritma Knuth-Morris-Pratt.
    - ComputeBorder(string pattern): Menghitung array border untuk algoritma KMP.

- Class: AlayConverter
  - ❖ Static Methods
    - NameToRegex(string input): Mengkonversi nama menjadi regex dengan menggunakan karakter yang mirip.
- Static Class: StringExtensions
  - ❖ Static Methods
    - FirstCharToUpper(string input): Mengonversi karakter pertama string menjadi huruf besar dan sisanya menjadi huruf kecil.
- Class: Data
  - ❖ Static Fields:
    - sidikJariList (List<Database.SidikJari>): Menyimpan daftar sidik jari yang dimuat dari basis data.
    - biodataList (List<Database.Biodata>): Menyimpan daftar biodata yang dimuat dari basis data.
    - chosenImageASCII (string): Menyimpan string ASCII dari gambar yang dipilih.
    - isImageChosen (Boolean): Menyimpan status apakah gambar sudah dipilih atau belum.
- Class: HammingDistance
  - ❖ Static Fields:
    - GetHammingDistance(string, string): Mencari *Hamming distance* antara dua buah string.
    - GetSimilarity(string, string): Mencari persentase kemiripan antara dua buah string, dengan memanfaatkan *Hamming distance*.

#### 4.1.2 Fungsi & Prosedur yang Dibangun

Algoritma BM

```
// Fungsi untuk membangun array last yang menyimpan posisi terakhir
// kemunculan setiap karakter dalam pattern
private static int[] BuildLast(string pattern)
{
    int[] last = new int[256]; // Menggunakan 256 untuk semua
    // karakter ASCII
    for (int k = 0; k < 256; k++)
        last[k] = -1;
    for (int i = 0; i < pattern.Length; i++)
        last[pattern[i]] = i;
}
```

```

        last[k] = -1; // Inisialisasi semua karakter ke -1
        for (int k = 0; k < pattern.Length; k++)
            last[pattern[k]] = k; // Set posisi terakhir kemunculan
        karakter
        return last;
    }

public static int BmMatch(string text, string pattern)
{
    int[] last = BuildLast(pattern);
    int n = text.Length;
    int m = pattern.Length;
    int i = m - 1;

    if (i > n - 1)
        return -1; // tidak ada kecocokan jika pola lebih panjang
    dari teks

    int j = m - 1;

    do
    {
        if (pattern[j] == text[i])
        {
            if (j == 0)
                return i; // kecocokan ditemukan
            else
            {
                // teknik looking-glass
                i--;
                j--;
            }
        }
        else
        {
            // teknik character jump
            int lo = last[text[i]]; // posisi terakhir kemunculan
            karakter pada pattern
            i = i + m - Math.Min(j, 1 + lo);
            j = m - 1;
        }
    } while (i <= n - 1);

    return -1; // tidak ada kecocokan
}

```

## Algoritma KMP

```
public static int KMPMatch(string text, string pattern)
{
    int n = text.Length;
    int m = pattern.Length;
    int[] b = ComputeBorder(pattern);
    int i = 0; // indeks pada text
    int j = 0; // indeks pada pattern

    while (i < n)
    {
        if (pattern[j] == text[i])
        {
            if (j == m - 1)
                return i - m + 1; // match ditemukan
            i++;
            j++;
        }
        else if (j > 0)
        {
            j = b[j - 1];
        }
        else
        {
            i++;
        }
    }

    return -1; // tidak ada kecocokan
}

public static int[] ComputeBorder(string pattern)
{
    int m = pattern.Length;
    int[] b = new int[m];
    b[0] = 0;
    int j = 0;
    int i = 1;

    while (i < m)
    {
        if (pattern[i] == pattern[j])
        {
            j++;
            b[i] = j;
        }
    }
}
```

```

        i++;
    }
    else
    {
        if (j > 0)
        {
            j = b[j - 1];
        }
        else
        {
            b[i] = 0;
            i++;
        }
    }
}

return b;
}

```

## Regular Expression (REGEX)

```

public static string NameToRegex(string input)
{
    Dictionary<char, string> map = new Dictionary<char, string>
    {
        { 'a', "[Aa4]?" },
        { 'b', "[Bb8]" },
        { 'c', "[Cc]" },
        { 'd', "[Dd]" },
        { 'e', "[Ee3]?" },
        { 'f', "[Ff]" },
        { 'g', "[Gg6]" },
        { 'h', "[Hh]" },
        { 'i', "[Ii1]?" },
        { 'j', "[Jj]" },
        { 'k', "[Kk]" },
        { 'l', "[Ll]" },
        { 'm', "[Mm]" },
        { 'n', "[Nn]" },
        { 'o', "[Oo0]?" },
        { 'p', "[Pp]" },
        { 'q', "[Qq]" },
        { 'r', "[Rr]" },
        { 's', "[Ss5]" },
        { 't', "[Tt7]?" },
        { 'u', "[Uu]?" },
    };
}

```

```

        {
            'v', "[Vv]"
        },
        {
            'w', "[Ww]"
        },
        {
            'x', "[Xx]"
        },
        {
            'y', "[Yy]"
        },
        {
            'z', "[Zz]"
        },
        {
            ' ', "[ ]"
        }
    };
    foreach (var item in map)
    {
        input = input.Replace(item.Key.ToString(), item.Value);
    }
    return input;
}

```

### Algoritma Hamming Distance

```

public static int GetHammingDistance(string a, string b) {
    // Get min length of both strings
    int minLength = Math.Min(a.Length, b.Length);

    // Trim the longer string
    a = a.Substring(0, minLength);
    b = b.Substring(0, minLength);

    // Calculate hamming distance
    int distance = 0;

    for (int i = 0; i < minLength; i++) {
        if (a[i] != b[i]) {
            distance++;
        }
    }

    return distance;
}

```

## 4.2 Penjelasan Tata Cara Penggunaan Program

Tata cara penggunaan program atau aplikasi desktop berdasarkan interface program dan juga fitur-fitur fungsional yang disediakan dapat dilihat sebagai berikut:

### 1) Memilih Gambar Sidik Jari

- Klik tombol Pilih Citra di bagian kiri bawah antarmuka aplikasi.
- Jendela dialog akan muncul untuk memilih file gambar dari sistem lokal Anda.
- Pilih file gambar yang diinginkan (format yang didukung adalah BMP, JPG, PNG).

- Setelah gambar dipilih, gambar tersebut akan ditampilkan pada kotak di bawah label Sidik Jari Masukan.

## 2) Memilih Algoritma Pencarian

- Gunakan tombol toggle yang berada di sebelah kanan tombol Pilih Citra untuk memilih antara algoritma KMP atau BM.
- Defaultnya adalah KMP, dan Anda dapat mengubahnya ke BM dengan menggeser tombol toggle.

## 3) Melakukan Pencarian

- Klik tombol Search yang berada di sebelah kanan tombol toggle.
- Aplikasi akan memulai proses pencarian dan menampilkan hasilnya pada kotak di bawah label Sidik Jari Cocok.
- Jika ditemukan sidik jari yang cocok, gambar tersebut akan ditampilkan di kotak Sidik Jari Cocok.

## 4) Menampilkan Hasil Pencarian

- Hasil waktu pencarian akan ditampilkan di label Waktu Pencarian di bagian kanan bawah aplikasi.
- Persentase kecocokan hasil pencarian akan ditampilkan di label Persentase Kecocokan.

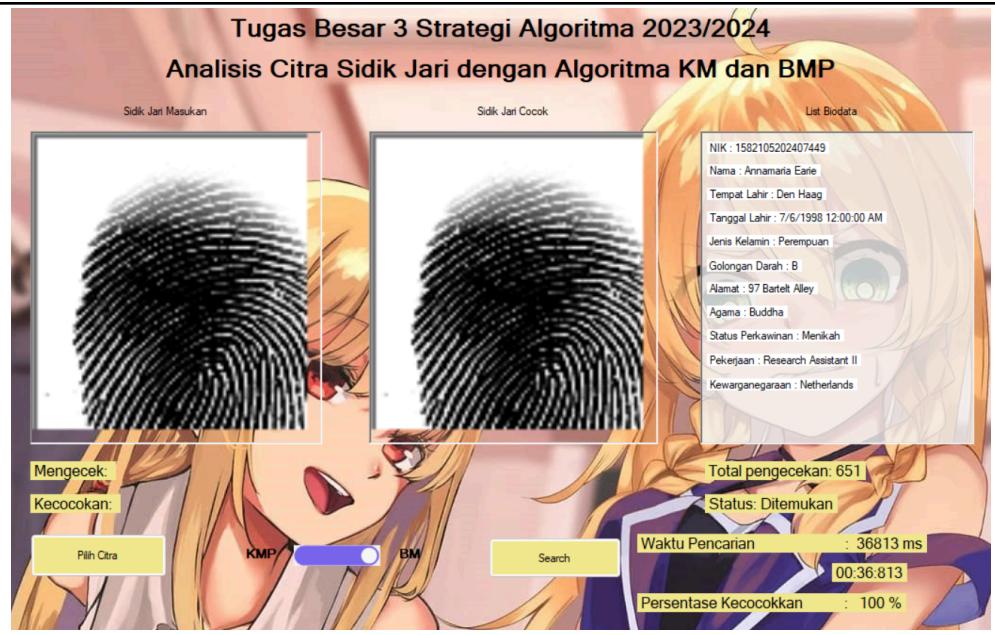
## 5) Melihat Detail Biodata

- Jika pencarian berhasil menemukan sidik jari yang cocok, informasi biodata akan muncul di kotak List Biodata di bagian kanan atas aplikasi.
- Informasi yang ditampilkan meliputi NIK, Nama, Tempat Lahir, Tanggal Lahir, Jenis Kelamin, Golongan Darah, Alamat, Agama, Status Perkawinan, Pekerjaan, dan Kewarganegaraan.

### 4.3 Hasil Pengujian

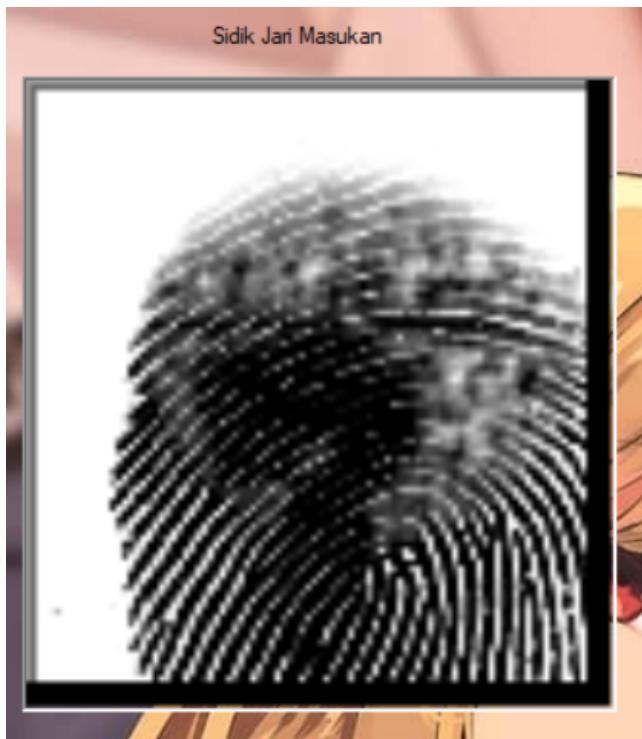
TEST 1 - 1xx, Exact Match	
Input: 15_F_Left_index_finger.BMP	
ALGORITMA	HASIL
KMP	<p style="text-align: center;"><b>Tugas Besar 3 Strategi Algoritma 2023/2024</b></p> <p style="text-align: center;"><b>Analisis Citra Sidik Jari dengan Algoritma KM dan BMP</b></p>

BM



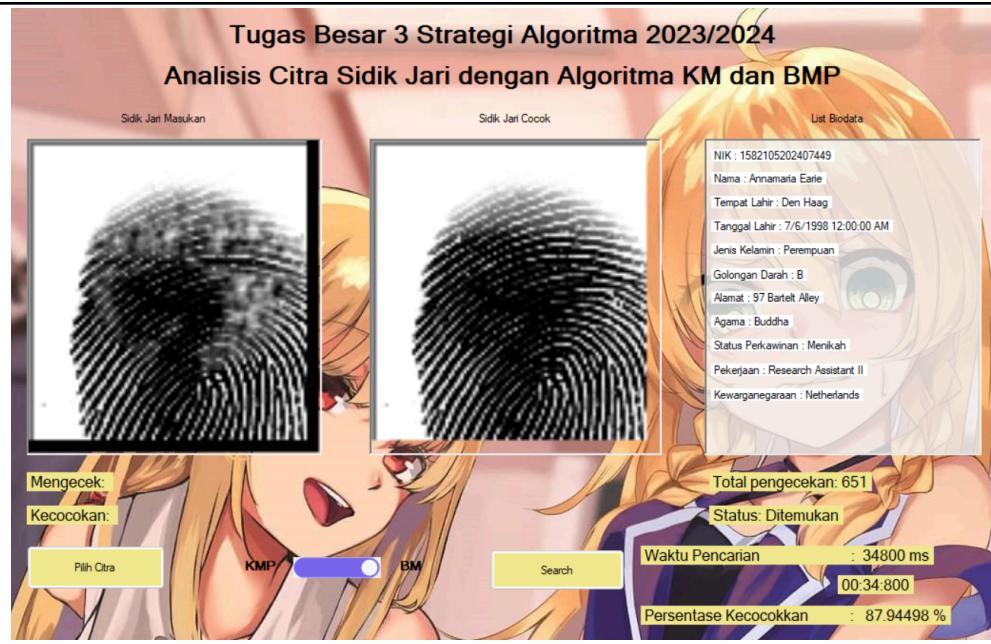
### TEST 2 - 1xx, Altered (Easy)

Input: 15\_F\_Left\_index\_finger\_Obl\_Easy.BMP



ALGORITMA	HASIL
KMP	<p>Tugas Besar 3 Strategi Algoritma 2023/2024</p> <p>Analisis Citra Sidik Jari dengan Algoritma KM dan BMP</p> <p>Sidik Jari Masukan      Sidik Jari Cocok      List Biodata</p> <p>Mengecek: Kecocokan: Pilih Citra      KMP      BM      Search</p> <p>Total pengecekan: 651 Status: Ditemukan Waktu Pencarian : 38169 ms 00:38:169 Percentase Kecocokan : 87.94498 %</p> <p>NIK : 1582105202407449 Nama : Annamaria Earle Tempat Lahir : Den Haag Tanggal Lahir : 7/6/1998 12:00:00 AM Jenis Kelamin : Perempuan Golongan Darah : B Alamat : 97 Bartelt Alley Agama : Buddha Status Perkawinan : Menikah Pekerjaan : Research Assistant II Kewarganegaraan : Netherlands</p>

BM



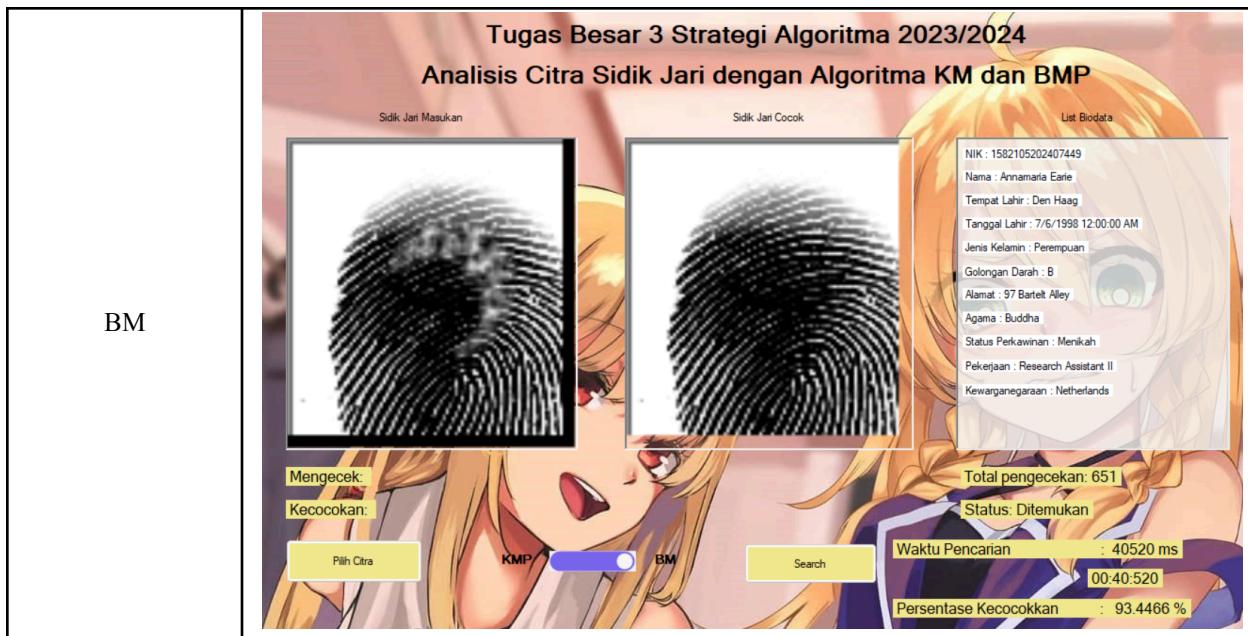
### TEST 3 - 1xx, Altered (Medium)

Input: 15\_F\_Left\_index\_finger\_Obl\_Medium.BMP



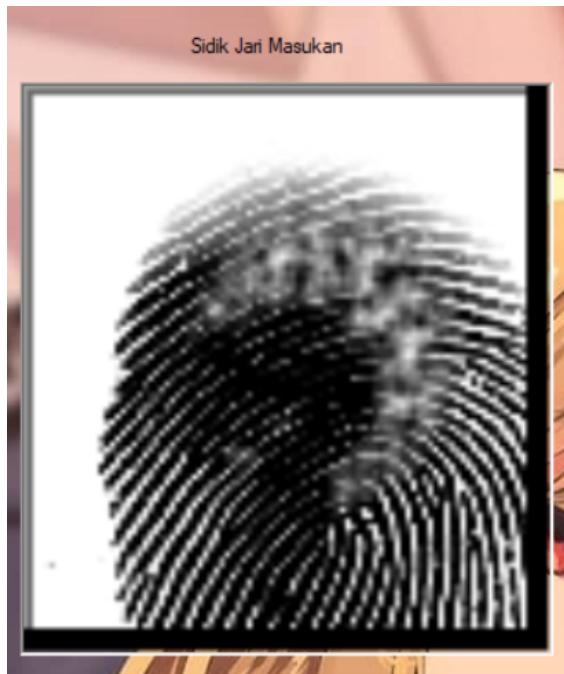
ALGORITMA	HASIL
KMP	<p><b>Tugas Besar 3 Strategi Algoritma 2023/2024</b></p> <p><b>Analisis Citra Sidik Jari dengan Algoritma KM dan BMP</b></p> <p>Sidik Jari Masukan      Sidik Jari Cocok      List Biodata</p> <p>The screenshot shows a user interface for fingerprint analysis. It features three main windows: "Sidik Jari Masukan" (Input Fingerprint) showing the input image, "Sidik Jari Cocok" (Matched Fingerprint) showing a result image, and "List Biodata" (Biodata List) displaying personal information. Below these windows, there are several status and performance metrics. On the left, there are buttons for "Pilih Citra" (Select Image), "KMP" (with a toggle switch), and "BM". On the right, there are buttons for "Search" and "Waktu Pencarian : 34861 ms" (Search Time : 34861 ms). At the bottom, there is a message "Status: Ditemukan" (Status: Found) and a metric "Percentase Kecocokan : 93.4466 %" (Matching Percentage : 93.4466 %).</p> <p>Mengecek: Kecocokan: Pilih Citra      KMP      BM      Search Total pengecekan: 651 Status: Ditemukan Waktu Pencarian : 34861 ms 00:34.861 Percentase Kecocokan : 93.4466 %</p>

BM



#### TEST 4 - 1xx, Altered (Hard)

Input: 15\_F\_Left\_index\_finger\_Obl\_Hard.BMP



ALGORITMA

KMP

HASIL

**Tugas Besar 3 Strategi Algoritma 2023/2024**  
**Analisis Citra Sidik Jari dengan Algoritma KM dan BMP**

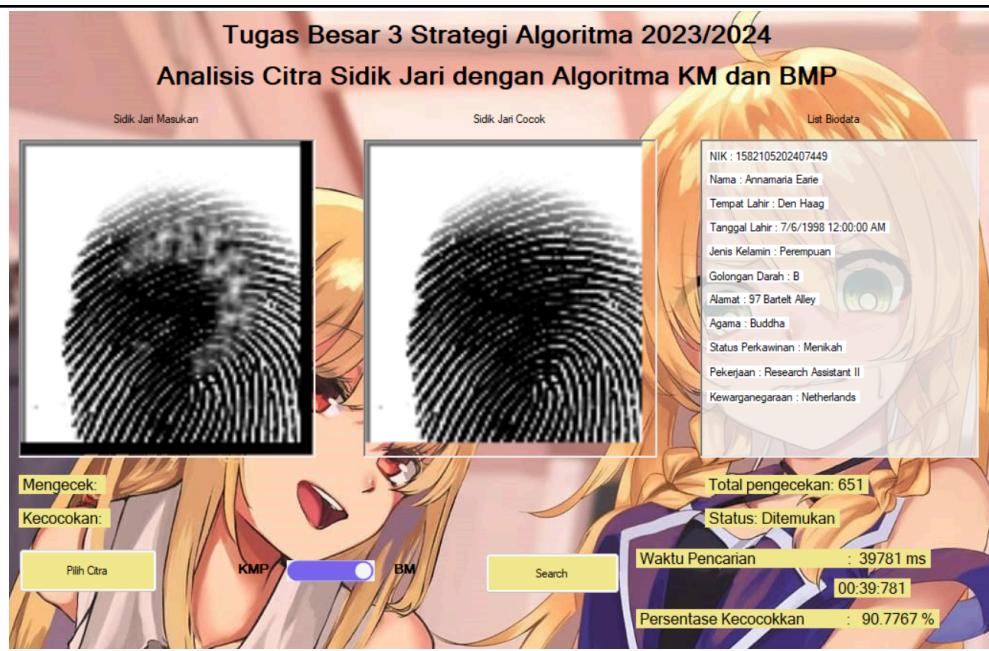
Sidik Jari Masukan      Sidik Jari Cocok      List Biodata

A screenshot of a software interface for fingerprint analysis. It features three main panels: "Sidik Jari Masukan" (Input Fingerprint) showing the input image, "Sidik Jari Cocok" (Matched Fingerprint) showing a matched image, and "List Biodata" (Biodata List) displaying a list of personal information. The "List Biodata" panel includes fields for NIK, Name, Date of Birth, Gender, Blood Type, Address, Religion, Marital Status, Job, and Nationality. At the bottom, there are buttons for "Pilih Citra" (Select Image), "KMP" (KMP algorithm selection), "BM" (BM algorithm selection), "Search" (Search button), and performance metrics: "Total pengecekan: 651", "Status: Ditemukan" (Status: Found), "Waktu Pencarian : 37835 ms" (Search time: 37835 ms), "00:37:835", and "Percentase Kecocokan : 90.7767 %".

Mengecek:  
Kecocokan:  
Pilih Citra      KMP      BM      Search  
Total pengecekan: 651  
Status: Ditemukan  
Waktu Pencarian : 37835 ms  
00:37:835  
Percentase Kecocokan : 90.7767 %

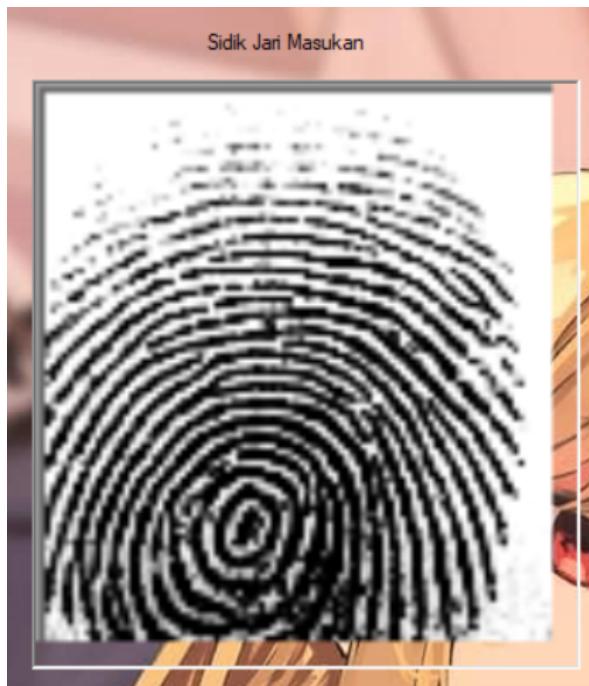
NIK : 1582105202407449  
Nama : Annamaria Earie  
Tempat Lahir : Den Haag  
Tanggal Lahir : 7/6/1998 12:00:00 AM  
Jenis Kelamin : Perempuan  
Golongan Darah : B  
Alamat : 97 Bartelt Alley  
Agama : Buddha  
Status Perkawinan : Menikah  
Pekerjaan : Research Assistant II  
Kewarganegaraan : Netherlands

BM



### TEST 5 - 2xx, Exact Match

Input: 266\_M\_Left\_ring\_finger.BMP



#### ALGORITMA

KMP

#### HASIL

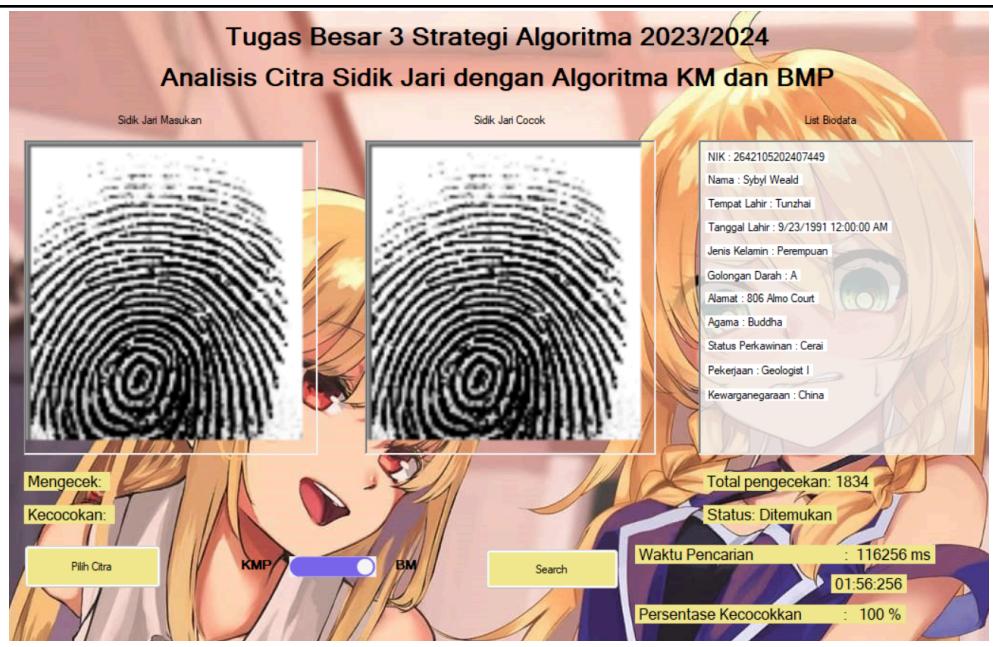
#### Tugas Besar 3 Strategi Algoritma 2023/2024 Analisis Citra Sidik Jari dengan Algoritma KMP dan BMP

The screenshot shows a software interface for fingerprint analysis. It features two main fingerprint images: "Sidik Jari Masukan" (input) on the left and "Sidik Jari Cocok" (match) on the right. Below these images are two buttons: "Pilih Citra" (Select Image) and "Search". Between the images is a toggle switch with "KMP" and "BM" options. To the right of the images is a "List Biodata" box containing the following information:

NIK : 2642105202407449
Nama : Sybil Weald
Tempat Lahir : Tunzhai
Tanggal Lahir : 9/23/1991 12:00:00 AM
Jenis Kelamin : Perempuan
Golongan Darah : A
Alamat : 806 Almo Court
Agama : Buddha
Status Perkawinan : Cerai
Pekerjaan : Geologist I
Kewarganegaraan : China

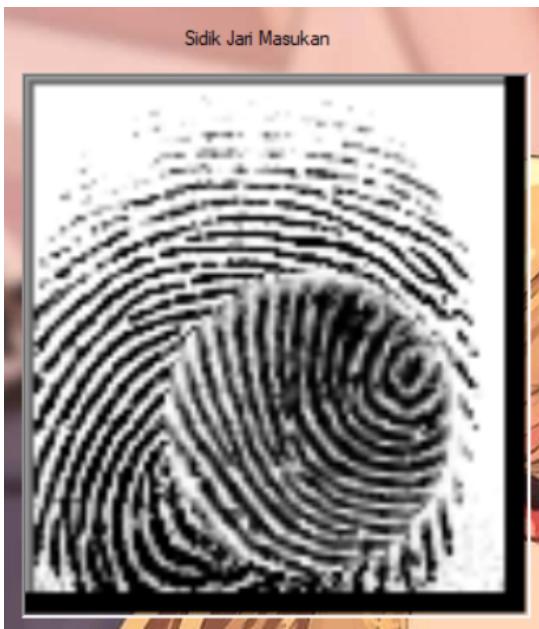
At the bottom right, there are performance metrics: "Total pengecekan: 1834", "Status: Ditemukan", "Waktu Pencarian : 115154 ms (01:55:154)", and "Persentase Kecocokan : 100 %".

BM



### TEST 6 - 2xx, Altered (Hard - CR)

Input: 266\_M\_Left\_ring\_finger\_CR.BMP



ALGORITMA

HASIL

KMP

Tugas Besar 3 Strategi Algoritma 2023/2024

Analisis Citra Sidik Jari dengan Algoritma KM dan BMP

Sidik Jari Masukan      Sidik Jari Cocok      List Biodata

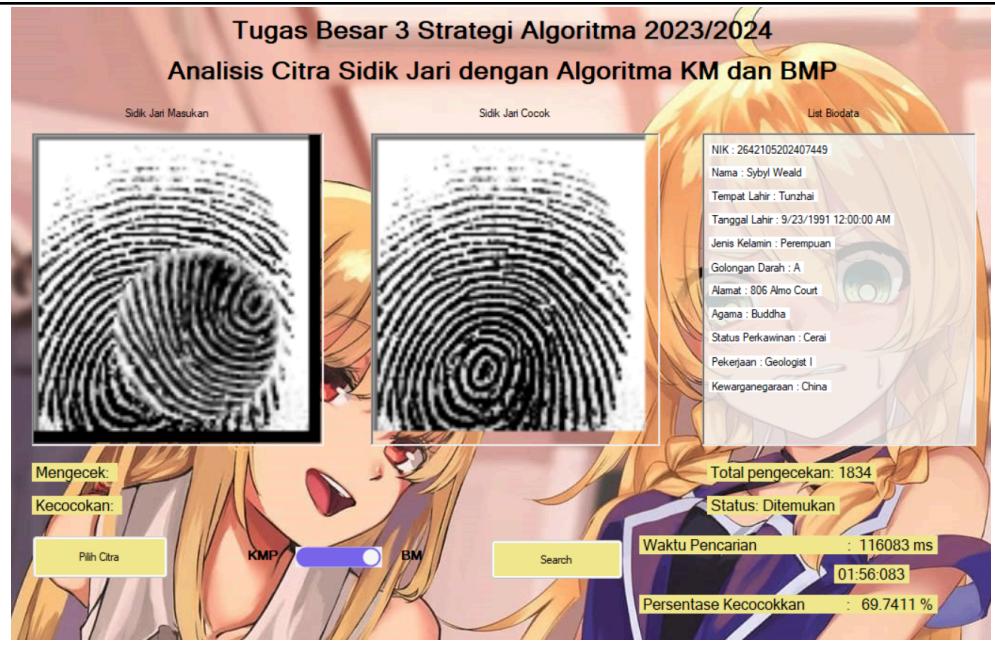
Mengecek: Kecocokan: Total pengecekan: 1834  
Pilih Citra      KMP BM Search Status: Ditemukan

NIK: 2642105202407449  
Nama: Sybil Weald  
Tempat Lahir: Tunzhal  
Tanggal Lahir: 9/23/1991 12:00:00 AM  
Jenis Kelamin: Perempuan  
Golongan Darah: A  
Alamat: 806 Almo Court  
Agama: Buddha  
Status Perkawinan: Cerai  
Pekerjaan: Geologist I  
Kewarganegaraan: China

Waktu Pencarian : 118269 ms  
01:58:269  
Percentase Kecocokan : 69.7411 %

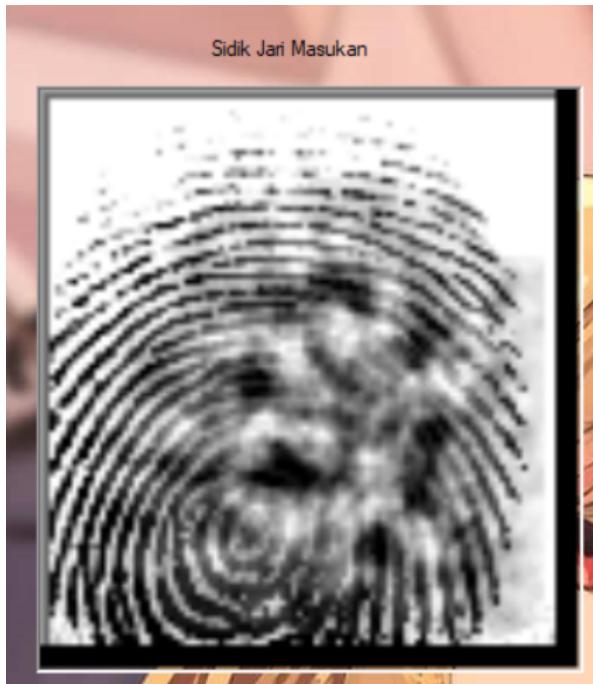
The screenshot displays a user interface for fingerprint analysis. On the left, under 'ALGORITMA', it says 'KMP'. On the right, under 'HASIL', there's a title 'Tugas Besar 3 Strategi Algoritma 2023/2024' and 'Analisis Citra Sidik Jari dengan Algoritma KM dan BMP'. Below the title are three boxes: 'Sidik Jari Masukan' (input fingerprint), 'Sidik Jari Cocok' (matched fingerprint), and 'List Biodata' (biodata of the subject). The 'List Biodata' box contains personal information: NIK, Name, Birthplace, Birthdate, Gender, Blood Type, Address, Religion, Marital Status, Profession, and Nationality. At the bottom, there are performance metrics: 'Total pengecekan: 1834', 'Status: Ditemukan' (Match Found), 'Waktu Pencarian : 118269 ms' (Search Time: 118269 ms), '01:58:269' (Time taken: 1 minute 58 seconds and 269 milliseconds), and 'Percentase Kecocokan : 69.7411 %' (Matching percentage: 69.7411%). The interface features a cartoon character of a woman with blonde hair and red lips in the background.

BM



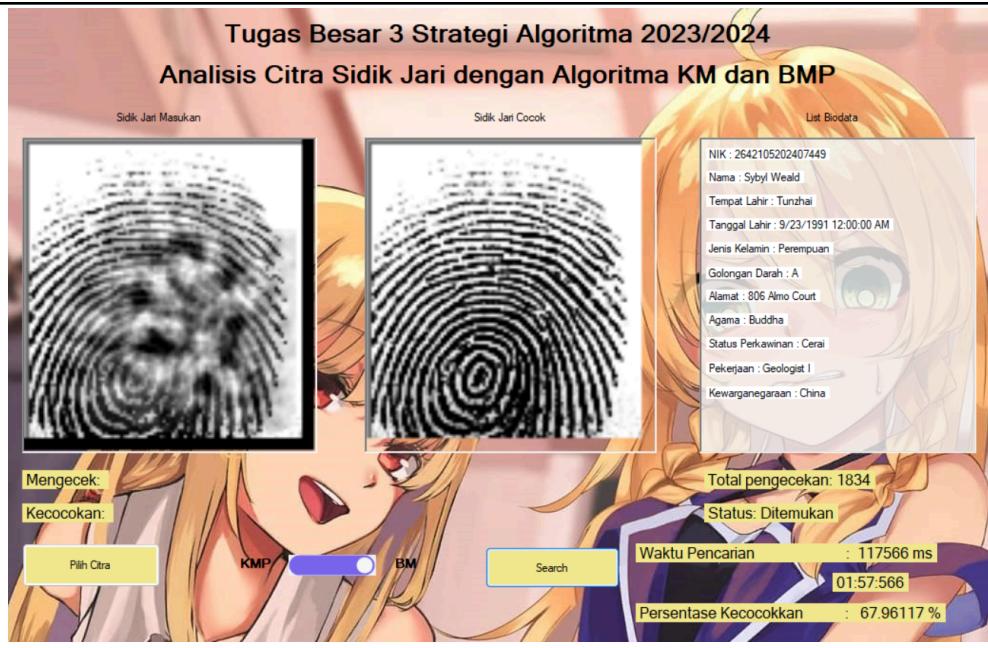
**TEST 7 - 2xx, Altered (Hard - OBL)**

Input: 266\_M\_Left\_ring\_finger\_Obl.BMP



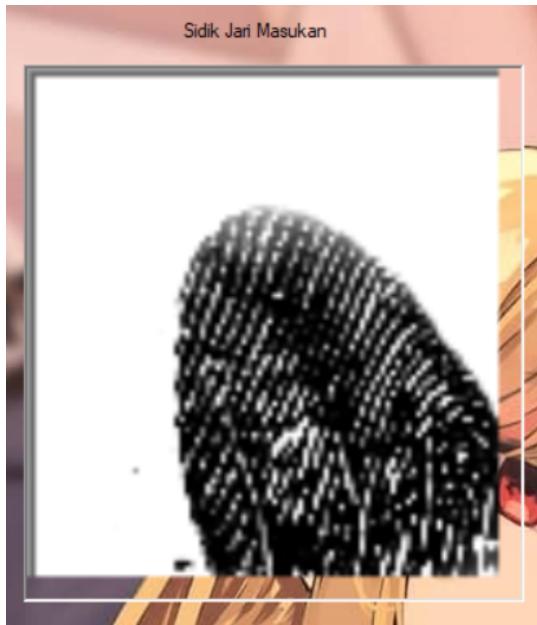
ALGORITMA	HASIL											
KMP	<p align="center"><b>Tugas Besar 3 Strategi Algoritma 2023/2024</b></p> <p align="center"><b>Analisis Citra Sidik Jari dengan Algoritma KM dan BMP</b></p> <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>Sidik Jari Masukan</p> <p>Mengecek: Kecocokan:</p> <p>Pilih Citra</p> </div> <div style="text-align: center;"> <p>Sidik Jari Cocok</p> <p>Search</p> </div> <div> <p>List Biodata</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>NIK : 2642105202407449</td></tr> <tr><td>Nama : Sybil Weald</td></tr> <tr><td>Tempat Lahir : Tunzhai</td></tr> <tr><td>Tanggal Lahir : 9/23/1991 12:00:00 AM</td></tr> <tr><td>Jenis Kelamin : Perempuan</td></tr> <tr><td>Golongan Darah : A</td></tr> <tr><td>Alamat : 806 Almo Court</td></tr> <tr><td>Agama : Buddha</td></tr> <tr><td>Status Perkawinan : Cerai</td></tr> <tr><td>Pekerjaan : Geologist I</td></tr> <tr><td>Kewarganegaraan : China</td></tr> </table> <p>Total pengecekan: 1834</p> <p>Status: Ditemukan</p> <p>Waktu Pencarian : 123953 ms 02:03.953</p> <p>Persentase Kecocokan : 67.96117 %</p> </div> </div>	NIK : 2642105202407449	Nama : Sybil Weald	Tempat Lahir : Tunzhai	Tanggal Lahir : 9/23/1991 12:00:00 AM	Jenis Kelamin : Perempuan	Golongan Darah : A	Alamat : 806 Almo Court	Agama : Buddha	Status Perkawinan : Cerai	Pekerjaan : Geologist I	Kewarganegaraan : China
NIK : 2642105202407449												
Nama : Sybil Weald												
Tempat Lahir : Tunzhai												
Tanggal Lahir : 9/23/1991 12:00:00 AM												
Jenis Kelamin : Perempuan												
Golongan Darah : A												
Alamat : 806 Almo Court												
Agama : Buddha												
Status Perkawinan : Cerai												
Pekerjaan : Geologist I												
Kewarganegaraan : China												

BM



### TEST 8 - 5xx, Exact Match

Input: 523\_F\_Left\_index\_finger.BMP



ALGORITMA

KMP

HASIL

#### Tugas Besar 3 Strategi Algoritma 2023/2024

#### Analisis Citra Sidik Jari dengan Algoritma KM dan BMP

The screenshot shows a software interface for fingerprint analysis. On the left, there is a "Sidik Jari Masukan" (Input Fingerprint) image. In the center, there is a "Sidik Jari Cocok" (Matched Fingerprint) image. To the right, there is a "List Biodata" box containing the following information:

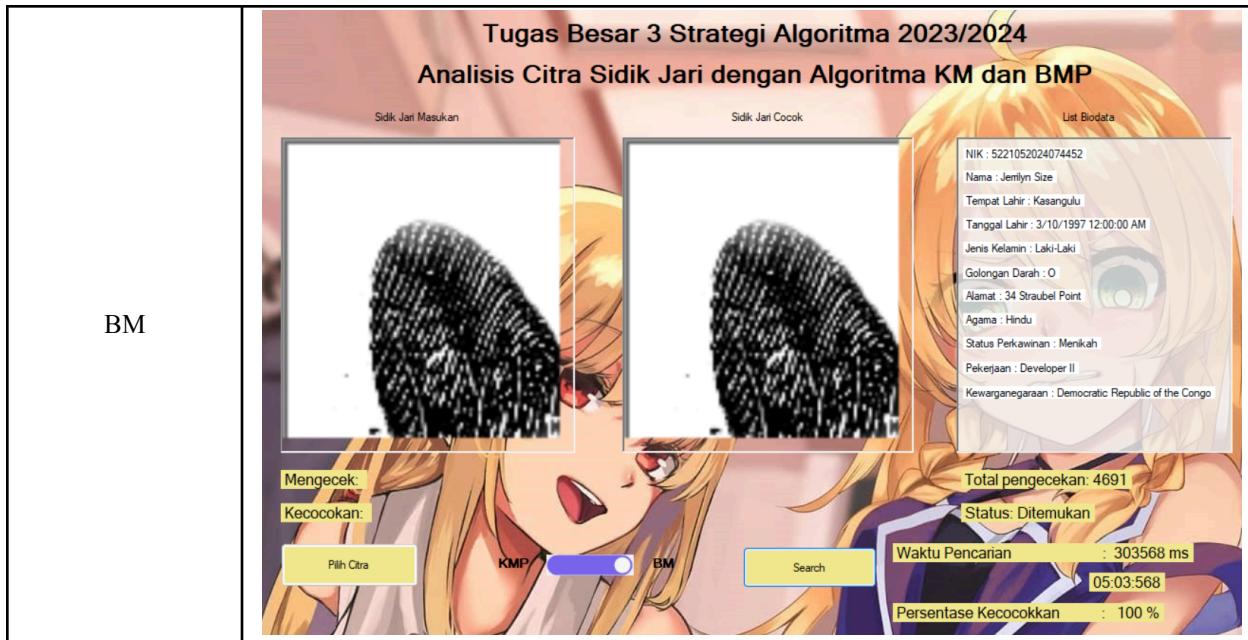
NIK : 5221052024074452
Nama : Jentlyn Size
Tempat Lahir : Kasangulu
Tanggal Lahir : 3/10/1997 12:00:00 AM
Jenis Kelamin : Laki-Laki
Golongan Darah : O
Alamat : 34 Straubel Point
Agama : Hindu
Status Pekawinan : Menikah
Pekerjaan : Developer II
Kewarganegaraan : Democratic Republic of the Congo

Below the biodata, there are several status messages and performance metrics:

- Total pengecekan: 4691
- Status: Ditemukan
- Waktu Pencarian : 307972 ms (05:07:972)
- Persentase Kecocokan : 100 %

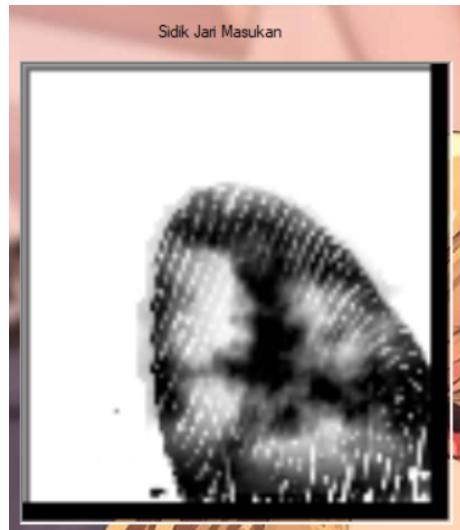
At the bottom left, there are buttons for "Pilih Citra" and "KMP". A slider is positioned between "KMP" and "BM". At the bottom right, there is a "Search" button.

BM

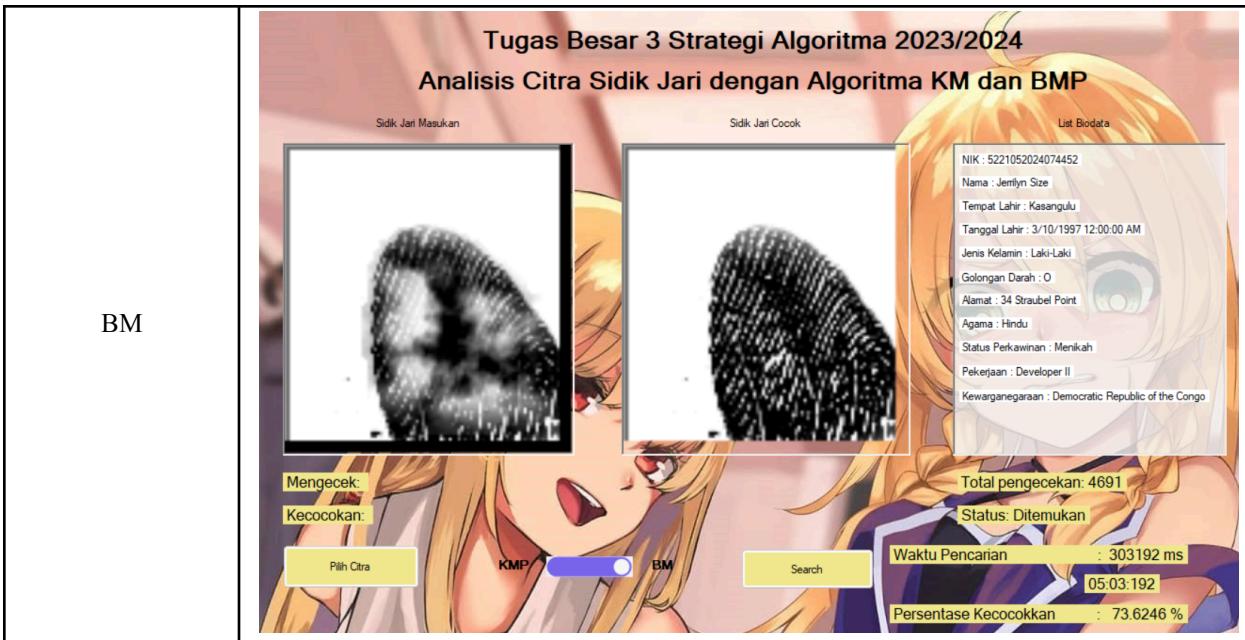


### TEST 9 - 5xx, Altered (Hard)

Input: 523\_F\_Left\_index\_finger\_Obl.BMP

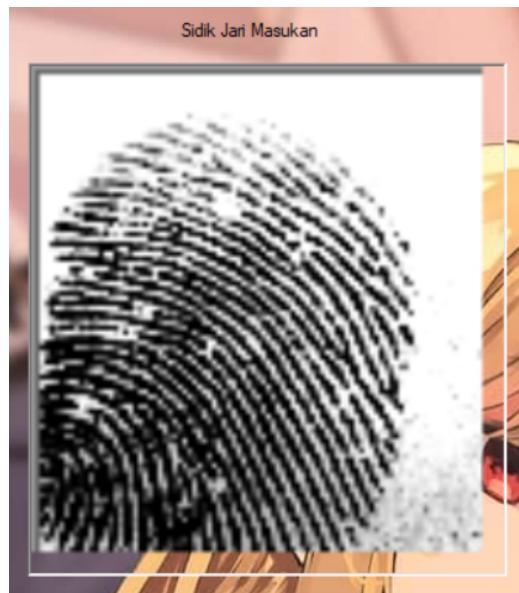


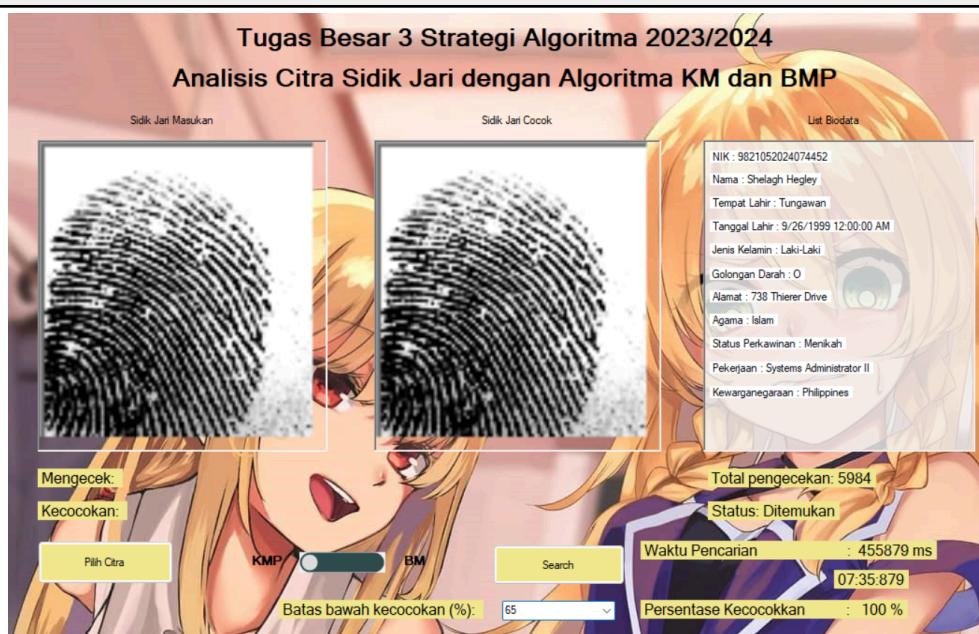
ALGORITMA	HASIL
KMP	<p>Tugas Besar 3 Strategi Algoritma 2023/2024</p> <p>Analisis Citra Sidik Jari dengan Algoritma KM dan BMP</p> <p>Sidik Jari Masukan      Sidik Jari Cocok      List Biodata</p> <p>Mengecek: Kecocokan: Pilih Citra      KMP      BM      Search</p> <p>Total pengecekan: 4691 Status: Ditemukan Waktu Pencarian : 322095 ms 05.22.095 Persentase Kecocokan : 73.6246 %</p> <p>NIK : 5221052024074452 Nama : Jenilyn Size Tempat Lahir : Kasangulu Tanggal Lahir : 3/10/1997 12:00:00 AM Jenis Kelamin : Laki-Laki Golongan Darah : O Alamat : 34 Straubel Point Agama : Hindu Status Perkawinan : Menikah Pekerjaan : Developer II Kewarganegaraan : Democratic Republic of the Congo</p>



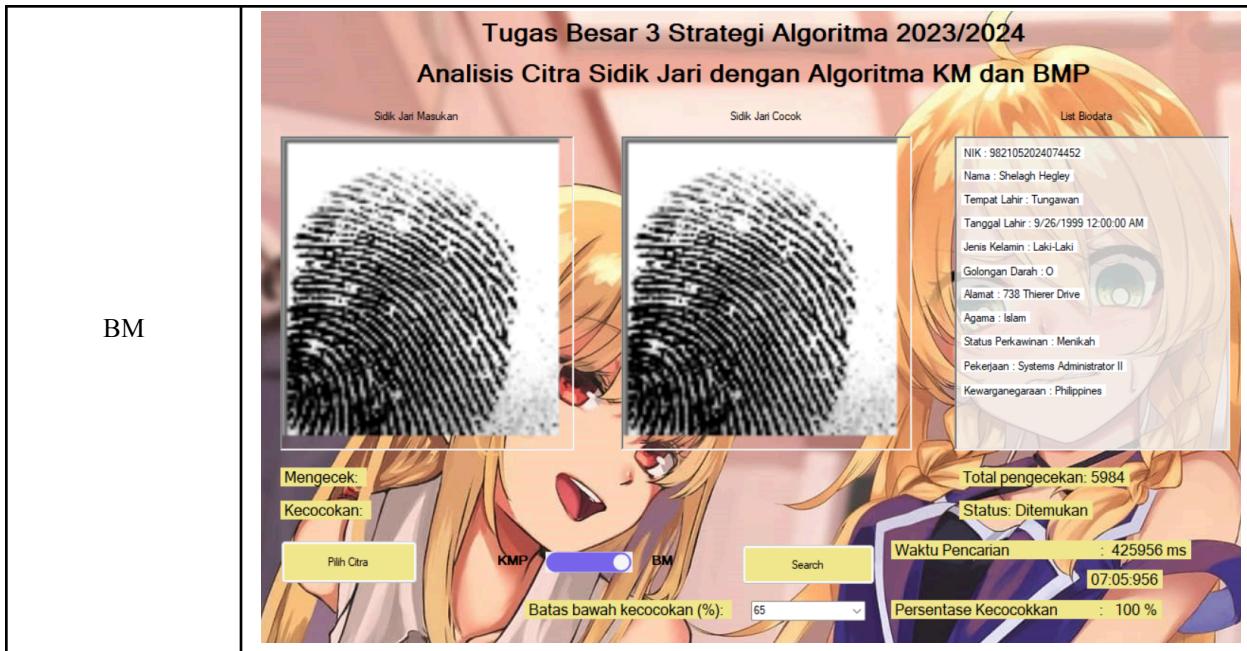
### TEST 10 - 9xx, Exact Match

Input: 99\_M\_Left\_ring\_finger.BMP



ALGORITMA	HASIL
KMP	<p style="text-align: center;"><b>Tugas Besar 3 Strategi Algoritma 2023/2024</b> <b>Analisis Citra Sidik Jari dengan Algoritma KM dan BMP</b></p>  <p>Mengecek: Kecocokan: Pilih Citra KMP BM Search Batas bawah kecocokan (%): 65 Waktu Pencarian : 455879 ms 07:35:879 Percentase Kecocokan : 100 %</p> <p>NIK : 9821052024074452 Nama : Shelagh Hegley Tempat Lahir : Tungawan Tanggal Lahir : 9/26/1999 12:00:00 AM Jenis Kelamin : Laki-Laki Golongan Darah : O Alamat : 738 Thierer Drive Agama : Islam Status Perkawinan : Menikah Pekerjaan : Systems Administrator II Kewarganegaraan : Philippines</p>

BM



#### 4.4 Analisis Pengujian

Dari 10 pengujian yang telah dilakukan diatas, serta 10 pengujian tambahan, didapatkan data perbedaan waktu antara algoritma KMP dengan algoritma Boyer-Moore sebagai berikut (dengan data yang dianggap *outlier* dihapus):

No.	Waktu KMP (ms)	Waktu Boyer-Moore (ms)	Selisih (KMP - BM) (ms)
1	39.493	36.813	2.68
2	38.169	34.8	3.369
3	34.861	40.52	-5.659
4	37.835	39.781	-1.946
5	115.154	116.256	-1.102
6	118.269	116.083	2.186
7	123.953	117.566	6.387
8	307.972	303.568	4.404
9	46.833	48.717	-1.884
10	18.061	17.873	0.188
11	33.054	32.501	0.553
12	9.649	9.475	0.174
13	78.354	78.857	-0.503
14	38.218	38.383	-0.165
15	22.982	22.815	0.167
16	33.073	32.679	0.394
17	149.328	149.647	-0.319
18	300.371	310.772	-10.401
Rata-rata			1.289875

Walaupun didapatkan bahwa algoritma Boyer-Moore lebih cepat dalam lebih banyak kasus dan secara rata-rata Boyer-Moore lebih cepat, keunggulan kecepatan tersebut hanya sebesar 1.28 detik (secara rata-rata), perbedaan yang cukup *negligible* dan tidak terlalu signifikan.

Diketahui bahwa kompleksitas waktu algoritma KMP<sup>1</sup> adalah  $O(m + n)$ , dan kompleksitas waktu algoritma Boyer-Moore<sup>2</sup> adalah  $O(n)$  secara rata-rata, dan  $O(mn)$  secara *worst-case*, dengan m adalah panjang *pattern* yang dicari, dan n adalah panjang teks.

Mengikuti hal tersebut, walaupun secara teknis algoritma Boyer-Moore memiliki kompleksitas waktu yang lebih cepat, pada kasus pengecekan sidik jari pada tugas besar ini, nilai m yang digunakan adalah 30, nilai yang cukup *negligible*. Sehingga tidak terlalu aneh bahwa data yang didapatkan menunjukkan performa kecepatan algoritma KMP dan Boyer-Moore yang tidak terlalu jauh berbeda.

Selain performa algoritma KMP dan Boyer-Moore, variabel lainnya yang dapat dianalisis berupa angka persentase kecocokan dua buah sidik jari yang dihitung menggunakan algoritma *Hamming distance*.

Berdasarkan hasil-hasil pengujian menggunakan gambar sidik jari yang terkena korupsi data, dapat dilihat bahwa penggunaan kemiripan *string* menggunakan *Hamming distance* sudah cukup efektif untuk memitigasi hal tersebut. Walaupun terdapat korupsi data gambar, program tetap dapat mencocokkan sidik jari masukan dengan sidik jari yang sesuai. Jika kedua gambar sama, walaupun salah satu gambar telah termodifikasi, algoritma *Hamming distance* pada umumnya dapat memberikan nilai kemiripan di antara 60% dan 90%, sehingga program dapat tetap menghasilkan *match* untuk sidik jari tersebut.

Program juga sudah memberikan hasil yang baik terkait pencegahan terjadinya *false positive*, yaitu ketika dua sidik jari yang berbeda memiliki nilai kemiripan yang cukup tinggi sehingga dianggap sama oleh program. Menggunakan algoritma *Hamming distance*, dua sidik jari yang berbeda umumnya akan memiliki nilai kemiripan di antara 20% dan 50%, sehingga menggunakan batas bawah kemiripan 65% akan selalu mencegah terjadinya *false positive*, jika diasumsikan seluruh pasangan sidik jari yang berbeda akan memiliki nilai kemiripan di dalam interval tersebut.

---

<sup>1</sup> Shiksha,  
<https://www.shiksha.com/online-courses/articles/introduction-to-kmp-algorithm/#:~:text=The%20KMP%20is%20the%20only,m%20is%20the%20pattern%20length.>, diakses 9 Juni 2024

<sup>2</sup> Encora,  
[https://www.encora.com/insights/the-boyer-moore-horspool-algorithm#:~:text=substring%20cannot%20match.-Complexity.time%20is%20O\(n\).](https://www.encora.com/insights/the-boyer-moore-horspool-algorithm#:~:text=substring%20cannot%20match.-Complexity.time%20is%20O(n).), diakses 9 Juni 2024.

## BAB V

### KESIMPULAN

#### 5.1 Kesimpulan

Pengecekan dua buah sidik jari dapat dianggap sebagai permasalahan membandingkan kesamaan dua buah gambar. Menggunakan algoritma KMP dan Boyer-Moore, dapat dilakukan perbandingan dua buah gambar sidik jari dengan cepat dan efisien, dengan mengubah gambar sidik jari menjadi sebuah *string* ASCII. Jika terdapat korupsi terhadap sidik jari masukan, maka dapat digunakan kemiripan dua buah gambar untuk mencari sidik jari yang cocok pada basis data menggunakan algoritma *Hamming distance*. Jika terhadap korupsi terhadap data bertipe *string* pada basis data, maka dapat digunakan *regular expressions (regex)* untuk mencari data yang sesuai berdasarkan pola data korup yang didapatkan.

Berdasarkan hasil pengujian program pengecekan sidik jari menggunakan algoritma KMP dan Boyer-Moore, didapat hasil bahwa kedua algoritma cukup cepat dan efisien untuk melakukan perbandingan dua buah gambar, dan tidak terlihat ada perbedaan yang signifikan terhadap performa kecepatan kedua algoritma. Jika sidik jari tidak ditemukan secara *exact match*, maka pencocokan sidik jari dapat menggunakan kemiripan gambar menggunakan *Hamming distance*, yang terbukti cukup efektif dalam hasil implementasi.

#### 5.2 Saran

Terkait implementasi program pencocokan sidik jari yang sudah dibuat para penulis, program hanya berupa penerapan yang cukup sederhana dari algoritma KMP, Boyer-Moore, dan *Hamming distance* pada permasalahan perbandingan sidik jari. Pada kasus pengecekan sidik jari di dunia nyata, rasanya diperlukan lebih banyak lagi *testing* dan pengembangan algoritma lebih lanjut untuk memastikan bahwa aplikasi pencocokan sidik jari ini sudah berjalan sesuai yang diharapkan.

#### 5.3 Tanggapan

Para penulis memiliki beberapa tanggapan sendiri terkait hasil implementasi yang sudah dibuat pada tugas besar ini. Pertama, karena proyek ini merupakan proyek pertama para penulis menggunakan kakas C#, Visual Studio, dan WinForms, proyek yang sudah dibuat memiliki struktur file dan kode yang belum tersusun secara rapi. Selain itu, desain *user interface* dari aplikasi masih tergolong cukup sederhana, sehingga jika terdapat waktu luang di masa depan, para penulis berharap untuk dapat mempelajari kakas WinForms lebih lanjut untuk memperbaiki tampilan aplikasi tugas besar ini.

#### 5.4 Refleksi

Saat pertama kali diperkenalkan terkait tugas besar ini, para penulis merasa cukup bingung dan kesusahan terkait cara untuk menyelesaikan masalah yang diberikan. Pertama, karena konsep masalah yang diberikan untuk tugas besar, yaitu pencocokan gambar sidik jari, pada awalnya

terlihat cukup kompleks. Kedua, untuk penggeraan tugas besar ini, diperlukan banyak sekali kakas baru yang dipelajari. Namun, setelah menyelesaikan implementasi programnya, para penulis merasakan bahwa walaupun tetap ada banyak sekali kakas yang perlu dipelajari, secara algoritma yang perlu diimplementasikan tidak terlalu kompleks. Penerapan algoritma KMP dan Boyer-Moore pada permasalahan pencocokan sidik jari cukup sederhana dan mudah dimengerti, sehingga sebagian besar dari kesulitan tugas besar ini hanya pada pembelajaran terkait kakas-kakas baru yang diperlukan untuk implementasi aplikasi.

“Akhirnya menamatkan Tugas Besar terakhir Strategi Algoritma”

## DAFTAR PUSTAKA

- [1] Alay Generator. (n.d.). Alay Generator. Retrieved June 9, 2024, from <https://alaygenerator.blogspot.com/>
- [2] Kartika, E. W., Riana, D., & Kurniawan, D. E. (2019). Fingerprint recognition using minutiae extraction technique. *IOP Conference Series: Materials Science and Engineering*, 662(2), 022040. <https://doi.org/10.1088/1757-899X/662/2/022040>
- [3] Microsoft. (2022, May 25). *Create a C# Windows Forms App in Visual Studio*. Microsoft Learn. Retrieved June 9, 2024, from <https://learn.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022>
- [4] Microsoft. (2021, September 13). *How do I use SQL Server with C# and .NET?*. Microsoft Learn. Retrieved June 9, 2024, from <https://learn.microsoft.com/en-us/shows/on-net/how-do-i-use-sql-server-with-csharp-and-dotnet>
- [5] Ruiz-Garcia, A. (2020). SOCOFing: A dataset for fingerprint presentation attack detection. Kaggle. Retrieved June 9, 2024, from <https://www.kaggle.com/datasets/ruizgara/socofing>

**Tautan Repository GitHub:**

[https://github.com/alandmprtma/Tubes3\\_C-Major](https://github.com/alandmprtma/Tubes3_C-Major)

**BONUS:**

**Tautan Video Youtube:**

<https://youtu.be/3QdyyGS78xo>