

## **LAPORAN TUGAS BESAR III IF2211**

### **STRATEGI ALGORITMA**

*Pemanfaatan Pattern Matching dalam Membangun Sistem Deteksi Individu Berbasis Biometrik Melalui Citra Sidik Jari*



Dosen Pengampu: Dr. Nur Ulfa Maulidevi, S.T, M.Sc.

Asisten pembimbing: Sulthan Dzaky Alfaro

Disusun oleh:

S1T4j4111

Lidya Rahmatul Fitri (10023485)

Bryan Cornelius Lauwrence (13522033)

Salsabiila (13522062)

Julian Chandra Sutadi (13522080)

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

## **Daftar Isi**

Daftar Isi	1
BAB I	2
BAB II	4
2.1. Algoritma Knuth-Morris-Pratt, Boyer-Moore, dan Regular Expression.	4
2.1.1. Algoritma Knuth-Morris-Pratt	4
2.1.2. Algoritma Boyer-Moore	6
2.1.3. Regular Expression	7
2.2. Teknik Pengukuran Persentase Kemiripan	8
2.3. Aplikasi Desktop	8
BAB III	9
3.1. Langkah-Langkah Pemecahan Masalah	9
3.2. Proses Penyelesaian Solusi dengan Algoritma KMP dan BM.	9
3.3. Fitur Fungsional dan Arsitektur Aplikasi Dekstop	10
3.4. Ilustrasi Kasus	11
BAB IV	12
4.1. Spesifikasi Teknis Program	12
4.2. Tata Cara Penggunaan Program	14
4.3. Hasil Pengujian	15
4.4. Analisis Hasil Pengujian	18
BAB V	19
5.1. Kesimpulan dan Saran	19
5.2. Tanggapan	19
5.3. Refleksi	19
LAMPIRAN	20
DAFTAR PUSTAKA	21

## **BAB I**

### **DESKRIPSI TUGAS**

Sistem Identifikasi Individu Berbasis Biometrik Sidik Jari dirancang untuk mengidentifikasi individu dengan mencocokkan sidik jari mereka terhadap biodata yang disimpan dalam basis data. Tujuan utama sistem ini adalah mengembangkan alat identifikasi yang menggunakan biometrik sidik jari untuk mencocokkan sidik jari dengan biodata yang mungkin rusak, serta memastikan keakuratan identifikasi melalui berbagai algoritma pencocokan. Sistem akan dibangun menggunakan bahasa pemrograman CSharp (C#) dan akan mengimplementasikan beberapa algoritma pencocokan, yaitu Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Regular Expression (Regex). Algoritma KMP dan BM digunakan untuk pencocokan pola yang efisien, sedangkan Regex digunakan untuk penanganan dan perbaikan data yang rusak.

Basis data yang digunakan dalam sistem ini adalah SQL, dengan opsi penggunaan MySQL, PostgreSQL, atau SQLite. Basis data ini akan menyimpan biodata dan citra sidik jari individu yang diperlukan untuk proses pencocokan. Fungsi utama dari sistem ini meliputi pencocokan sidik jari dengan biodata yang tersimpan dalam database, penanganan biodata yang rusak, tampilan hasil pencocokan yang mencakup biodata, persentase kemiripan, dan waktu eksekusi, serta kemampuan untuk memilih algoritma pencocokan yang diinginkan, baik KMP maupun BM.

Antarmuka sistem ini dirancang sebagai aplikasi desktop berbasis CSharp dengan elemen-elemen seperti tombol untuk memasukkan citra sidik jari, tampilan citra sidik jari, tombol untuk memilih algoritma, tombol pencarian, tampilan sidik jari yang paling mirip, informasi waktu eksekusi, persentase kemiripan, daftar biodata hasil pencarian, dan desain antarmuka yang ramah pengguna. Penggunaan sistem ini melibatkan beberapa langkah sederhana: memasukkan citra sidik jari, memilih algoritma pencocokan, menekan tombol pencarian, dan melihat hasil pencocokan yang ditampilkan dengan detail biodata, persentase kemiripan, dan waktu eksekusi.

Fitur tambahan yang disarankan untuk pengembangan sistem meliputi kemampuan untuk memperbaiki nama yang rusak menggunakan Regex, serta penambahan fitur kreatif lainnya untuk meningkatkan fungsionalitas program. Penting untuk memperhatikan batasan persentase kemiripan yang dapat diubah sesuai kebutuhan, serta mendokumentasikan kode dengan baik dan menjelaskan algoritma yang digunakan. Pengujian menyeluruh juga

diperlukan untuk memastikan kinerja program yang optimal dan sesuai dengan spesifikasi yang telah ditetapkan. Hal ini mencakup pengujian fungsionalitas, keakuratan pencocokan, dan kinerja sistem di berbagai kondisi untuk memastikan sistem bekerja dengan baik dan memberikan hasil yang diharapkan.

## BAB II

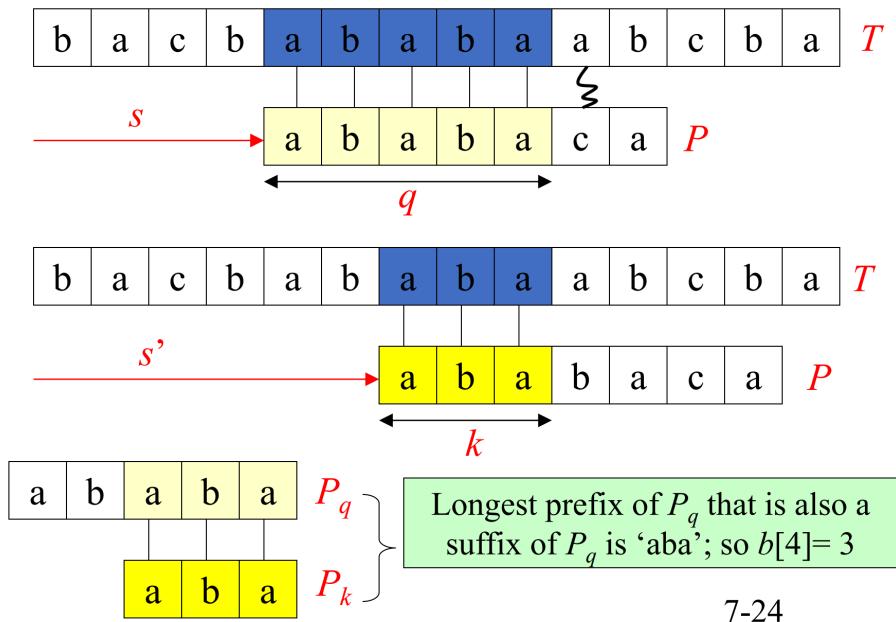
### LANDASAN TEORI

#### 2.1. Algoritma Knuth-Morris-Pratt, Boyer-Moore, dan *Regular Expression*.

Algoritma Knuth-Morris-Pratt, Boyer-Moore, dan *regular expression* menjadi elemen kunci dalam pemanfaatan *pattern matching* untuk membangun sistem deteksi individu berbasis biometrik melalui citra sidik jari. Algoritma Knuth-Morris-Pratt dikenal karena kemampuannya mencocokkan pola dengan teks utama secara efisien dengan kompleksitas waktu yang rendah, berkat pendekatan yang memanfaatkan informasi sebelumnya untuk mempercepat pencarian pola. Sementara itu, Boyer-Moore menonjol dalam kecepatan pencocokan pola dalam teks dengan menggunakan pendekatan heuristik yang mengurangi jumlah karakter yang harus diperiksa, cocok untuk pencarian pola dalam teks kompleks. Sedangkan *regular expression* memberikan fleksibilitas dalam pencocokan pola dengan pola yang kompleks dan beragam, memungkinkan identifikasi format dan struktur data yang berbeda dalam teks. Kombinasi kedua algoritma ini memberikan pondasi yang kuat untuk sistem deteksi individu yang handal dan efisien dalam mengidentifikasi individu berbasis biometrik melalui citra sidik jari.

##### 2.1.1. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencarian kata pada sebuah teks dengan pengecekan yang dilakukan dari kiri ke kanan, tetapi perpindahan posisi dilakukan dengan menggunakan informasi dari karakter yang telah dicocokan sebelumnya. Pada algoritma ini, saat ditemukan karakter yang berbeda saat pencocokan antara kata dengan teks, akan dilakukan pemrosesan untuk menentukan banyak pergeseran kata berdasarkan prefiks dan sufiks pada kata tersebut. Berikut merupakan ilustrasi proses pencocokan kata atau pola (*pattern*) pada teks.



Gambar 1. Ilustrasi Pencocokan Algoritma KMP

(Sumber:

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

Untuk pemrosesan sufiks dan prefiks, algoritma KMP menggunakan fungsi pinggiran atau *border function*  $b(k)$  yang didefinisikan sebagai panjang prefiks terbesar untuk  $P[0.. k]$  yang juga merupakan sufiks  $P[1.. k]$ .  $P$  merupakan kata atau pola yang ingin dicocokan dan  $k$  posisi atau indeks tepat 1 sebelum terjadi ketidakcocokan terjadi. Cara kerja algoritma KMP mirip dengan algoritma *brute force*, hanya berbeda pada proses pergeseran kata atau pola pada saat perbedaan ditemukan. Saat terdapat perbedaan di indeks  $j$  pada kata, maka indeks  $j$  selanjutnya adalah  $j_{new} = b(j - 1)$ . Kompleksitas waktu dari algoritma ini adalah  $O(m + n)$  yang lebih cepat dibandingkan algoritma *brute force*.

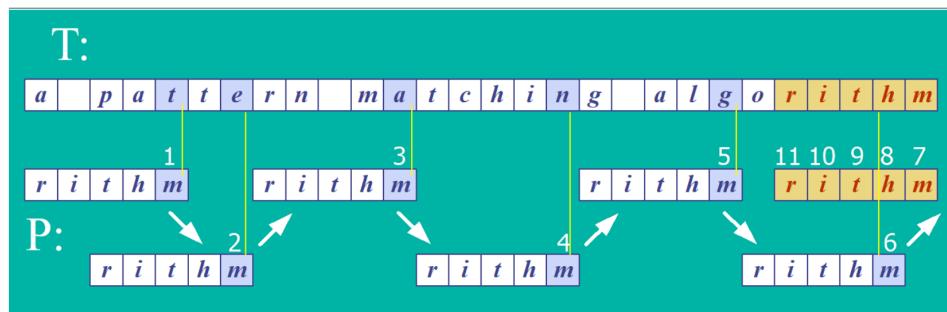
Algoritma KMP memiliki kelebihan yaitu tidak perlu untuk bergerak mundur saat pencocokan kata dengan teks sehingga sangat baik untuk digunakan untuk pencocokan dengan ukuran file yang besar atau pembacaan dari masukan eksternal melalui *network stream*. Namun, algoritma ini kurang baik digunakan jika banyak alfabet pada teks meningkat karena kemungkinan ketidakcocokan akan semakin tinggi (semakin sulit menemukan prefiks dan

sufiks yang sama). Selain itu, algoritma ini cenderung lebih cepat jika ketidakcocokan terjadi di akhir kata atau pola, dan semakin banyak alfabet maka ketidakcocokan akan cenderung lebih banyak ditemukan di awal kata.

### 2.1.2. Algoritma Boyer-Moore

Algoritma Boyer-Moore (BM) merupakan algoritma pencocokan teks yang didasarkan pada dua teknik, yaitu teknik *looking-glass* dan teknik *character-jump*. Teknik *looking-glass* adalah teknik pencocokan yang pengecekan karakternya dimulai dari indeks terakhir pola. Pada algoritma ini, terdapat tiga kasus implementasi teknik *character-jump*. Kasus pertama adalah saat ditemukan ketidakcocokan antara  $P[j]$  dengan  $T[i]$  dan elemen  $T[i]$  berada pada  $P[0..j]$ , elemen  $T[i]$  tersebut akan disejajarkan dengan elemen yang sama pada kata atau pola sehingga didapatkan nilai  $j$  dan  $i$  yang baru.

Pada kasus kedua dari *character-jump*, saat ditemukan ketidakcocokan antara  $P[j]$  dengan  $T[i]$  dan elemen  $T[i]$  berada pada  $P[j..(m - 1)]$ , maka posisi kata atau pola akan digeser sebanyak 1 ke kanan dan  $j$  akan ditempatkan pada  $(m - 1)$ . Terakhir, pada kasus ketiga dari *character-jump*, jika tidak berlaku kasus 1 maupun kasus 2, maka kata atau pola akan digeser sedemikian sehingga  $P[0]$  sejajar dengan  $T[i + 1]$ .



Gambar 2. Ilustrasi Pencocokan Algoritma Boyer-Moore

(Sumber:

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

Algoritma Boyer-Moore dan KMP sama-sama melakukan *preprocessing* terhadap kata atau pola. Perbedaan dari keduanya terletak pada fungsi yang digunakan untuk tahap *preprocessing*, di mana pada algoritma

Boyer-Moore fungsi yang digunakan adalah fungsi *last occurrence* atau  $L()$ . Fungsi  $L(x)$  didefinisikan sebagai indeks terbesar pada kata sehingga  $P[index] == x$  atau  $-1$  jika tidak terdapat  $x$  pada kata atau pola. Kompleksitas waktu dari algoritma ini adalah  $O(nm + A)$  dan algoritma ini akan semakin cepat dengan ukuran alfabet  $A$  yang semakin meningkat, dan sebaliknya.

### 2.1.3. Regular Expression

Regular Expression adalah salah satu bentuk string matching. String matching yang dilakukan adalah pencarian teks dengan pola tertentu. Jadi, selama teks yang dicocokkan memiliki pola yang sama dengan pola regex, teks tersebut akan dianggap valid. Ketentuan pola yang dibuat pada regex sebagai berikut:

QUALIFIERS	QUANTIFIERS	OTHERS	
.	wildcard	+	one or more
\w	alphanumeric & _	*	zero or more
\d	digits	qual?	zero or one
\s	space	quant?	as few as possible <small>(note)</small>
\W	NOT alphanumeric & _	{x}	match exactly x
\D	NOT digits	{x,y}	between x and y
\S	NOT space	{x,}	minimum x
[A-Z]	letter from A to Z		
[a-z]	letter from a to z		
[0-9]	number from 0 to 9		
[abc]	a or b or C (anything from the set)		
[^abc]	anything but those in the set		
		note	
		+ or * are Greedy – match as much as poss	
		+? *? are Lazy / Non-greedy – match as few as poss	

Gambar 1. Ilustrasi Pencocokan Algoritma KMP

(Sumber:

<https://www.thedataschool.com.au/mipadmin/regex-101-in-a-nutshell/>

Regular Expression mencocokkan suatu teks dengan pola tertentu. Regular expression digunakan untuk mencocokkan suatu teks dengan bahasa alaynya. Teks asli akan diubah menjadi bentuk regex bahasa alaynya. Kemudian, regex tersebut akan dicocokkan dengan bahasa alaynya.

## 2.2. Teknik Pengukuran Persentase Kemiripan

Setelah masukan sidik jari telah diberikan, kemiripan akan diukur dengan menggunakan algoritma KMP atau Boyer-Moore, sesuai dengan keinginan pengguna. Jika berhasil ditemukan sidik jari pada basis data dengan menggunakan algoritma ini, maka kemiripan masukan dengan sidik jari pada basis data akan sama dengan 100%. Hal ini hanya terjadi jika seluruh pola yang terdapat pada senarai pola hasil ekstraksi informasi dari masukan berhasil ditemukan pada salah satu sidik jari pada basis data.

Jika tidak ditemukan sidik jari dengan menggunakan salah satu dari algoritma tersebut, maka akan digunakan algoritma *brute force* dengan mempertimbangkan jarak Levenshtein (*Levenshtein distance*). Setiap pola pada senarai akan dihitung *Levenshtein distance*-nya dengan setiap *substring* pada teks dan akan disimpan jarak terkecilnya. Selanjutnya, akan dihitung jarak yang dinormalisasi dengan membagi jarak tersebut dengan panjang pola. Setiap pola pada senarai akan melewati proses ini dan akan didapatkan rata-ratanya. Tingkat kemiripan kemudian dapat dikalkulasikan dengan menggunakan rumus  $similarity = 1 - avgDistance$ .

## 2.3. Aplikasi Desktop

Aplikasi dekstop yang dibangun menggunakan bahasa pemrograman C# dengan UI Framework WPF. Aplikasi dibangun melalui file XAML sehingga hanya bisa dijalankan melalui Windows. Basis data yang digunakan pada aplikasi ini disimpan secara lokal pada MariaDB.

## BAB III

### ANALISIS PEMECAHAN MASALAH

#### 3.1. Langkah-Langkah Pemecahan Masalah

Gambar sidik jari yang dipilih akan diproses terlebih dahulu. Gambar akan dicari bagian terbaik dengan proses normalisasi dan segmentasi. Prosesnya mengambil setiap bagian  $16 \times 16$  dari gambar lalu dicari rata-rata dan standard deviasinya. Dari data tersebut akan ditentukan bagian yang valid. Setelah diperoleh bagian yang valid, gambar diubah menjadi bentuk biner dan dipilih bagian  $64 \times 64$  yang berawal dari kelipatan 8 agar ASCII valid. Potongan gambar diubah menjadi ASCII. Kumpulan ASCII inilah yang akan dibandingkan dengan gambar sidik jari yang tersedia.

Gambar dari database pun diproses dengan cara yang sama. Namun, gambar dari database tidak diambil potongannya melainkan diambil seluruhnya untuk dicocokkan dengan algoritma KMP maupun BM. Jika tidak ditemukan gambar yang mirip melalui algoritma KMP ataupun BM, proses pencarian dilakukan dengan *Levenshtein distance*. Gambar yang paling mirip akan ditampilkan dengan batas minimal kemiripan 70%.

Setelah ditemukan gambar yang mirip, gambar dicocokkan dengan yang ada di *database*. Nama pemilik sidik jari diambil dari pasangan data dari gambar yang ada di tabel sidik jari. Nama tersebut adalah nama aslinya. Namanya akan diubah ke dalam bentuk regex lalu dicari nama alay yang memenuhi regex tersebut dari tabel biodata.

#### 3.2. Proses Penyelesaian Solusi dengan Algoritma KMP dan BM.

Pada program ini, untuk algoritma KMP, terdapat dua metode yang digunakan, yaitu `kmpMatch()` dan `computeBorder()`. Dalam `kmpMatch()`, akan dipanggil metode `computeBorder()` untuk mendapatkan senarai dari  $b(k)$  yang kemudian digunakan untuk melakukan pencocokan seperti yang telah dijelaskan sebelumnya. Sementara itu, untuk algoritma BM, terdapat dua metode yang digunakan, yaitu `bmMatch()` dan `buildLast()`. Dalam `bmMatch()`, akan dipanggil metode `buildLast()` yang membuat senarai dari  $L(x)$  dengan  $x$  adalah setiap karakter pada pola. Metode `kmpMatch` dan

bmMatch akan mengembalikan indeks pertama *substring* yang sama dengan pola masukan dan -1 jika tidak ditemukan *substring* yang sama dengan pola masukan.

### 3.3. Fitur Fungsional dan Arsitektur Aplikasi Dekstop

Aplikasi yang dibangun memiliki antarmuka sebagai berikut:



Gambar 1. Ilustrasi Pencocokan Algoritma KMP

(Sumber: Dokumen Penulis)

Aplikasi dibagi menjadi header dan bagian utama atau isi. Header merupakan bagian yang berwarna hijau tua sebagai judul aplikasi. Bagian sisanya adalah bagian isi untuk menampung masukkan dan keluaran.

Pada bagian isi terdapat tombol “PILIH CITRA”. Tombol ini berfungsi sebagai tombol untuk memilih gambar yang akan diproses dan dicocokkan dengan gambar pada database. Tombol selanjutnya adalah toggle button untuk memilih algoritma yang digunakan. Tombol yang dipilih akan berubah warna menjadi hijau tua. Terakhir, terdapat tombol “CARI” untuk memulai pencarian sidik jari dan biodata. Selain tombol, bagian isi juga diisi dengan tiga kotak. Kotak di bagian kiri adalah kotak untuk menampilkan gambar sidik jari yang dipilih. Kotak di tengah untuk menampilkan gambar sidik jari yang cocok dengan sidik jari yang dipilih jika ditemukan hasil. Kotak di kanan akan menampilkan biodata dari pemilik sidik jari. Setelah pengguna memilih gambar, gambar yang dipilih otomatis akan ditampilkan pada antarmuka. Kemudian, setelah pengguna memulai pencarian, program akan

memulai perhitungannya dan menampilkan sidik jari cocok serta biodata jika ditemukan. Program pun akan menampilkan waktu eksekusi dan persentase kemiripan sidik jari di atas kotak paling kanan.

### 3.4. Ilustrasi Kasus

Pengguna dapat memasukkan citra sidik jari bertipe file bitmap dengan menekan tombol bertuliskan “PILIH CITRA”. Secara *default*, algoritma pencarian akan menggunakan algoritma BM, dan jika pengguna ingin menggunakan algoritma KMP pengguna dapat menekan tombol KMP yang bersebelahan dengan tombol BM, dan sebaliknya. Setelah itu, pengguna dapat menekan tombol “CARI” untuk memulai pencarian. Jika tidak terdapat gambar masukan, maka akan muncul sebuah jendela *pop-up* yang memberikan pesan bahwa tidak terdapat masukan. Apabila setelah dilakukan pencarian ditemukan sidik jari pada basis data yang memiliki tingkat kemiripan di atas 70%, maka sidik jari tersebut akan ditampilkan beserta biodata dari pemilik sidik jari tersebut.

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1. Spesifikasi Teknis Program

Pada aplikasi ini terdapat beberapa namespace atau package, yaitu Algorithms, Controller, Models, dan Repository. Berikut adalah struktur namespace, kelas, dan metode.

Algorithms	
Kelas	Methods
BM	<ul style="list-style-type: none"><li>• public static int bmMatch(string text, string pattern)</li><li>• public static int[] buildLast(string pattern)</li></ul>
KMP	<ul style="list-style-type: none"><li>• public static int kmpMatch(string text, string pattern)</li><li>• public static int[] computeBorder(string pattern)</li></ul>
Levenshtein	<ul style="list-style-type: none"><li>• public static int levenshteinDistance(string string1, string string2)</li><li>• public static int findClosestSubstring(string text, string pattern)</li></ul>

Controller	
Kelas	Methods
FingerprintReader	<ul style="list-style-type: none"><li>• private static (Mat, Mat) createSegmentedImg(Mat img, int w, double threshold)</li><li>• private static Mat applyMorphologicalOperations(Mat img, int size)</li><li>• private static Mat normalizeImage(Mat img, Mat mask)</li><li>• private static Mat convertToBinary(Mat img)</li><li>• private static Mat extractCenterRegion(Mat img, Mat mask, int width, int height)</li></ul>

	<ul style="list-style-type: none"> <li>• private static List&lt;string&gt; convertToAscii(Mat binaryImage)</li> <li>• public static List&lt;string&gt; imgToPattern(string inputFilePath)</li> <li>• public static string imgToText(string inputFilePath)</li> </ul>
Processor	<ul style="list-style-type: none"> <li>• public void setAttributes(string inputFilePath)</li> <li>• public bool bmComparator(string compImg)</li> <li>• public bool kmpComparator(string compImg)</li> <li>• public double levComparator(string compImg)</li> <li>• public double bruteForceLevComparator(string compImg)</li> </ul>

Models	
Kelas	Methods
Fingerprint	<ul style="list-style-type: none"> <li>• public FingerprintRepository(string connectionString)</li> <li>• public async Task&lt;IEnumerable&lt;Fingerprint&gt;&gt; GetAllFingerprintAsync()</li> <li>• public async Task&lt;int&gt; InsertFingerprintAsync(Fingerprint fingerprint)</li> </ul>
Person	<ul style="list-style-type: none"> <li>• Hanya atribut</li> </ul>

Repository	
Kelas	Methods
FingerprintRepository	<ul style="list-style-type: none"> <li>• public FingerprintRepository(string connectionString)</li> <li>• public async Task&lt;IEnumerable&lt;Fingerprint&gt;&gt; GetAllFingerprintAsync()</li> <li>• public async Task&lt;int&gt; InsertFingerprintAsync(Fingerprint fingerprint)</li> </ul>
Person	<ul style="list-style-type: none"> <li>• public PersonRepository(string</li> </ul>

	<pre>connectionString) • public async Task&lt;IEnumerable&lt;Person&gt;&gt; GetAllPersonsAsync() • public async Task&lt;Person&gt; GetPersonByNIKAsync(string nik) • public async Task&lt;Person&gt; GetPersonByNameAsync(string _nama) • public async Task&lt;int&gt; InsertPersonAsync(Person person) • public async Task&lt;int&gt; UpdatePersonAsync(Person person) • public async Task&lt;int&gt; DeletePersonAsync(string nik)</pre>
--	--

## 4.2. Tata Cara Penggunaan Program

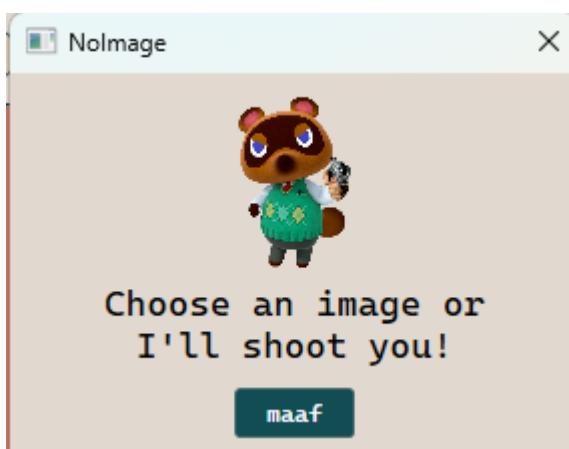
Berikut adalah langkah penggunaan program:

1. Pindah direktori ke src/Biometric lalu jalankan aplikasi.

```
User@DESKTOP-GA6U3VG MINGW64 /d/ITB/1.KULIAH/Semester4/Stima/Tubes-3/Tubes3_S1T4j4111 (main)
$ cd src/Biometric
```

```
User@DESKTOP-GA6U3VG MINGW64 /d/ITB/1.KULIAH/Semester4/Stima/Tubes-3/Tubes3_S1T4j4111/src/Biometric (main)
$ dotnet run
```

2. Aplikasi memiliki *background music* selama program dijalankan
3. Aplikasi dapat mencari sidik jari dengan algoritma *string matching* KMP/BM atau *Levenshtein distance* jika tidak ditemukan sidik jari yang seluruhnya *match*
4. Algoritma memunculkan *pop-up* pesan kesalahan jika pengguna melakukan pencarian sebelum memilih sebuah gambar



5. Masukkan gambar dengan menekan tombol Pilih Gambar



6. Pilih algoritma yang ingin digunakan dengan menekan toggle KMP - BM



7. Tekan tombol Cari. Program akan menampilkan waktu eksekusi dan persentase kemiripan gambar masukan dengan gambar yang dicari.



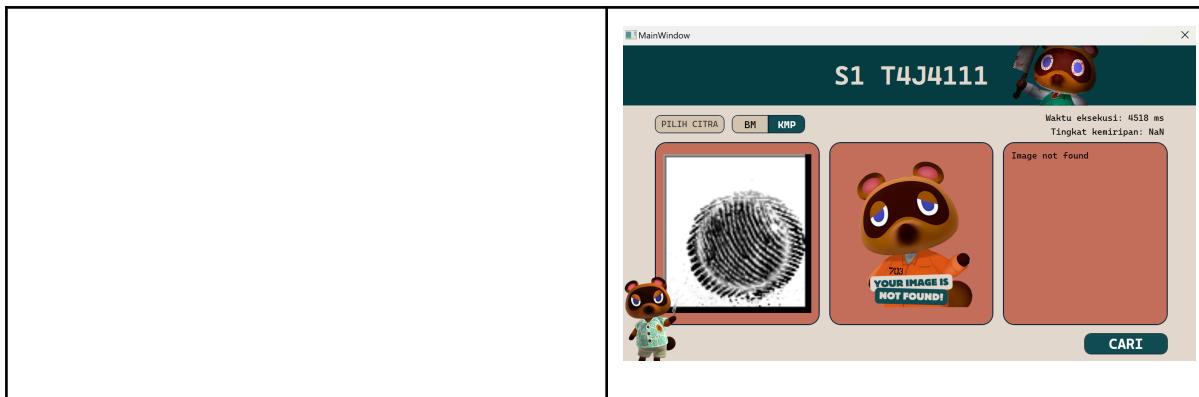
#### 4.3. Hasil Pengujian

Masukkan	Keluaran

IF2211 - Strategi Algoritma  
Tugas Besar 3 - S1 T4j4111

Query Fingerprint	Result Fingerprint	Result Data
		<p>Waktu eksekusi: 5688 ms Tingkat kemiripan: Na</p> <p>Image not found</p>
		<p>Waktu eksekusi: 4022 ms Tingkat kemiripan: 100.0%</p> <p>NIM: 6717211104100 Nama: Sabri Wahyuni Nama: Sabri Wahyuni Tempat lahir: Denpasar Tanggal lahir: 27-3-2001 Jenis Kelamin: Perempuan Alamat: Jalan Cihampelas No. 07 Tangerang Selatan, Kalimantan Selatan 79806 Agama: Kong Hu Cu Status Perkawinan: Belum Menikah Pekerjaan: Dance movement psychotherapist Kewarganegaraan: Indonesia</p>
		<p>Waktu eksekusi: 5920 ms Tingkat kemiripan: 100.0%</p> <p>NIM: 67172111025182 Nama Alay: SdBR WhYn Nama: Sabri Wahyuni Nama: Sabri Wahyuni Tempat lahir: Denpasar Tanggal lahir: 27-3-2001 Jenis Kelamin: Perempuan Alamat: Jalan Cihampelas No. 07 Tangerang Selatan, Kalimantan Selatan 79806 Agama: Kong Hu Cu Status Perkawinan: Belum Menikah Pekerjaan: Dance movement psychotherapist Kewarganegaraan: Indonesia</p>
		<p>Waktu eksekusi: 1331 ms Tingkat kemiripan: 100.0%</p> <p>NIM: S6061611992216 Nama Alay: Jnr lkSMWati Nama: Junari Laksminwati Nama: Junari Laksminwati Tempat lahir: Serang Tanggal lahir: 6-1-1945 Jenis Kelamin: Laki-Laki Alamat: Jalan Stasiun Monokromo No. 49 Batu, KT 38522 Agama: Kristen Status Perkawinan: Cerai Pekerjaan: IT consultant Kewarganegaraan: Indonesia</p>
		<p>Waktu eksekusi: 1517 ms Tingkat kemiripan: 100.0%</p> <p>NIM: S6061611992216 Nama Alay: Jnr lkSMWati Nama: Junari Laksminwati Nama: Junari Laksminwati Tempat lahir: Serang Tanggal lahir: 6-1-1945 Jenis Kelamin: Laki-Laki Alamat: Jalan Stasiun Monokromo No. 49 Batu, KT 38522 Agama: Kristen Status Perkawinan: Cerai Pekerjaan: IT consultant Kewarganegaraan: Indonesia</p>

	 <p>Waktu eksekusi: 279043 ms Tingkat kemiripan: 76.56%</p> <p>NIK: 1708250709140004 Nama Alay: Z4Nab LST12 Nama: Zainab Lestari Tempat lahir: Bandung Tanggal lahir: 1-5-1982 Jenis Kelamin: Laki-Laki Alamat: Gg. Suryakencajan No. 8 Semarang, Jawa Barat 02916 Agama: Kristen Status Pekerjaan: Cewek Pekerjaan: Video editor Kewarganegaraan: Indonesia</p>
	 <p>Waktu eksekusi: 271121 ms Tingkat kemiripan: 76.56%</p> <p>NIK: 1708250709140004 Nama Alay: Z4Nab LST12 Nama: Zainab Lestari Tempat lahir: Bandung Tanggal lahir: 1-5-1982 Jenis Kelamin: Laki-Laki Alamat: Gg. Suryakencajan No. 8 Semarang, Jawa Barat 02916 Agama: Kristen Status Pekerjaan: Cewek Pekerjaan: Video editor Kewarganegaraan: Indonesia</p>
	 <p>Waktu eksekusi: 246329 ms Tingkat kemiripan: 72.46%</p> <p>NIK: 59132508956259 Nama Alay: jn SMEr Nama: Ajibin Samosir Tempat lahir: Bengkulu Tanggal lahir: 31-5-1985 Jenis Kelamin: Laki-Laki Alamat: Gang Kendalsari No. 6 Padang, SU 78168 Agama: Katolik Status Pekerjaan: Cewek Pekerjaan: Commercial art gallery manager Kewarganegaraan: Indonesia</p>
	 <p>Waktu eksekusi: 267002 ms Tingkat kemiripan: 72.46%</p> <p>NIK: 59132508956259 Nama Alay: jn SMEr Nama: Ajibin Samosir Tempat lahir: Bengkulu Tanggal lahir: 31-5-1985 Jenis Kelamin: Laki-Laki Alamat: Gang Kendalsari No. 6 Padang, SU 78168 Agama: Katolik Status Pekerjaan: Cewek Pekerjaan: Commercial art gallery manager Kewarganegaraan: Indonesia</p>
	 <p>Waktu eksekusi: 4820 ns Tingkat kemiripan: null</p> <p>Image not found</p>



#### 4.4. Analisis Hasil Pengujian

Berdasarkan hasil yang didapatkan pada saat pengujian, ditemukan bahwa algoritma BM atau Boyer-Moore dapat memberikan hasil dalam waktu eksekusi yang lebih cepat. Salah satu alasan mengapa hal tersebut terjadi adalah karena jumlah alfabet yang relatif besar sehingga algoritma BM lebih diuntungkan dibanding algoritma KMP karena banyaknya lompatan dalam fase pencarian.

Untuk pencarian yang tidak dapat ditemukan dengan algoritma BM ataupun KMP, umumnya pada masukan sidik jari yang telah melewati alterasi, hasil pencarian yang diberikan sesuai dengan sidik jari sebelum dilakukan alterasi meski waktu eksekusi yang sangat lama akibat algoritma yang pada dasarnya merupakan algoritma *brute force* ditambahkan dengan perhitungan *Levenshtein distance* untuk setiap *substring*-nya.

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1. Kesimpulan dan Saran**

Dari percobaan dan pengujian yang telah dilakukan, algoritma KMP dan BM terbukti dapat mencocokkan dua buah sidik jari meskipun dalam beberapa kasus lebih baik digunakan *Levenshtein distance*. Waktu eksekusi juga cenderung lebih cepat dilakukan oleh BM daripada KMP karena perbandingan pada BM dilakukan dari belakang dan KMP lebih cocok ketika karakter lebih homogen (kurang beragam). Regex dapat digunakan untuk pencocokan *string* dengan pola tertentu dalam waktu yang cukup cepat

Saran kami adalah untuk menggunakan algoritma yang lebih cocok bergantung pada kasus dan pola teks yang tersedia.

#### **5.2. Tanggapan**

Tanggapan tugas besar ini cukup menarik karena mendorong untuk eksplorasi lebih jauh dan pembangunan aplikasi *desktop* dengan bahasa yang baru. Tubes ini pun menarik karena membuka pengetahuan mengenai pencocokan sidik jari dan Regex juga menarik karena dapat mencari string yang cukup berbeda dengan apa yang diinginkan. Saran kami ke depannya, semoga dibuat tubes yang lebih cocok dengan tingkat pemahaman yang sama.

#### **5.3. Refleksi**

Refleksi kami untuk tugas besar ini adalah untuk mengerjakan terlalu dekat dengan *deadline*.

## LAMPIRAN

Link repository dari Tugas Besar III IF Kelompok “S1T4j4111” adalah sebagai berikut :

[https://github.com/julianchandras/Tubes3\\_S1T4j4111](https://github.com/julianchandras/Tubes3_S1T4j4111)

Video Tugas Besar 3 IF Kelompok “S1T4j4111” adalah sebagai berikut :

[https://youtu.be/9D8k5Wj-UGc?si=NGeAb\\_jvL2WxZ-Ko](https://youtu.be/9D8k5Wj-UGc?si=NGeAb_jvL2WxZ-Ko)

## DAFTAR PUSTAKA

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf) (diakses pada 2 Maret 2024 pukul 13.38 WIB)

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/String-Matching-dengan-Regex-2019.pdf> (diakses pada 9 Juni 2024 pukul 12.25 WIB)