BAB 3
ANALISIS DAN PERANCANGAN SISTEM SISTEM

3.1 Analisis Sistem

Analisis sistem dilakukan dengan tujuan untuk mengidentifikasi dan mengevaluasi seluruh komponen yang terkait dengan sistem yang akan dibangun. Tahap analisis merupakan tahap yang kritis dan sangat penting, karena kesalahan pada tahap ini akan menyebabkan kesalahan pada tahap selanjutnya. Tahap analisis sistem dilakukan dengan cara menguraikan suatu sistem yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan sehingga ditemukan kelemahan dan keuntungan pada sistem tersebut. Dalam membangun perangkat lunak ini dilakukan beberapa analisis sebagai berikut.

3.1.1 Analisis Masalah

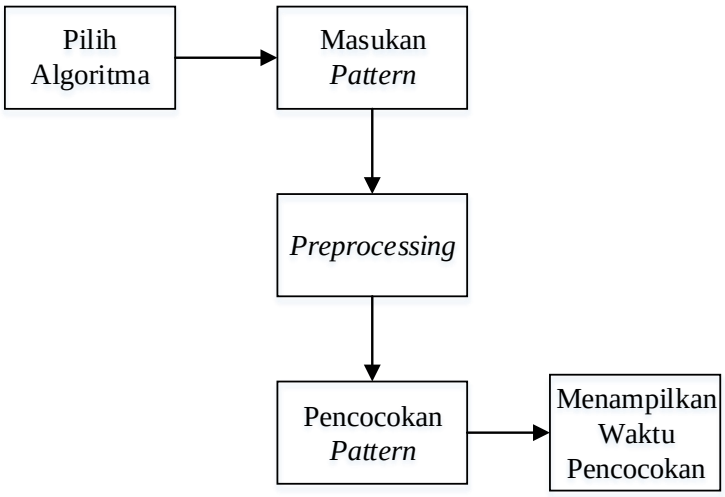
Berdasarkan latar belakang yang sudah diuraikan, maka masalah yang didapat adalah bagaimana mengetahui tingkat akursi dan waktu yang diperlukan algoritma *Zhu-Takaoka* dan algoritma *Reverse Colussi* dalam melakukan pencocokan *String* pada aplikasi Perbandingan Algoritma *Reverse Colussi* dan *Zhu-Takaoka* dalam satu kali proses.

3.1.2 Analisis Data Masukan

Program yang dibuat untuk pengujian yaitu aplikasi perbandingan Algoritma *Reverse Colussi* dan *Zhu Takaoka* yang akan digunakan untuk menguji waktu pencocokan kata. Pengujian kedua algoritma tersebut dilakukan secara bersamaan pada saat pencocokan kata. Dalam aplikasi tersebut, setiap algoritma akan menampilkan catatan waktu pencocokan kata yang digunakan dalam proses pencocokan *string*. Sumber teks yang digunakan sebagai data pengujian yaitu berupa sinopsis dari buku berbahasa Sunda, sedangkan *pattern* (pola) yang digunakan sebagai kata kunci dalam pencocokan kata diambil secara acak dari sumber teks dengan panjang pola yang telah ditentukan dalam batasan masalah penelitan ini.

Keluaran dari aplikasi tersebut adalah berupa catatan waktu yang digunakan oleh algoritma *Reverse Colussi* dan algoritma *Zhu Takaoka* dalam pencocokan string yang terdiri dari waktu *preprocessing* dan waktu pencarian. Selain menampilkan waktu proses, keluaran dari aplikasi tersebut juga menampilkan jumlah *pattern* yang ditemukan, serta waktu dari pengujian kasus terburuk (*worst case*) dan waktu dari pengujian kasus terbaik (*best case*) yang ditampilkan dalam bentuk grafik.

3.1.3 Deskripsi Umum



Gambar 3.1 Deskripsi Umum Pencocokan *String*

Secara umum terdapat beberapa tahap dalam memproses setiap kata sebelum melakukan pencocokan pada suatu kalimat, yaitu:

1. Tahap memilih algoritma

   Ketika akan melakukan pencocokan pada aplikasi perbandingan algoritma *Reverse Colussi* dan *Zhu-Takaoka,* langkah pertama adalah tentukan dahulu algoritma apa yang akan digunakan untuk melakukan pencocokan.
2. Tahap memasukan kata kunci atau *pattern*

   Pada tahap ini masukan *pattern* atau atau panjang *pattern* yang akan digunakan sebagai kata kunci untuk melakukan pencocokan.

3. Tahap *preprocessing*

   Pada tahap ini, kedua algoritma akan melakukan *preprocessing*, tahap ini bertujuan untuk membuat tabel *bad character* dan good *suffix*, kedua tabel tersebut akan digunakan sebagai jumlah pergeseran karakter pada saat pencocokan *string*.
4. Tahap pencocokan

   Tahap ini adalah tahap inti, yaitu proses dimana *pattern* (pola) yang menjadi kata kunci dicocokan dengan sumber teks yang sebelumnya sudah dimasukan ke dalam database. Setiap pergeseran dalam proses pencocokan diambil dari tabel *bad character* atau *good suffix*.
5. Tahap menampilkan hasil pencocokan

   Tahap ini adalah tahap terakhir dimana catatan waktu yang digunakan dalam proses pencocokan akan ditampilkan.

3.1.4 Analisis Metode

Analisis metode pada penelitian ini yaitu menganalisis akurasi dan waktu yang diperlukan oleh algoritma *Reverse Colussi* dan algoritma *Zhu-Takaoka* pada proses pencocokan kata yang diimplementasikan pada aplikasi Perbandingan Algoritma *Reverse Colussi* dan *Zhu-Takaoka*. Terdapat dua tipe analisis algoritma yaitu :

1. Kebenaran algoritma dapat dilakukan dengan cara berurutan, memeriksa bentuk logika, implementasi algoritma, pengujian algoritma dengan data, dan menggunakan cara matematika.
2. Penyederhanaan algoritma dilakukan dengan cara membagi algoritma menjadi bentuk yang lebih sederhana.

3.1.4.1 Analisis Kasus Algoritma

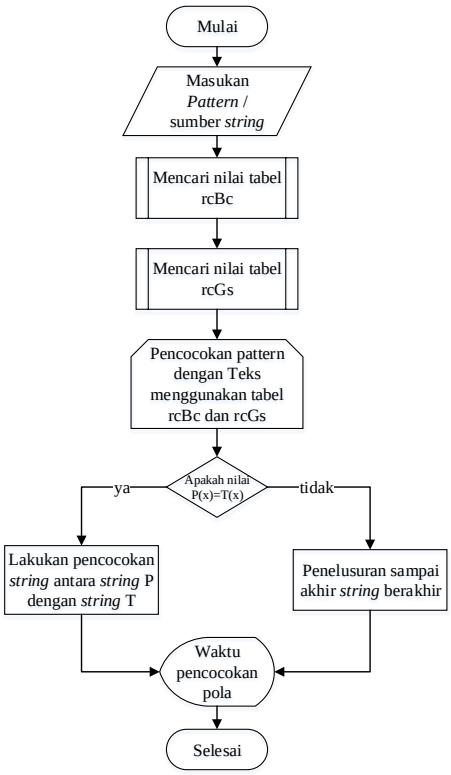
Analisis kasus algoritma merupakan tahap terpenting dalam pengujian algoritma, karena dari proses tersebut dapat diketahui cara kerja dari kedua algoritma dalam menyelesaikan suatu masalah.

Analisis kasus dapat dimulai dari menentukan semua kondisi yang mungkin atau dimulai dari menentukan variasi aksi. Berikut analisis kasus algoritma *Reverse Colussi* dan algoritma *Zhu-Takaoka*.

3.1.4.1.1 Algoritma Reverse Colussi

Algoritma *Reverse Colussi* merupakan pengembangan dari Algoritma *Boyer-Moore* yang mempunyai dua langkah awal yaitu langkah *preprocessing* dan pencocokan *pattern*. Tahap *preprocessing* dilakukan untuk mencari nilai tabel rcBc(*Reverse Colussi Bad Character*) dan rcGs(*Reverse Colussi Good Suffix*). Tabel rcBc digunakan pada saat melakukan pencocokan *pattern* dengan sumber teks, tapi ditemukan ketidakcocokan karakter pada saat pertama kali dibandingkan. Maka nilai pergeseran selanjutnya diambil dari nilai tabel rcBc. Sedangkan untuk tabel rcGs digunakan pada saat melakukan pencocokan *pattern* dengan sumber teks dan ditemukan kecocokan pada saat pertama kali dibandingkan. Maka nilai pergeseran selanjutnya akan diambil dari nilai tabel rcGs.

Untuk mencari nilai rcBc dilakukan dengan cara menentukan sepasangan karakter, hasilnya akan dimasukan ke dalam tabel rcBc. Sedangkan untuk mencari nilai rcGs dilakukan dengan cara menentukan nilai hmin dan nilai rmin pada tabel rcGs. Alur kerja algoritma *Reverse Colussi* dapat dilihat Pada Gambar 3.2.



Gambar 3.2 Flowchart Algoritma *Reverse Colussi*.

Sebagai langkah awal, pengguna akan memasukan kata kunci atau menentukan panjang pola yang akan dicocokan. Setelah kata kunci diinputkan langkah selanjutnya adalah melakukan *preprocessing* yaitu mencari nilai rcBc dari sumber teks dan nilai rcGs dari *pattern* yang ditentukan. Setelah nilai tabel rcBc dan rcGs terisi semua, langkah selanjutnya adalah melakukan pencocokan yaitu dengan cara membandingkan setiap karakter dari mulai karakter paling kanan. Apabila pada perbandingan pertama sudah tidak ada kecocokan, maka nilai perpindahan yang digunakan adalah nilai perpindahan dari table rcBc. Tetapi, jika pada perbandingan pertama ada kecocokan, maka nilai perpindahan selanjutnya bisa dilihat dari tabel rcGs. Proses perbandingan dilakukan sampai sisa sumber teks yang belum dibandingkan memiliki jumlah karakter kurang dari jumlah karakter pada *pattern*. Setelah proses perbandingan, langkah selanjutnya adalah menampilkan waktu yang telah digunakan dalam melakukan pencocokan oleh algoritma *Reverse Colussi*, baik pola itu ditemukan ataupun tidak ditemukan, dan proses pencocokan selesai.

3.1.4.1.1.1 Contoh Kasus

Sumber teks : **KI MASKARA BALAD PAJUANG TI PRIANGAN, BAPANA SUNJAYA ATAWA AKINA NYI RODIAH.**

Pattern : **SUNJAYA**

s = m = 7

1. Mencari rcBc.

Untuk mencari nilai rcBc maka carilah sepasang karakter pada *pattern*, proses tersebut akan dicari dengan menggunakan beberapa rumus, dengan variabel sebagai berikut.

Inisialisasi

Y adalah karakter terakhir dari window pada T.

s adalah panjang pergeseran yang akan digunakan pada langkah terakhir.

k bilangan integer yang terdapat pada *pattern*, ketika ditemukan karakter.

m panjang *pattern*.

Untuk memberikan nilai pada tabel rcBc ada beberapa aturan ketika sepasang karakter ditemukan ataupun tidak ditemukan, berikut aturan yang akan menentukan berapa nilai yang akan dimasukan kedalam tabel rcBc.

a. Kondisi 1:

Jika ditemukan P_{m-k-1} =Y and $P_{m-k-s-1}$=P_{m-s-1},

Masukan k terkecil ke tabel rcBc[Y, s]

b. **Kondisi 2:**

Jika ditemukan P_{m-k-1} =Y and k>m-s-1,

Masukan k terkecil ke tabel rcBc[Y, s]

c. **Kondisi 3:**

Jika tidak ditemukan, Masukan m ke tabel rcBc[Y, s]

Untuk menampung nilai rcBc yang akan dicari, maka dibuatlah tabel isian rcBc. Judul baris pada tabel isian rcBc diisi dengan melihat karakter yang terdapat pada sumber teks, lalu diurutkan sesuai abjad tak berulang. Untuk mengisi judul kolom, masukan angka sesuai panjang *pattern*. Maka akan terbentuk tabel seperti pada tabel 3.1.

Tabel 3.1 rcBc awal

	1	2	3	4	5	6	7
A							
B							
D							
G							
H							
I							
J							
K							
L							
M							
N							
O							
P							
R							
S							
T							
U							
W							
Y							
,							
.							
Sp							

Langkah pertama mencari nilai pada baris A kolom 1.

Maka, Y = A, s = 1, m = 7, pasangan karakter YA.

Y = A

T								A								

s = 1:

Pattern	S	U	N	J	A	Y	A

X = Y

k = 2

$P_{m-k-1} = P_{7-2-1} = P_4 = A$

$P_{m-k-s-1} = P_{7-2-1-1} = P_3 = J$

$P_{m-s-1} = P_{7-1-1} = P_5 = Y$

Karena jika dimasukan kedalam aturan yang ada sebelumnya tidak memenuhi syarat untuk kondisi 1 dan kondisi 2, maka baris A kolom 1 diisi dengan menggunakan kondisi 3. Masukan panjang m pada baris A kolom 1, sehingga akan terlihat seperti tabel 3.2.

Tabel 3.2 rcBc karkater A, 1

	1	2	3	4	5	6	7
A	**7**						
B							
D							
G							
H							
I							
J							
K							
L							
M							
N							
O							
P							
R							
S							
T							
U							
W							
Y							
,							
.							
Sp							

Kondisi yang sama juga ditemukan pada baris A kolom 2, 3, dan 4. Sehingga langsung saja masukan panjang m kedalam tabel tersebut. Seperti pada tabel 3.3. Langkah selanjutnya, mencari nilai pada baris A kolom 5.

Maka, Y = A, s = 5, m = 7, U _ _ _ _ A

Y = A

T															
											A				

s = 5:

Pattern

S	U	N	J	A	Y	A

X = U

k = 2

$P_{m-k-1} = P_{7-2-1} = P_4 = A$

$P_{m-k-s-1} = P_{7-2-5-1} = P_{-1}$

$P_{m-s-1} = P_{7-5-1} = P_1 = U$

Karena ditemukan $P_{m-k-1} = Y$ dan $k > m – s – 1$, sesuai dengan kondisi ke 2, sehingga untuk mengisi baris A kolom 5 menggunakan kondisi 2. Masukan panjang k terkecil pada baris A kolom 5, sehingga akan terlihat seperti tabel 3.3.

Tabel 3.3 rcBc karkater A, 5

	1	2	3	4	5	6	7
A	7	7	7	7	**2**		
B							
D							
G							
H							
I							
J							
K							
L							
M							
N							
O							
P							
R							
S							
T							
U							
W							
Y							
,							
.							
Sp							

Kondisi yang sama juga ditemukan pada baris A kolom 6 dan 7. Sehingga langsung masukan panjang k terkecil kedalam tabel tersebut. Seperti pada tabel 3.4. Langkah selanjutnya, mencari nilai pada baris B kolom 1.

Maka, Y = B, s = 1, m = 7, YB, tidak ada nilai k, karena karakter B tidak ada pada *pattern*.

Y = B

T								B								

s = 1:

Pattern	S	U	N	J	A	Y	A

X = Y

Karena tidak ditemukan nilai k, maka sudah dapat dipastikan tidak memenuhi syarat untuk kondisi 1 dan kondisi 2. Sehingga untuk mengisi baris B kolom 1 menggunakan kondisi 3. Masukan panjang m pada baris B kolom 1, sehingga akan terlihat seperti tabel 3.4.

Tabel 3.4 rcBc karkater B, 1

	1	2	3	4	5	6	7
A	7	7	7	7	2	2	2
B	**7**						
D							
G							
H							
I							
J							
K							
L							
M							
N							
O							
P							
R							
S							
T							
U							
W							
Y							
,							
.							
Sp							

Kondisi yang sama juga ditemukan pada baris B, D, G, H, I, K, L, M, O, P, R, T, W, [,], [.], Sp pada semua kolom. Sehingga langsung masukan panjang m kedalam tabel tersebut. Seperti pada tabel 3.5.

Langkah selanjutnya mencari nilai baris J kolom 1.

Maka, Y = J, s = 1, m = 7, YJ.

Y = J

T								J								

s = 1:

Pattern	S	U	N	J	A	Y	A

X = Y

k = 3

$P_{m-k-1} = P_{7-3-1} = P_3 = J$
$P_{m-k-s-1} = P_{7-3-1-1} = P_2 = N$
$P_{m-s-1} = P_{7-1-1} = P_5 = Y$

Karena Jika dimasukan kedalam aturan yang ada sebelumnya tidak memenuhi syarat untuk kondisi 1 dan kondisi 2, maka untuk mengisi baris J kolom 1 menggunakan kondisi 3. Masukan panjang m pada baris J kolom 1, sehingga akan terlihat seperti tabel 3.5.

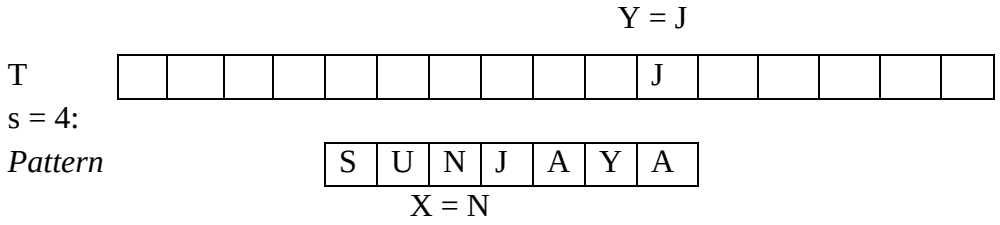
Tabel 3.5 rcBc karkater J, 1

	1	2	3	4	5	6	7
A	7	7	7	7	2	2	2
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**						
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N							
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S							
T	7	7	7	7	7	7	7
U							
W	7	7	7	7	7	7	7
Y							
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Kondisi yang sama juga ditemukan pada baris J kolom 2 dan 3. Maka langsung masukan panjang m kedalam tabel tersebut. Seperti pada tabel 3.6.

Langkah selanjutnya mencari nilai baris J kolom 4.

Maka, Y = J, s = 4, m = 7, N _ _ _ J.



k = 3

$P_{m-k-1} = P_{7-3-1} = P_3 = J$

$P_{m-k-s-1} = P_{7-3-4-1} = P_{-1}$

$P_{m-s-1} = P_{7-4-1} = P_2 = N$

Karena ditemukan $P_{m-k-1} = Y$ dan $k > m - s - 1$, sesuai dengan kondisi ke 2, sehingga untuk mengisi baris J kolom 4 menggunakan kondisi 2. Masukan panjang k terkecil pada baris J kolom 4, sehingga akan terlihat seperti tabel 3.6.

Tabel 3.6 rcBc karkater J, 4

	1	2	3	4	5	6	7
A	**7**	**7**	**7**	**7**	**2**	**2**	**2**
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**			
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N							
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S							
T	7	7	7	7	7	7	7
U							
W	7	7	7	7	7	7	7
Y							
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Kondisi yang sama ditemukan pada baris J kolom 5, 6 dan 7. Maka langsung masukan panjang k terkecil kedalam tabel tersebut. Seperti pada tabel 3.7.

Langkah selanjutnya mencari nilai baris N kolom 1.

Maka, Y = N, s = 1, m = 7, YN.

Y = N

T								N								

s = 1:

Pattern

S	U	N	J	A	Y	A

X = Y

k = 4

$P_{m-k-1} = P_{7-4-1} = P_2 = N$

$P_{m-k-s-1} = P_{7-4-1-1} = P_1 = U$

$P_{m-s-1} = P_{7-1-1} = P_5 = Y$

Karena Jika dimasukan kedalam aturan yang ada sebelumnya tidak memenuhi syarat untuk kondisi 1 dan kondisi 2, maka untuk mengisi baris N kolom 1 menggunakan kondisi 3. Masukan panjang m pada baris N kolom 1, sehingga akan terlihat seperti tabel 3.7.

Tabel 3.7 rcBc karkater N, 1

	1	2	3	4	5	6	7
A	7	7	7	7	2	2	2
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**	**3**	**3**	**3**
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N	**7**						
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S							
T	7	7	7	7	7	7	7
U							
W	7	7	7	7	7	7	7
Y							
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Kondisi yang sama ditemukan pada baris N kolom 2. Sehingga langsung masukan panjang m kedalam tabel tersebut. Seperti pada tabel 3.8.

Langkah selanjutnya mencari nilai baris N kolom 3.

Maka, Y = N, s = 3, m = 7, J _ _ N.

Y = N

T										N						
s = 3:																
Pattern	S	U	N	J	A	Y	A									

X = N

k = 4

$P_{m-k-1} = P_{7-4-1} = P_2 = N$

$P_{m-k-s-1} = P_{7-4-3-1} = P_{-1}$

$P_{m-s-1} = P_{7-3-1} = P_3 = J$

Karena ditemukan $P_{m-k-1} = Y$ dan $k > m – s – 1$, sesuai dengan kondisi ke 2, sehingga untuk mengisi baris N kolom 3 menggunakan kondisi 2. Masukan panjang k terkecil pada baris N kolom 3, sehingga akan terlihat seperti tabel 3.8.

Tabel 3.8 rcBc karkater N, 3

	1	2	3	4	5	6	7
A	**7**	**7**	**7**	**7**	**2**	**2**	**2**
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**	**3**	**3**	**3**
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N	**7**	**7**	**4**				
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S							
T	7	7	7	7	7	7	7
U							
W	7	7	7	7	7	7	7
Y							
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Kondisi yang sama juga ditemukan pada baris N kolom 4, 5, 6 dan 7. Sehingga langsung masukan panjang k terkecil kedalam tabel tersebut, seperti pada tabel 3.9.

Langkah selanjutnya mencari nilai baris S kolom 1.

Maka, Y = S, s = 1, m = 7, YS.

Y = S

T								N								

s = 1:

Pattern	S	U	N	J	A	Y	A

X = Y

k = 6

$P_{m-k-1} = P_{7-6-1} = P_0 = S$

$P_{m-k-s-1} = P_{7-6-1-1} = P_{-1}$

$P_{m-s-1} = P_{7-1-1} = P_5 = Y$

Karena ditemukan $P_{m-k-1} = Y$ dan $k > m – s – 1$, sesuai dengan kondisi ke 2, sehingga untuk mengisi baris S kolom 1 menggunakan kondisi 2. Masukan panjang k terkecil pada baris S kolom 1, sehingga akan terlihat seperti tabel 3.9.

Tabel 3.9 rcBc karkater S, 1

	1	2	3	4	5	6	7
A	**7**	**7**	**7**	**7**	**2**	**2**	**2**
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**	**3**	**3**	**3**
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N	**7**	**7**	**4**	**4**	**4**	**4**	**4**
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S	**6**						
T	7	7	7	7	7	7	7
U							
W	7	7	7	7	7	7	7
Y							
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Kondisi yang sama juga ditemukan pada baris S kolom 2, 3, 4, 5, 6 dan 7. Sehingga langsung masukan panjang k terkecil kedalam tabel tersebut, seperti pada tabel 3.10.

Langkah selanjutnya mencari nilai barisU kolom1.

Maka, Y = U, s = 1, m = 7, YU.

Y = U

T								U								

s = 1:

Pattern

S	U	N	J	A	Y	A

X = Y

k = 5

$P_{m-k-1} = P_{7-5-1} = P_1 = U$

$P_{m-k-s-1} = P_{7-5-1-1} = P_0 = S$

$P_{m-s-1} = P_{7-1-1} = P_5 = Y$

Karena Jika dimasukan kedalam aturan yang ada sebelumnya tidak memenuhi syarat untuk kondisi 1 dan kondisi 2, maka untuk mengisi baris U kolom 1 menggunakan kondisi 3. Masukan panjang m pada baris U kolom 1, sehingga akan terlihat seperti tabel 3.10.

Tabel 3.10 rcBc karkater U, 1

	1	2	3	4	5	6	7
A	**7**	**7**	**7**	**7**	**2**	**2**	**2**
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**	**3**	**3**	**3**
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N	**7**	**7**	**4**	**4**	**4**	**4**	**4**
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S	**6**	**6**	**6**	**6**	**6**	**6**	**6**
T	7	7	7	7	7	7	7
U	**7**						
W	7	7	7	7	7	7	7
Y							
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Langkah selanjutnya mencari nilai baris U kolom 1.

Maka, Y = U, s = 2, m = 7, A _ U.

Y = U

T									U							

s = 1:

Pattern	S	U	N	J	A	Y	A

X = A

k = 5

$P_{m-k-1} = P_{7-5-1} = P_1 = U$

$P_{m-k-s-1} = P_{7-5-2-1} = P\text{-}1 = S$

$P_{m-s-1} = P_{7-2-1} = P_4 = A$

Karena ditemukan $P_{m-k-1} = Y$ dan $k > m – s – 1$, sesuai dengan kondisi ke 2, maka untuk mengisi baris U kolom 2 menggunakan kondisi 2. Masukan k terkecil pada baris U kolom 2, sehingga akan terlihat seperti tabel 3.11.

Tabel 3.11 rcBc karkater U, 2

	1	2	3	4	5	6	7
A	**7**	**7**	**7**	**7**	**2**	**2**	**2**
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**	**3**	**3**	**3**
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N	**7**	**7**	**4**	**4**	**4**	**4**	**4**
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S	**6**	**6**	**6**	**6**	**6**	**6**	**6**
T	7	7	7	7	7	7	7
U	**7**	**5**					
W	7	7	7	7	7	7	7
Y							
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Kondisi yang sama juga ditemukan pada baris U kolom 3, 4, 5, 6 dan 7. Sehingga langsung masukan panjang k terkecil kedalam tabel tersebut, seperti pada tabel 3.12.

Langkah selanjutnya mencari nilai baris Y kolom 1.

Maka, Y = Y, s = 1, m = 7, YY.

Y = Y

T								Y								

s = 1:

Pattern

S	U	N	J	A	Y	A

X = Y

k = 1

$P_{m-k-1} = P_{7-1-1} = P_5 = Y$

$P_{m-k-s-1} = P_{7-1-1-1} = P_4 = A$

$P_{m-s-1} = P_{7-1-1} = P_5 = Y$

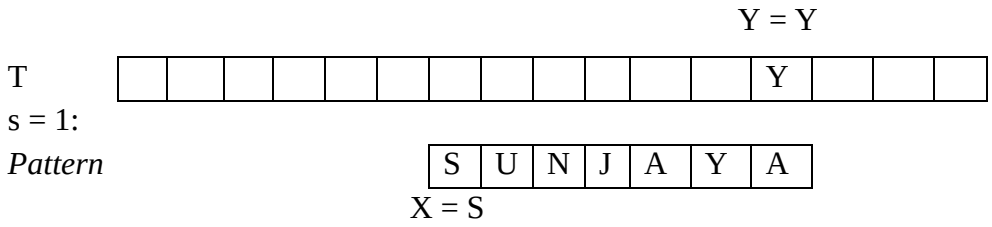
Karena Jika dimasukan kedalam aturan yang ada sebelumnya tidak memenuhi syarat untuk kondisi 1 dan kondisi 2, maka untuk mengisi baris Y kolom 1 menggunakan kondisi 3. Masukan panjang m pada baris Y kolom 1, sehingga akan terlihat seperti tabel 3.12.

Tabel 3.12 rcBc karkater Y, 1

	1	2	3	4	5	6	7
A	**7**	**7**	**7**	**7**	**2**	**2**	**2**
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**	**3**	**3**	**3**
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N	**7**	**7**	**4**	**4**	**4**	**4**	**4**
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S	**6**	**6**	**6**	**6**	**6**	**6**	**6**
T	7	7	7	7	7	7	7
U	**7**	**5**	**5**	**5**	**5**	**5**	**5**
W	7	7	7	7	7	7	7
Y	**7**						
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Kondisi yang sama juga ditemukan pada baris Y kolom 2, 3, 4 dan 5. Maka langsung masukan panjang m kedalam tabel tersebut. Seperti pada tabel 3.13. Langkah selanjutnya mencari nilai baris Y kolom 1.

Maka, Y = Y, s = 6, m = 7, S _ _ _ _ _ Y.



k = 1

$P_{m-k-1} = P_{7-1-1} = P_5 = Y$

$P_{m-k-s-1} = P_{7-1-6-1} = P_{-1}$

$P_{m-s-1} = P_{7-6-1} = P_0 = S$

Karena ditemukan $P_{m-k-1} = Y$ dan k > m – s – 1, sesuai dengan kondisi ke 2, maka untuk mengisi baris Y kolom 6 menggunakan kondisi 2. Masukan k terkecil pada baris Y kolom 6, sehingga akan terlihat seperti tabel 3.13.

Tabel 3.13 rcBc karkater Y, 6

	1	2	3	4	5	6	7
A	**7**	**7**	**7**	**7**	**2**	**2**	**2**
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**	**3**	**3**	**3**
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N	**7**	**7**	**4**	**4**	**4**	**4**	**4**
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S	**6**	**6**	**6**	**6**	**6**	**6**	**6**
T	7	7	7	7	7	7	7
U	**7**	**5**	**5**	**5**	**5**	**5**	**5**
W	7	7	7	7	7	7	7
Y	**7**	**7**	**7**	**7**	**7**	**1**	
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

Kondisi yang sama ditemukan pada baris Y kolom 7. Sehingga langsung masukan panjang k terkecil kedalam tabel tersebut, seperti pada tabel 3.14. Sehingga tabel rcBc terisi semua seperti pada Tabel 3.14 rcBc.

Tabel 3.14 rcBc

	1	2	3	4	5	6	7
A	**7**	**7**	**7**	**7**	**2**	**2**	**2**
B	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7
J	**7**	**7**	**7**	**3**	**3**	**3**	**3**
K	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7
N	**7**	**7**	**4**	**4**	**4**	**4**	**4**
O	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7
S	**6**	**6**	**6**	**6**	**6**	**6**	**6**
T	7	7	7	7	7	7	7
U	**7**	**5**	**5**	**5**	**5**	**5**	**5**
W	7	7	7	7	7	7	7
Y	**7**	**7**	**7**	**7**	**7**	**1**	**1**
,	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7

2. Mencari *special positions* dan *non-special positions*

Suatu *substring* bisa dikatakan *special position* jika ada salah satu atau beberapa karakter yang muncul secara berulang. Menentukan *special position* berguna untuk mendapatkan nilai hmin pada *pattern*.

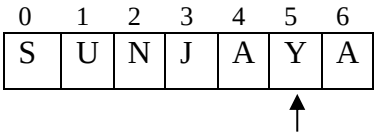
a. Mencari hmin

Nilai hmin digunakan untuk menentukan nilai pada tabel rcGs. Tabel 3.15 menunjukan tabel awal untuk melakukan pencocokan nilai hmin.

Tabel 3.15 hmin awal

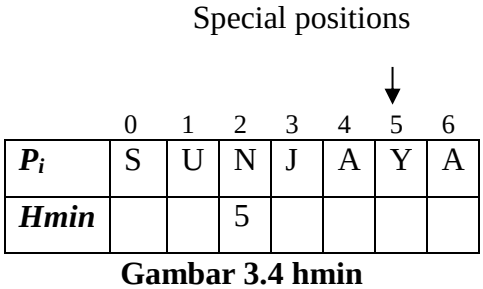
Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Pattern	S	U	N	J	A	Y	A
Hmin	?	?	?	?	?	?	?

Dari *pattern* pada Tabel 3.15, karakter A merupakan *substring* yang sering diulang. Sehingga Y adalah *special position*. Gambar 3.3 adalah gambar *special position* dari *pattern* tersebut.



Gambar 3.3 Special Position

Jumlah pergeseran untuk setiap *special positions* disimpan dalam tabel yang disebut hmin. pergeseran tersebut, dihitung dari jarak karakter yang berulang.



Gambar 3.4 hmin

Special position i = 5, simpan jumlah pergeseran 2 (7 – 4 – 1 = 2) pada tabel *hmin*[2] = 5

Tabel 3.16 hmin akhir

Index	0	1	2	3	4	5	6
Pattern	S	U	N	J	A	Y	A
Hmin			5				

Tabel 3.16 hmin akhir adalah hasil akhir dari pencocokan hmin pada *pattern* yang sedang dicari.

b. Mencari rmin

Nilai rmin digunakan untuk menentukan nilai tabel rcGs berikutnya. Tabel 3.17 merupakan tabel awal untuk pencocokan nilai rmin.

Tabel 3.17 rmin awal

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Pattern	S	U	N	J	A	Y	A
Hmin			5				
Rmin							

Rmin dapat diisi dengan ketentuan sebagai berikut: Jika *suffix* S yang terdapat pada sisi kanan dari *nonspecial positions* ke-i sama dengan *prefix* maka *rmin(i) = m*-|*S*|. (|S| panjang S), tetapi jika tidak ada S maka *rmin(i)=m*.

	S	U	N	J	A	Y	A
Rmin	6	6	6	6	6		

S

Gambar 3.5 rmin

Suffix S sama dengan *prefix* yang ada di sisi kanan dari *non-special positions*, jadi nilai dari *rmin non-special positions* adalah *m*-|*S*| *(7-1).* Dapat dilihat Seperti tabel 3.18.

Tabel 3.18 rmin akhir

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Pattern	S	U	N	J	A	Y	A
Hmin			5				
Rmin	6	6	6	6	6		

Setelah tabel *hmin* dan *rmin* dibuat, selanjutnya kedua tabel tersebut dapat digunakan untuk membuat tabel rcGs, seperti pada tebel 3.19.

Tabel 3.19 rcGs tahap 1

I	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Pi	S	U	N	J	A	Y	A	
hmin[i]			5					
rmin[i]	6	6	6	6	6	6		
rcGs[i]	0							

Pertama-tama, isi indeks dari *special positions* yang tidak kosong kedalam table rcGs, maka akan menghasilkan *value* seperti pada table 3.20.

Tabel 3.20 rcGs tahap 2

I	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Pi	S	U	N	J	A	Y	A	
hmin[i]			5					
rmin[i]	6	6	6	6	6	6		
rcGs[i]	0	2						

Kemudian isikan semua nilai dari tabel *rmin* yang tidak kosong kedalam tabel rcGs.

Tabel 3.21 rcGs tahap 3

I	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Pi	S	U	N	J	A	Y	A	
hmin[i]			5					
rmin[i]	6	6	6	6	6	6		
rcGs[i]	0	2	6	6	6	6	6	

Jika *P exact match* dengan *T*, maka pergeseran *P* menjadi *rcGs*[7]=*m*-|*S*| (7-1).

Tabel 3.22 rcGs

I	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Pi	S	U	N	J	A	Y	A	
hmin[i]			5					
rmin[i]	6	6	6	6	6	6		
rcGs[i]	0	2	6	6	6	6	6	6

3. Fase pencocokan *pattern*

Setelah proses *preprocessing* selesai, dengan hasil yang dapat dilihat pada Tabel 3.14 untuk tabel rcBc dan pada Tabel 3.22 untuk tabel rcGs, langkah selanjutnnya adalah proses pencocokan. Proses pencocokan dengan contoh sumber *string* seperti berikut.

Teks : **KI MASKARA BALAD PAJUANG TI PRIANGAN, BAPANA SUNJAYA ATAWA AKINA NYI RODIAH.**

Pattern : **SUNJAYA**

Percobaan ke-1

I	0	1	2	3	4	5	**6**	7	8	9	10	11	12	13	14	15	16	17	18	...
Teks	K	I		M	A	S	**K**	A	R	A		B	A	L	A	D		P	A	...
							1													
Pattern	S	U	N	J	A	N	**A**													

Shift by 7(*rcBc*[K][*s*], *s* = 7), and change s = 7

Dari hasil percobaan ke-1, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter K pada teks. Artinya, pada percobaan ke-1 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh tujuh karakter. Nilai pergeseran tujuh karakter ini diperoleh dari tabel rcBc K, 7 dan bernilai tujuh.

Percobaan ke-2

I	0	1	2	3	4	5	6	7	8	9	10	11	12	**13**	14	15	16	17	18	...
Teks	K	I		M	A	S	K	A	R	A		B	A	**L**	A	D		P	A	...
														1						
Pattern								S	U	N	J	A	Y	**A**						

Shift by 7 (*rcBc*[L][*s*], *s* = 7), and change s = 7

Dari hasil percobaan ke-2, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter L pada teks. Artinya, pada percobaan ke-2 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh tujuh karakter. Nilai pergeseran tujuh karakter ini diperoleh dari tabel rcBc L, 7 dan bernilai tujuh.

Percobaan ke-3

I	14	15	16	17	18	19	**20**	21	22	23	24	25	26	27	28	29	30	31	...
Teks	A	D		P	A	J	**U**	A	N	G		T	I		P	R	I	A	...
							1												
Pattern	S	U	N	J	A	Y	**A**												...

Shift by 5 (*rcBc*[U][*s*], *s* = 7), and change s = 5

Dari hasil percobaan ke-3, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter U pada teks. Artinya, pada percobaan ke-3 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh lima karakter. Nilai pergeseran lima karakter ini diperoleh dari tabel rcBc U, 7 dan bernilai lima.

Percobaan ke-4

I	14	15	16	17	18	19	20	21	22	23	24	**25**	26	27	28	29	30	31	...
Teks	A	D		P	A	J	U	A	N	G		**T**	I		P	R	I	A	...
												1							
Pattern						S	U	N	J	A	Y	**A**							...

Shift by 7 (*rcBc*[T][*s*], *s* = 5), and change s = 7

Dari hasil percobaan ke-4, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter T pada teks. Artinya, pada percobaan ke-4 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh tujuh karakter. Nilai pergeseran tujuh karakter ini diperoleh dari tabel rcBc T, 5 dan bernilai tujuh.

Percobaan ke-5

I	26	27	28	29	30	31	**32**	33	34	35	36	37	38	39	40	41	42	43	...
Teks	I		P	R	I	A	**N**	G	A	N	,		B	A	P	A	N	A	...
							1												
Pattern	S	U	N	J	A	Y	**A**												...

Shift by 4 (*rcBc*[N][*s*], *s* = 7), and change s = 4

Dari hasil percobaan ke-5, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter N pada teks. Artinya, pada percobaan ke-5 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh empat

karakter. Nilai pergeseran empat karakter ini diperoleh dari tabel rcBc N, 7 dan bernilai 4.

Percobaan ke-6

I	26	27	28	29	30	31	32	33	34	35	**36**	37	38	39	40	41	42	43	...
Teks	I		P	R	I	A	N	G	A	N	,		B	A	P	A	N	A	...
											1								
Pattern					S	U	N	J	A	Y	**A**								...

Shift by 7 (*rcBc*[,][*s*], *s* = 4), and change s = 7

Dari hasil percobaan ke-6, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter [,] pada teks. Artinya, pada percobaan ke-6 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh tujuh karakter. Nilai pergeseran tujuh karakter ini diperoleh dari tabel rcBc [,], 4 dan bernilai 7.

Percobaan ke-7

I	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	**43**	...
Teks	I		P	R	I	A	N	G	A	N	,		B	A	P	A	N	**A**	...
																		1	
Pattern												S	U	N	J	A	Y	**A**	...

Shift by 2 (*rcGs*[1]), and change s = 2

Dari hasil percobaan ke-7, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter A pada teks artinya pada percobaan ke-7 terjadi kecocokan, maka dilakukan pergeseran selanjutnya sejauh dua karakter. Nilai pergeseran dua karakter ini diperoleh dari tabel rcGs[1] dan bernilai 2.

Percobaan ke-8

I	39	40	41	42	43	44	**45**	46	47	48	49	50	51	52	53	54	55	56	...
Teks	A	P	A	N	A		**S**	U	N	J	A	Y	A		A	T	A	W	...
							1												
Pattern	S	U	N	J	A	Y	**A**												...

Shift by 6 (*rcBc*[S][s], s = 2), and change s = 6

Dari hasil percobaan ke-8, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter S pada teks. Artinya, pada percobaan ke-8 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh enam

karakter. Nilai pergeseran enam karakter ini diperoleh dari tabel rcBc S, 2 dan bernilai 6.

Percobaan ke-9

I	39	40	41	42	43	44	**45**	**46**	**47**	**48**	**49**	**50**	**51**	52	53	54	55	56	...
Teks	A	P	A	N	A		**S**	**U**	**N**	**J**	**A**	**Y**	**A**		A	T	A	W	...
Pattern							**S**	**U**	**N**	**J**	**A**	**Y**	**A**						...

Shift by 6 (*rcGs*[7]), and change s = 6

Dari hasil percobaan ke-9, dapat dilihat bahwa semua karakter terjadi kecocokan, maka dilakukan pergeseran selanjutnya sejauh enam karakter. Nilai pergeseran enam karakter ini diperoleh dari tabel rcGs[7] dan bernilai 6.

Percobaan ke-10

I	51	52	53	54	55	56	**57**	58	59	60	61	62	63	64	65	66	67	68	...
Teks	A		A	T	A	W	**A**		A	K	I	N	A		N	Y	I		...
							1												
Pattern	S	U	N	J	A	Y	**A**												...

Shift by 2 (*rcGs*[1]), and change s = 2

Dari hasil percobaan ke-10, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter A pada teks. Artinya, pada percobaan ke-10 terjadi kecocokan, maka dilakukan pergeseran selanjutnya sejauh dua karakter. Nilai pergeseran dua karakter diperoleh dari tabel rcGs[1] dan bernilai 2.

Percobaan ke-11

I	51	52	53	54	55	56	57	58	**59**	60	61	62	63	64	65	66	67	68	...
Teks	A		A	T	A	W	A		**A**	K	I	N	A		N	Y	I		...
									1										
Pattern			S	U	N	J	A	Y	**A**										...

Shift by 2 (*rcGs*[1]), and change s = 2

Dari hasil percobaan ke-11, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter A pada teks. Artinya, pada percobaan ke-11 terjadi kecocokan, maka dilakukan pergeseran selanjutnya sejauh dua karakter. Nilai pergeseran dua karakter diperoleh dari tabel rcGs[1] dan bernilai 2.

Percobaan ke-12

I	51	52	53	54	55	56	57	58	59	60	**61**	62	63	64	65	66	67	68	...
Teks	A		A	T	A	W	A		A	K	**I**	N	A		N	Y	I		...
											1								
Pattern					S	U	N	J	A	Y	**A**								...

Shift by 7 (*rcBc*[I][s], s = 2), and change s = 7

Dari hasil percobaan ke-12, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter I pada teks. Artinya, pada percobaan ke-12 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh tujuh karakter. Nilai pergeseran tujuh karakter ini diperoleh dari tabel rcBc I, 2 dan bernilai 7.

Percobaan ke-13

I	69	70	71	72	73	74	**75**
Teks	R	O	D	I	A	H	**.**
							1
Pattern	S	U	N	J	A	Y	**A**

Shift by 7 (*rcBc*[Sp][s], s = 7), and change s = 7

Dari hasil percobaan ke-13, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter [.] pada teks. Artinya, pada percobaan ke-13 terjadi ketidakcocokan, maka dilakukan pergeseran selanjutnya sejauh tujuh karakter. Nilai pergeseran tujuh karakter ini diperoleh dari tabel rcBc [.], 7 dan bernilai 7.

Karena karakter [.] pada indeks ke 75 merupakan karakter akhir dari Teks, maka pencocokan selesai.

Kesimpulan

1. Pada tahap pencocokan dengan algoritma *Reverse Colussi* di atas, pencocokan dilakukan sebanyak 13 percobaan. Ditemukan pada percobaan ke-9.
2. Jumlah indeks pada teks adalah 75, *Pattern* ditemukan pada indeks ke 45-51.
3. Jumlah perbandingan adalah 62, *pattern* ditemukan pada perbandingan ke 45.

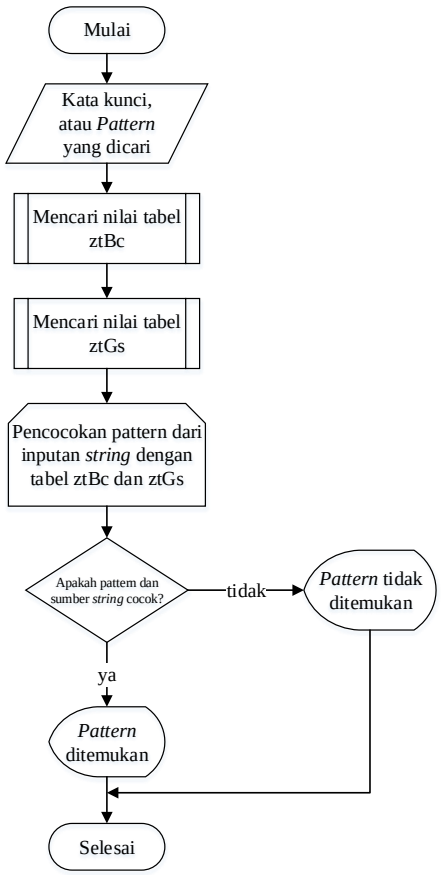
3.1.4.1.2 Algoritma Zhu-Takaoka

Algoritma *Zhu-Takaoka* (Algoritma BM") yang merupakan pengembangan dari Algoritma *Boyer-Moore* mempunyai ciri-ciri yang sama dalam proses pencocokan *string*. Ciri-ciri tersebut terbagi menjadi dua fase, yaitu fase *preprocessing* dan fase pencocokan. Perbedaan antara Algoritma *Boyer-Moore* dan Algoritma *Zhu-Takaoka* yaitu terletak pada tahap penentuan *bad character rule*. Dalam *Boyer-Moore*, bad charater hanya terdiri *array* satu dimensi, sedangkan dalam *Zhu-Takaoka* dikembangkan menjadi *array* dua dimensi. Karakteristik algoritma *Zhu-Takaoka*, diantaranya:

1. Pengembangan dari algoritma *Boyer-Moore.*
2. Menggunakan *array* dua dimensi untuk menghitung nilai pergeseran.
3. Melakukan pencocokan dari kanan ke kiri.

Tahap *preprocessing* dilakukan untuk menentukan nilai tabel ztBc(*Zhu-Takaoka Bad Character*) dan ztGs(*Zhu-Takaoka Good Suffix*). Tabel ztBc digunakan pada saat melakukan pencocokan *pattern* dengan sumber teks dan ditemukan ketidakcocokan karakter pada saat pertama kali dibandingkan, maka nilai pergeseran selanjutnya diambil dari nilai tabel ztBc. Sedangkan untuk tabel ztGs digunakan pada saat melakukan pencocokan *pattern* dengan sumber teks dan ditemukan kecocokan pada saat pertama kali dibandingkan, maka nilai pergeseran selanjutnya diambil dari nilai tabel ztGs.

Pada Gambar 3.2 Flowchart Algoritma *Zhu-Takaoka* dapat dilihat alur kerja algoritma Algoritma *Zhu-Takaoka*.



Gambar 3.6 Flowchart Algoritma *Zhu-Takaoka*.

Sebagai langkah awal, pengguna akan memasukan panjang *pattern* atau kata yang dicocokan. Setelah kata kunci atau panjang *pattern* dimasukan, langkah selanjutnya adalah melakukan *preprocessing* yaitu mencari nilai ztBc dan ztGs dari *pattern* yang telah dimasukan. Setelah nilai tabel ztBc dan ztGs terisi semua, langkah selanjutnya adalah melakukan pencocokan, yaitu dengan cara membandingkan setiap karakter dari mulai karakter paling kanan. Apabila pada perbandingan pertama sudah tidak ada kecocokan, maka nilai perpindahan yang digunakan adalah nilai perpindahan dari table ztBc. Tetapi jika pada perbandingan pertama ada kecocokan, nilai perpindahan selanjutnya bisa dilihat dari tabel ztGs. Proses perbandingan dilakukan sampai sisa sumber teks yang belum dibandingkan memiliki jumlah karakter kurang dari jumlah karakter pada *pattern*. Setelah proses perbandingan, langkah selanjutnya adalah menampilkan waktu yang telah digunakan dalam melakukan pencocokan oleh algoritma *Zhu Takaoka*, baik pola itu ditemukan ataupun tidak ditemukan, dan proses pencocokan selesai.

3.1.4.1.2.1 Contoh Kasus

Teks : **KI MASKARA BALAD PAJUANG TI PRIANGAN, BAPANA SUNJAYA ATAWA AKINA NYI RODIAH.**

Pattern : **SUNJAYA**

Seperti yang sudah dijelaskan sebelumnya, langkah pertama adalah menentukan tabel ztBc(*Zhu-Takaoka Bad Character*) dan tabel ztGs(*Zhu-Takaoka Good Suffix*) kemudian melakukan proses pencocokan.

1. Membuat tabel ztBc

Untuk memudahkan proses mencari tabel ztBc langkah awal adalah dengan menemukan tabel bmBc, yaitu dengan mengidentifikasi setiap perpindahan. Tabel 3.23 adalah tabel bmBc sebelum diisi.

Tabel 3.23 bmBc awal

Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Nilai OH	?	?	?	?	?	?	?

Penjelasan

a. ***Index*** adalah indeks dari *pattern*.
b. **Karakter** adalah *pattern* yang dicari.
c. **Nilai OH(*Occurrence Heuristic*)** adalah nilai pergeseran *Bad Character*. Nilai OH nantinya akan digunakan ketika ditemukan ketidakcocokan pada saat pertama kali mencocokan *string*.

Untuk mengisi Nilai OH, langkah pertama adalah dengan melakukan pencacahan di setiap karakter pada *pattern*. Pencacahan dimulai dari karakter ke-2 paling kanan (*Index = Length* (*pattern*) - 2) dari *pattern*. Untuk langkah pertama perpindahan diisi dengan angka 1. Seperti pada Tabel 3.24.

Tabel 3.24 Mencari Nilai OH (Pergeseran ke-1)

Mulai : *Index = Length*(*pattern*) – 2 = 7 – 2 = 5							
Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Pindah = 1							

Bandingkan setiap karakter yang dicacah terhadap karakter yang terdapat pada tabel BmBc sebelumnya, jika karakter yang dicacah tidak ditemukan dalam tabel tersebut, maka tambahkan karakter tersebut ke dalam BmBc dan Nilai OH (*Occurrence Heuristic*) diisi dengan jumlah pindah karakter yang sedang dibandingkan. Untuk perbandingan pertama, bandingkan karakter pertama dengan nilai *Null*. Akan seperti Tabel 3.25.

Tabel 3.25 Mencari Nilai OH (Perbandingan ke-1)

Bandingkan Y dengan *Null*		Hasil	
BmBc(awal)		BmBc(akhir)	
Karakter	*Null*	**Karakter**	Y
Nilai OH	*Null*	**Nilai OH**	1

Pada pencacahan karakter kedua, dilakukan pada *Index* sebelumnya dikurangi 1, yaitu pada *Index* 4. Dan jumlah perpindahan diubah menjadi 2. Seperti Tabel 3.26.

Tabel 3.26 Mencari Nilai OH (Pergeseran ke-2)

Mulai : *Index* = 5 – 1 = 4							
Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Pindah = 2							

Pada perbandingan kedua, bandingkan karakter A yang ada pada *Index* 4 dengan karakter Y yang berada pada *Index* sebelumnya. Karena karakter A tidak ditemukan, maka karakter A dimasukan kedalam tabel bmBc dengan Nilai OH 2, sesuai jumlah perpindahan. Hasilnya dapat dilihat pada Tabel 3.27.

Tabel 3.27 Mencari Nilai OH (Perbandingan ke-2)

Bandingkan A dengan Y		Hasil		
BmBc(awal)		BmBc(akhir)		
Karakter	Y	**Karakter**	Y	A
Nilai OH	1	**Nilai OH**	1	2

Pada pencacahan karakter ketiga, dilakukan pada *Index* sebelumnya dikurangi 1, yaitu pada *Index* 3. Dan jumlah perpindahan diubah menjadi 3. Seperti Tabel 3.28.

Tabel 3.28 Mencari Nilai OH (Pergeseran ke-3)

							Mulai : *Index* = 4 – 1 = 3
Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Pindah = 3							

Pada perbandingan ketiga, bandingkan karakter J yang ada pada *Index* 3 dengan karakter A dan Y yang berada pada *Index* sebelumnya. Karena karakter J tidak ditemukan, maka karakter J dimasukan kedalam tabel bmBc dengan Nilai OH 3, sesuai jumlah perpindahan. Hasilnya dapat dilihat pada Tabel 3.29.

Tabel 3.29 Mencari Nilai OH (Perbandingan ke-3)

Bandingkan J dengan A, Y			Hasil			
BmBc(awal)			BmBc(akhir)			
Karakter	Y	A	**Karakter**	Y	A	J
Nilai OH	1	2	**Nilai OH**	1	2	3

Pada pencacahan karakter keempat, dilakukan pada *Index* sebelumnya dikurangi 1, yaitu pada *Index* 2. Dan jumlah perpindahan diubah menjadi 4. Seperti Tabel 3.30.

Tabel 3.30 Mencari Nilai OH (Pergeseran ke-4)

							Mulai : *Index* = 3 – 1 = 2
Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Pindah = 4							

Pada perbandingan keempat, bandingkan karakter N yang ada pada *Index* 2 dengan karakter J, A dan Y yang berada pada *Index* sebelumnya. Karena karakter N tidak ditemukan, maka karakter N dimasukan kedalam tabel bmBc dengan Nilai OH 4, sesuai jumlah perpindahan. Hasilnya dapat dilihat pada Tabel 3.31.

Tabel 3.31 Mencari Nilai OH (Perbandingan ke-4)

Bandingkan N dengan J, A, Y				Hasil				
BmBc(awal)				BmBc(akhir)				
Karakter	Y	A	J	**Karakter**	Y	A	J	N
Nilai OH	1	2	3	**Nilai OH**	1	2	3	4

Pada pencacahan karakter kelima, dilakukan pada *Index* sebelumnya dikurangi 1, yaitu pada *Index* 1. Dan jumlah perpindahan diubah menjadi 5. Seperti Tabel 3.32.

Tabel 3.32 Mencari Nilai OH (Pergeseran ke-5)

							Mulai : *Index* = 2 – 1 = 1
Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Pindah = 5							

Pada perbandingan kelima, bandingkan karakter U yang ada pada *Index* 1 dengan karakter N, J, A dan Y yang berada pada *Index* sebelumnya. Karena karakter U tidak ditemukan, maka karakter U dimasukan kedalam tabel bmBc dengan Nilai OH 5, sesuai jumlah perpindahan. Hasilnya dapat dilihat pada Tabel 3.33.

Tabel 3.33 Mencari Nilai OH (Perbandingan ke-5)

Bandingkan U dengan N, J, A, Y					Hasil					
BmBc(awal)					BmBc(akhir)					
Karakter	Y	A	J	N	**Karakter**	Y	A	J	N	U
Nilai OH	1	2	3	4	**Nilai OH**	1	2	3	4	5

Pada pencacahan karakter keenam, dilakukan pada *Index* sebelumnya dikurangi 1, yaitu pada *Index* 0. Dan jumlah perpindahan diubah menjadi 6. Seperti Tabel 3.34.

Tabel 3.34 Mencari Nilai OH (Pergeseran ke-6)

							Mulai : *Index* = 1 – 1 = 0
Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Pindah = 6							

Pada perbandingan keenam, bandingkan karakter S yang ada pada *Index* 0 dengan karakter U, N, J, A dan Y yang berada pada *Index* sebelumnya. Karena karakter S tidak ditemukan, maka karakter S dimasukan kedalam tabel bmBc dengan nilai OH sejumlah perpindahan, yaitu 6. Hasilnya dapat dilihat pada Tabel 3.35.

Tabel 3.35 Mencari Nilai OH (Perbandingan ke-6)

Bandingkan S dengan U, N, J, A, Y						Hasil						
BmBc(awal)						BmBc(akhir)						
Karakter	Y	A	J	N	U	**Karakter**	Y	A	J	N	U	S
Nilai OH	1	2	3	4	5	**Nilai OH**	1	2	3	4	5	6

Pencacahan ketujuh merupakan proses pencacahan terakhir, karena *Index* sebelumnya adalah 0, oleh sebab itu pencacahan dilakukan pada *Index* paling kanan (*Index*= Len(*pattern*) – 1), yaitu pada *Index* 6. Dan jumlah perpindahan diubah menjadi 7. Seperti Tabel 3.36.

Tabel 3.36 Mencari Nilai OH (Pergeseran ke-7)

Mulai : *Index* = len(*pattern*) – 1 = 7 - 1 = 6							
Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Pindah = 7							

Pada perbandingan ketujuh, bandingkan karakter A yang ada pada *Index* 6 dengan karakter S, U, N, J, A dan Y yang berada pada *Index* yang sudah dibandingkan sebelumnya. Karena karakter A ditemukan, maka karakter A dimasukan kedalam tabel bmBc dengan Nilai OH karater sebelumnya. Hasilnya bisa dilihat di Tabel 3.37.

Tabel 3.37 Mencari Nilai OH (Perbandingan ke-7)

Bandingkan A dengan S, U, N, J, A, Y							Hasil							
BmBc(awal)							BmBc(akhir)							
Karakter	Y	A	J	N	U	S	**Karakter**	Y	A	J	N	U	S	A
Nilai OH	1	2	3	4	5	6	**Nilai OH**	1	2	3	4	5	6	2

Untuk mendapatkan hasil akhir dari proses ini, perhatikan Tabel 3.37 pada bagian BmBc(akhir), masukan setiap Nilai OH masing-masing karakter ke dalam karakter *pattern*. Sehingga akan terlihat seperti Tabel 3.38.

Tabel 3.38 BmBc akhir

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Pattern	S	U	N	J	A	Y	A
Nilai OH	6	5	4	3	2	1	2

Tabel 3.38 merupakan acuan untuk menentukan tabel ztBc. Langkah selanjutnya adalah masukan semua karakter yang terdapat pada sumber teks dan buatlah menjadi dua dimensi, karakter yang dimasukan diurutkan sesuai abjad tak berulang. seperti Tabel 3.39.

Tabel 3.39 ztBc awal

	A	B	D	G	H	I	J	K	L	M	N	O	P	R	S	T	U	W	Y	,	.	Sp
A																						
B																						
D																						
G																						
H																						
I																						
J																						
K																						
L																						
M																						
N																						
O																						
P																						
R																						
S																						
T																						
U																						
W																						
Y																						
,																						
.																						
Sp																						

Setelah tabel dua dimensi terbentuk, langkah selanjutnya adalah mengisi tabel dengan cara melihat karakter pada baris dan karakter pada kolom. Contohnya, untuk baris 1 kolom 1 didapat karakter A dan A. Lihat baris *pattern* pada Tabel 3.38, apakah pola tersebut ada pada *pattern*? Jawabannya akan bervariasi, sehingga ada aturan sebagai berikut:

1. Jika pola yang dicari ada, maka masukan nilai OH karakter kolom.
2. Jika pola yang dicari tidak ada, maka masukan panjang *pattern*.
3. Jika karakter pada suatu kolom yang dituju adalah karakter pertama pada *pattern*, maka masukan Nilai OH karakter tersebut, tanpa melihat karakter baris.

Untuk pola AA, pola ini tidak ada pada *pattern*. Maka masukan 7(aturan 2). Hasilnya dapat dilihat pada Tabel 3.40.

Tabel 3.40 ztBc (Perbandingan ke-1)

	A	B	D	G	H	I	J	K	L	M	N	O	P	R	S	T	U	W	Y	,	.	Sp
A	**7**																					
B																						
D																						
G																						
H																						
I																						
J																						
K																						
L																						
M																						
N																						
O																						
P																						
R																						
S																						
T																						
U																						
W																						
Y																						
,																						
.																						
Sp																						

Selanjutnya mengisi baris 1 kolom 15 didapat karakter A dan S. Lihat pada tabel 3.38, pola AS tidak terdapat pada *pattern*. Tetapi karakter pada kolom 15 baris 1 adalah karakter pertama pada *pattern*, maka masukan 6 (aturan 3). Hasilnya dapat dilihat pada Tabel 3.41.

Tabel 3.41 ztBc (Perbandingan ke-2)

	A	B	D	G	H	I	J	K	L	M	N	O	P	R	S	T	U	W	Y	,	.	Sp
A	**7**														**6**							
B																						
D																						
G																						
H																						
I																						
J																						
K																						
L																						
M																						
N																						
O																						
P																						
R																						
S																						
T																						
U																						
W																						
Y																						
,																						
.																						
Sp																						

Selanjutnya baris 1 kolom 19 didapat karakter A dan Y. Lihat pada tabel 3.38, pola AY terdapat pada *pattern*. Masukan nilai 1 (aturan 1). Seperti Tabel 3.42.

Tabel 3.42 ztBc (Perbandingan ke-3)

	A	B	D	G	H	I	J	K	L	M	N	O	P	R	S	T	U	W	Y	,	.	Sp
A	**7**														**6**				**1**			
B																						
D																						
G																						
H																						
I																						
J																						
K																						
L																						
M																						
N																						
O																						
P																						
R																						
S																						
T																						
U																						
W																						
Y																						
,																						
.																						
Sp																						

Langkah tersebut dilakukan sampai tabel ztBc terisi semua tergantung kondisi setiap karakter. Seperti Tabel 3.43.

Tabel 3.43 ztBc akhir

	A	B	D	G	H	I	J	K	L	M	N	O	P	R	S	T	U	W	Y	,	.	Sp
A	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	**1**	7	7	7
B	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
D	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
G	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
H	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
I	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
J	**2**	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
K	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
L	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
M	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
N	7	7	7	7	7	7	**3**	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
O	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
P	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
R	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
S	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	**5**	7	7	7	7	7
T	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
U	7	7	7	7	7	7	7	7	7	7	**4**	7	7	7	**6**	7	7	7	7	7	7	7
W	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
Y	**2**	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
,	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
.	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7
Sp	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**6**	7	7	7	7	7	7	7

2. Membuat tabel ztGs

Setelah mendapatkan hasil akhir pada tabel *ztBc*, maka dilanjutkan dengan membuat tabel akhiran/*suffix* dari *pattern*. Tabel untuk mengisi Nilai MH dapat dilihat pada Tabel 3.44.

Tabel 3.44 BmGs awal

	BmGs						
Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Nilai *MH*	**?**	**?**	**?**	**?**	**?**	**?**	**?**

Penjelasan

a. *Index* adalah indeks dari *pattern*.
b. Karakter adalah *pattern* yang dicari.
c. Nilai MH(*Match Heuristic*) adalah nilai pergeseran *Good Suffix*. Nilai MH nantinya akan digunakan ketika ditemukan kecocokan pada saat mencocokan *string* karakter pertama atau lebih.

Untuk mengisi Nilai MH langkah pertama adalah buatlah tabel *suffix* dari kanan ke kiri dan tabel *suffix* dari kiri ke kanan.

1. Tabel *suffix* dari kanan ke kiri, kolom *suffix* diperoleh dari pencacahan *pattern* yang dimulai dari kanan ke kiri, yaitu dari karakter A sampai karakter S.
2. Tabel *suffix* dari kiri ke kanan, kolom *suffix* diperoleh dari pencacahan *pattern* yang dimulai dari kiri ke kanan, yaitu dari S sampai karakter A.

Hasilnya akan terlihat seperti Tabel 3.45.

Tabel 3.45 *Suffix* (kanan ke kiri & *Suffix* kiri ke kanan)

Pattern: SUNJAYA (kanan ke kiri)					*Pattern: SUNJAYA (kiri ke kanan)*			
Idx	***Prefix***	***Suffix***	**Len**		**Idx**	***Prefix***	***Suffix***	**Len**
6	A	SUNJAY	6		0	S	UNJAYA	6
5	YA	SUNJA	5		1	SU	NJAYA	5
4	AYA	SUNJ	4		2	SUN	JAYA	4
3	JAYA	SUN	3		3	SUNJ	AYA	3
2	NJAYA	SU	2		4	SUNJA	YA	2
1	UNJAYA	S	1		5	SUNJAY	A	1
0	SUNJAYA	*NULL*	0		6	SUNJAYA	*NULL*	0

1. Pembuatan tabel *suffix* dari kanan ke kiri berfungsi untuk dijadikan *suffix comparator* pada Tabel *Suffix*.
2. Pembuatan *suffix* dari kiri ke kanan berfungsi sebagai *suffix* yang nantinya akan dibandingkan dan digunakan untuk mengisi nilai MH.

Dari Tabel 3.45 akan terbentuk Tabel 3.46, tabel ini berfungsi untuk mencari nilai MH. *Suffix Comparator* yang terdapat pada tabel 3.46 diambil dari kolom *suffix* yang ada pada Tabel 3.24 *Suffix* (kanan ke kiri). Sedangkan baris *suffix* yang terdapat pada tabel 3.46 diambil dari kolom *suffix* yang ada pada Tabel 3.24 *Suffix* (kiri ke kanan). Maka akan terlihat seperti Tabel 3.46.

Tabel 3.46 *Suffix* lengkap beserta *suffix comparator*

	Suffix							
Index	0	1	2	3	4	5	6	***Move***
Prefix	S	U	N	J	A	Y	A	
Suffix	**UNJAYA**	**NJAYA**	**JAYA**	**AYA**	**YA**	**A**	*null*	
Suffix Comparator	SUNJAY							**1**
	SUNJA							**2**
	SUNJ							**3**
	SUN							**4**
	SU							**5**
	S							**6**
	null							**7**
Nilai MH	**?**	**?**	**?**	**?**	**?**	**?**	**?**	

Langkah selanjutnya, adalah melakukan pemecahan akhiran pada *suffix* terhadap *Suffix Comparator* Tabel. Berikan nilai MH pada tabel BmGs dengan *moving long* yang berkesesuaian. Terdapat beberapa ketentuan yang dapat menjadi acuan dalam mencari nilai MH.

1) Cacah *suffix* pada masing-masing *Index* terhadap seluruh *suffix comparator*
2) Jika *length* kedua *suffix* tidak sama panjang, maka potong salah satu *suffix* yang memiliki *length* terbesar. Besaran pemotongan *suffix* disesuaikan dengan nilai *length* terkecil kearah paling kanan karakter. Jika ditemukan kecocokan antar *suffix*, bandingkan *prefix* kedua *suffix* (jika masing-masing

suffix memiliki *prefix*). Jika *prefix* yang diperbandingkan adalah sama, maka kecocokan tidak dapat diterima.

3) Lakukan poin 2 hingga *Index suffix* mencapai 0.
4) Untuk *Index suffix* terbesar, secara default akan diberikan nilai 1 pada Nilai MH, sedangkan yang lainnya diberikan nilai sesuai dengan nilai *move* yang diraih.

Untuk proses pertama diberikan nilai 1 pada Nilai MH, karena *Index* terbesar (Ketentuan 4). Prosesnya dapat dilihat pada Tabel 3.47.

Tabel 3.47 Proses 1 PreBmGs

***Process* #1-01**

***Suffix* Tabel**

Index	6	Mov
Prefix	A	
Suffix	*Null*	
Suffix Comparator	SUNJAY	1
	SUNJA	2
	SUNJ	3
	SUN	4
	SU	5
	S	6
	Null	7
Nilai MH	1	

BmGs

Index	0	1	2	3	4	5	6
Karakter	S	U	N	J	A	Y	A
Nilai MH	?	?	?	?	?	?	1

Proses kedua, terdapat beberapa perbandingan untuk mencari nilai MH. Bandingkan karakter *suffix* dengan *Suffix Comparator*. Jumlah karakter yang dibandingkan disesuaikan dengan jumlah karakter yang paling sedikit (Ketentuan 2). Pada perbandingan kedua, terdapat karakter *suffix* yang sama dengan *suffix comparator* dan *prefix*-nya berbeda, pada *Process* #2-02. Sehingga jumlah perpindahan dimasukan ke dalam Nilai MH yaitu 2. Bisa dilihat pada Tabel 3.48.

Tabel 3.48 Proses 2 PreBmGs

***Process* #2-01** ***Suffix* Tabel**			***Process* #2-02** ***Suffix* Tabel**			***Process* #2-03** ***Suffix* Tabel**		
Index	5	Mov	*Index*	5	Mov	*Index*	5	Mov
Prefix	Y		*Prefix*	**Y**		*Prefix*	Y	
Suffix	**A**		*Suffix*	**A**		*Suffix*	**A**	
Suffix Comparator	SUNJA**Y**	**1**	*Suffix Comparator*	SUNJAY	1	*Suffix Comparator*	SUNJAY	1
	SUNJA	2		SUN **J A**	**2**		SUNJA	2
	SUNJ	3		SUNJ	3		SUN**J**	**3**
	SUN	4		SUN	4		SUN	4
	SU	5		SU	5		SU	5
	S	6		S	6		S	6
	Null	7		*Null*	7		*Null*	7
Nilai MH	?		**Nilai MH**	2		**Nilai MH**	2	

***Process* #2-04** ***Suffix* Tabel**			***Process* #2-05** ***Suffix* Tabel**			***Process* #2-06** ***Suffix* Tabel**		
Index	5	Mov	*Index*	5	Mov	*Index*	5	Mov
Prefix	**Y**		*Prefix*	**Y**		*Prefix*	**Y**	
Suffix	**A**		*Suffix*	**A**		*Suffix*	**A**	
Suffix Comparator	SUNJAY	1	*Suffix Comparator*	SUNJAY	1	*Suffix Comparator*	SUNJAY	1
	SUNJA	2		SUNJA	2		SUNJA	2
	SUNJ	3		SUNJ	3		SUNJ	3
	SU**N**	**4**		SUN	4		SUN	4
	SU	5		S**U**	**5**		SU	5
	S	6		S	6		**S**	**6**
	Null	7		*Null*	7		*Null*	7
Nilai MH	2		**Nilai MH**	2		**Nilai MH**	2	

***Process* #2-07** ***Suffix* Tabel**		
Index	5	Mov
Prefix	Y	
Suffix	**A**	
Suffix Comparator	SUNJAY	1
	SUNJA	2
	SUNJ	3
	SUN	4
	SU	5
	S	6
	Null	**7**
Nilai MH	2	

BmGs							
Index	0	1	2	**3**	4	5	6
Karakter	S	U	N	J	A	Y	A
Nilai MH	?	?	?	?	?	2	1

Proses ketiga, terdapat beberapa perbandingan untuk mencari nilai MH. Bandingkan karakter *suffix* dengan *Suffix Comparator*. Jumlah karakter yang dibandingkan disesuaikan dengan jumlah karakter yang paling sedikit (Ketentuan 2). Ternyata karakter *suffix* tidak ada yang sama dengan *suffix comparator* pada setiap perbandingan. Sehingga jumlah perpindahan dimasukan ke dalam Nilai MH yaitu 7. Bisa dilihat pada Tabel 3.49.

Tabel 3.49 Proses 3 PreBmGs

***Process* #3-01** ***Suffix* Tabel**				***Process* #3-02** ***Suffix* Tabel**				***Process* #3-03** ***Suffix* Tabel**			
Index	4	Mov		*Index*	4	Mov		*Index*	4	Mov	
Prefix	**A**			*Prefix*	**A**			*Prefix*	**A**		
Suffix	**YA**			*Suffix*	**YA**			*Suffix*	**YA**		
Suffix Comparator	SUNJ**AY**	**1**		*Suffix Comparator*	SUNJAY	1		*Suffix Comparator*	SUNJAY	1	
	SUNJA	2			SUN**JA**	**2**			SUNJA	2	
	SUNJ	3			SUNJ	3			SU**NJ**	**3**	
	SUN	4			SUN	4			SUN	4	
	SU	5			SU	5			SU	5	
	S	6			S	6			S	6	
	Null	7			*Null*	7			*Null*	7	
Nilai MH	?			**Nilai MH**	?			**Nilai MH**	?		

***Process* #3-04** ***Suffix* Tabel**				***Process* #3-05** ***Suffix* Tabel**				***Process* #3-06** ***Suffix* Tabel**			
Index	4	Mov		*Index*	4	Mov		*Index*	4	Mov	
Prefix	A			*Prefix*	**A**			*Prefix*	**A**		
Suffix	**YA**			*Suffix*	**YA**			*Suffix*	Y**A**		
Suffix Comparator	SUNJAY	1		*Suffix Comparator*	SUNJAY	1		*Suffix Comparator*	SUNJAY	1	
	SUNJA	2			SUNJA	2			SUNJA	2	
	SUNJ	3			SUNJ	3			SUNJ	3	
	S**UN**	**4**			SUN	4			SUN	4	
	SU	5			**SU**	**5**			SU	5	
	S	6			S	6			**S**	**6**	
	Null	7			*Null*	7			*Null*	7	
Nilai MH	?			**Nilai MH**	?			**Nilai MH**	?		

***Process* #3-07** ***Suffix* Tabel**			
Index	4	Mov	
Prefix	**A**		
Suffix	**YA**		
Suffix Comparator	SUNJAY	1	
	SUNJA	2	
	SUNJ	3	
	SUN	4	
	SU	5	
	S	6	
	Null	**7**	
Nilai MH	7		

BmGs							
Index	0	1	2	**3**	4	5	6
Karakter	S	U	N	J	A	Y	A
Nilai MH	?	?	?	?	7	2	1

Proses keempat, terdapat beberapa perbandingan untuk mencari nilai MH. Bandingkan karakter *suffix* dengan *Suffix Comparator*. Jumlah karakter yang dibandingkan disesuaikan dengan jumlah karakter yang paling sedikit (Ketentuan 2). Ternyata karakter *suffix* tidak ada yang sama dengan *suffix comparator* pada setiap perbandingan. Sehingga jumlah perpindahan dimasukan ke dalam Nilai MH yaitu 7. Bisa dilihat pada Tabel 3.50.

Tabel 3.50 Proses 4 PreBmGs

***Process* #4-01** ***Suffix* Tabel**				***Process* #4-02** ***Suffix* Tabel**				***Process* #4-03** ***Suffix* Tabel**			
Index	3	Mov		*Index*	3	Mov		*Index*	3	Mov	
Prefix	**J**			*Prefix*	**J**			*Prefix*	**J**		
Suffix	**AYA**			*Suffix*	**AYA**			*Suffix*	**AYA**		
Suffix Comparator	SUN**JAY**	**1**		*Suffix Comparator*	SUNJAY	1		*Suffix Comparator*	SUNJAY	1	
	SUNJA	2			SU**NJA**	**2**			SUNJA	2	
	SUNJ	3			SUNJ	3			S**UNJ**	**3**	
	SUN	4			SUN	4			SUN	4	
	SU	5			SU	5			SU	5	
	S	6			S	6			S	6	
	Null	7			*Null*	7			*Null*	7	
Nilai MH	?			**Nilai MH**	?			**Nilai MH**	?		

***Process* #4-04** ***Suffix* Tabel**				***Process* #4-05** ***Suffix* Tabel**				***Process* #4-06** ***Suffix* Tabel**			
Index	3	Mov		*Index*	3	Mov		*Index*	3	Mov	
Prefix	**J**			*Prefix*	**J**			*Prefix*	**J**		
Suffix	**AYA**			*Suffix*	A**YA**			*Suffix*	AY**A**		
Suffix Comparator	SUNJAY	1		*Suffix Comparator*	SUNJAY	1		*Suffix Comparator*	SUNJAY	1	
	SUNJA	2			SUNJA	2			SUNJA	2	
	SUNJ	3			SUNJ	3			SUNJ	3	
	SUN	**4**			SUN	4			SUN	4	
	SU	5			**SU**	**5**			SU	5	
	S	6			S	6			**S**	**6**	
	Null	7			*Null*	7			*Null*	7	
Nilai MH	?			**Nilai MH**	?			**Nilai MH**	?		

***Process* #4-07** ***Suffix* Tabel**			
Index	3	Mov	
Prefix	**J**		
Suffix	**AYA**		
Suffix Comparator	SUNJAY	1	
	SUNJA	2	
	SUNJ	3	
	SUN	4	
	SU	5	
	S	6	
	Null	**7**	
Nilai MH	7		

BmGs							
Index	0	1	2	**3**	4	5	6
Karakter	S	U	N	J	A	Y	A
Nilai MH	?	?	?	7	7	2	1

Proses kelima, terdapat beberapa perbandingan untuk mencari nilai MH. Bandingkan karakter *suffix* dengan *Suffix Comparator*. Jumlah karakter yang dibandingkan disesuaikan dengan jumlah karakter yang paling sedikit (Ketentuan 2). Ternyata karakter *suffix* tidak ada yang sama dengan *suffix comparator* pada setiap perbandingan. Sehingga jumlah perpindahan dimasukan ke dalam Nilai MH yaitu 7. Bisa dilihat pada Tabel 3.51.

Tabel 3.51 Proses 5 PreBmGs

***Process* #5-01 *Suffix* Tabel**				***Process* #5-02 *Suffix* Tabel**				***Process* #5-03 *Suffix* Tabel**			
Index	2	Mov		*Index*	2	Mov		*Index*	2	Mov	
Prefix	**N**			*Prefix*	**N**			*Prefix*	N		
Suffix	**JAYA**			*Suffix*	**JAYA**			*Suffix*	**JAYA**		
Suffix Comparator	SU**NJAY**	**1**		*Suffix Comparator*	SUNJAY	1		*Suffix Comparator*	SUNJAY	1	
	SUNJA	2			S**UNJA**	**2**			SUNJA	2	
	SUNJ	3			SUNJ	3			**SUNJ**	**3**	
	SUN	4			SUN	4			SUN	4	
	SU	5			SU	5			SU	5	
	S	6			S	6			S	6	
	Null	7			*Null*	7			*Null*	7	
Nilai MH	?			**Nilai MH**	?			**Nilai MH**	?		

***Process* #5-04 *Suffix* Tabel**				***Process* #5-05 *Suffix* Tabel**				***Process* #5-06 *Suffix* Tabel**			
Index	2	Mov		*Index*	2	Mov		*Index*	2	Mov	
Prefix	N			*Prefix*	N			*Prefix*	N		
Suffix	J**AYA**			*Suffix*	JA**YA**			*Suffix*	JAY**A**		
Suffix Comparator	SUNJAY	1		*Suffix Comparator*	SUNJAY	1		*Suffix Comparator*	SUNJAY	1	
	SUNJA	2			SUNJA	2			SUNJA	2	
	SUNJ	3			SUNJ	3			SUNJ	3	
	SUN	**4**			SUN	4			SUN	4	
	SU	5			**SU**	**5**			SU	5	
	S	6			S	6			**S**	**6**	
	Null	7			*Null*	7			*Null*	7	
Nilai MH	?			**Nilai MH**	?			**Nilai MH**	?		

***Process* #5-07 *Suffix* Tabel**			
Index	2	Mov	
Prefix	**N**		
Suffix	**JAYA**		
Suffix Comparator	SUNJAY	1	
	SUNJA	2	
	SUNJ	3	
	SUN	4	
	SU	5	
	S	6	
	Null	**7**	
Nilai MH	7		

BmGs							
Index	0	1	2	**3**	4	5	6
Char	S	U	N	J	A	Y	A
Nilai MH	?	?	7	7	7	2	1

Proses keenam, terdapat beberapa perbandingan untuk mencari nilai MH. Bandingkan karakter *suffix* dengan *Suffix Comparator*. Jumlah karakter yang dibandingkan disesuaikan dengan jumlah karakter yang paling sedikit (Ketentuan 2). Ternyata karakter *suffix* tidak ada yang sama dengan *suffix comparator* pada setiap perbandingan. Sehingga jumlah perpindahan dimasukan ke dalam Nilai MH yaitu 7. Bisa dilihat pada Tabel 3.52.

Tabel 3.52 Proses 6 PreBmGs

***Process* #6-01** ***Suffix* Tabel**		
Index	1	Mov
Prefix	**U**	
Suffix	**NJAYA**	
Suffix Comparator	S**UNJAY**	**1**
	SUNJA	2
	SUNJ	3
	SUN	4
	SU	5
	S	6
	Null	7
Nilai MH	?	

***Process* #6-02** ***Suffix* Tabel**		
Index	1	Mov
Prefix	**U**	
Suffix	**NJAYA**	
Suffix Comparator	SUNJAY	1
	SUNJA	**2**
	SUNJ	3
	SUN	4
	SU	5
	S	6
	Null	7
Nilai MH	?	

***Process* #6-03** ***Suffix* Tabel**		
Index	1	Mov
Prefix	U	
Suffix	N**JAYA**	
Suffix Comparator	SUNJAY	1
	SUNJA	2
	SUNJ	**3**
	SUN	4
	SU	5
	S	6
	Null	7
Nilai MH	?	

***Process* #6-04** ***Suffix* Tabel**		
Index	1	Mov
Prefix	U	
Suffix	NJ**AYA**	
Suffix Comparator	SUNJAY	1
	SUNJA	2
	SUNJ	3
	SUN	**4**
	SU	5
	S	6
	Null	7
Nilai MH	?	

***Process* #6-05** ***Suffix* Tabel**		
Index	1	Mov
Prefix	U	
Suffix	NJA**YA**	
Suffix Comparator	SUNJAY	1
	SUNJA	2
	SUNJ	3
	SUN	4
	SU	**5**
	S	6
	Null	7
Nilai MH	?	

***Process* #6-06** ***Suffix* Tabel**		
Index	1	Mov
Prefix	**U**	
Suffix	NJAY**A**	
Suffix Comparator	SUNJAY	1
	SUNJA	2
	SUNJ	3
	SUN	4
	SU	5
	S	**6**
	Null	7
Nilai MH	?	

***Process* #6-07** ***Suffix* Tabel**		
Index	1	Mov
Prefix	U	
Suffix	NJAYA	
Suffix Comparator	SUNJAY	1
	SUNJA	2
	SUNJ	3
	SUN	4
	SU	5
	S	6
	Null	**7**
Nilai MH	7	

BmGs							
Index	0	1	2	**3**	4	5	6
Karakter	S	U	N	J	A	Y	A
Nilai MH	?	7	7	7	7	2	1

Proses ketujuh, terdapat beberapa perbandingan untuk mencari nilai MH. Bandingkan karakter *suffix* dengan *Suffix Comparator*. Jumlah karakter yang dibandingkan disesuaikan dengan jumlah karakter yang paling sedikit (Ketentuan 2). Ternyata karakter *suffix* tidak ada yang sama dengan *suffix comparator* pada setiap perbandingan. Sehingga jumlah perpindahan dimasukan ke dalam Nilai MH yaitu 7. Bisa dilihat pada Tabel 3.53.

Tabel 3.53 Proses 7 PreBmGs

***Process* #7-01** ***Suffix* Tabel**				***Process* #7-02** ***Suffix* Tabel**				***Process* #7-03** ***Suffix* Tabel**			
Index	0	Mov		*Index*	0	Mov		*Index*	0	Mov	
Prefix	S			*Prefix*	S			*Prefix*	S		
Suffix	**UNJAYA**			*Suffix*	U**NJAYA**			*Suffix*	UN**JAYA**		
Suffix Comparator	**SUNJAY**	**1**		*Suffix Comparator*	SUNJAY	1		*Suffix Comparator*	SUNJAY	1	
	SUNJA	2			**SUNJA**	**2**			SUNJA	2	
	SUNJ	3			SUNJ	3			**SUNJ**	**3**	
	SUN	4			SUN	4			SUN	4	
	SU	5			SU	5			SU	5	
	S	6			S	6			S	6	
	Null	7			*Null*	7			*Null*	7	
Nilai MH	?			**Nilai MH**	?			**Nilai MH**	?		

***Process* #7-04** ***Suffix* Tabel**				***Process* #7-05** ***Suffix* Tabel**				***Process* #7-06** ***Suffix* Tabel**			
Index	0	Mov		*Index*	0	Mov		*Index*	0	Mov	
Prefix	S			*Prefix*	S			*Prefix*	S		
Suffix	UNJ**AYA**			*Suffix*	UNJA**YA**			*Suffix*	UNJAYA		
Suffix Comparator	SUNJAY	1		*Suffix Comparator*	SUNJAY	1		*Suffix Comparator*	SUNJAY	1	
	SUNJA	2			SUNJA	2			SUNJA	2	
	SUNJ	3			SUNJ	3			SUNJ	3	
	SUN	**4**			SUN	4			SUN	4	
	SU	5			**SU**	**5**			SU	5	
	S	6			S	6			**S**	**6**	
	Null	7			*Null*	7			*Null*	7	
Nilai MH	?			**Nilai MH**	?			**Nilai MH**	?		

***Process* #7-07** ***Suffix* Tabel**			
Index	0	Mov	
Prefix	**S**		
Suffix	**UNJAYA**		
Suffix Comparator	SUNJAY	1	
	SUNJA	2	
	SUNJ	3	
	SUN	4	
	SU	5	
	S	6	
	Null	**7**	
Nilai MH	7		

BmGs							
Index	0	1	2	**3**	4	5	6
Karakter	S	U	N	J	A	Y	A
Nilai MH	7	7	7	7	7	2	1

Hasil akhir dari setiap langkah-langkah sebelumnya dapat dilihat pada Tabel 3.54.

Tabel 3.54 bmGs/ztGs

Index	0	1	2	3	4	5	6
Karkater	S	U	N	J	A	Y	A
Nilai MH	7	7	7	7	7	2	1

Karena tabel bmGs dan ztGs sama, maka untuk proses *preprocessing* sudah selesai dengan hasil yang dapat dilihat pada Tabel 3.43 untuk tabel ztBc, dan pada Tabel 3.54 untuk tabel bmGs atau ztGs.

3. Proses Pencocokan

Langkah terakhir adalah proses pencocokan. Dengan data sebagai berikut.

Teks : **KI MASKARA BALAD PAJUANG TI PRIANGAN, BAPANA SUNJAYA ATAWA AKINA NYI RODIAH.**

Pattern : **SUNJAYA**

Percobaan ke-1:

I	0	1	2	3	4	5	**6**	7	8	9	10	11	12	13	14	15	16	17	18	...
Teks	K	I		M	A	S	**K**	A	R	A		B	A	L	A	D		P	A	...
Pattern	S	U	N	J	A	Y	**A**													

Pergeseran 7 (ztBc[S][K])

Dari hasil percobaan ke-1, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakte K pada sumber teks. Artinya, pada percobaan ke-1 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh tujuh karakter. Nilai pergeseran tujuh karakter ini diperoleh dari tabel 3.43 ztBc dimana dicocokannya dua karakter akhir pada teks yang sejajar dengan karakter akhir *pattern*. Pada contoh ini diisi dengan karakter S dan karakter K. Maka cek pada tabel ztBc baris S dan kolom K maka bernilai tujuh.

Percobaan ke-2:

I	0	1	2	3	4	5	6	7	8	9	10	11	12	**13**	14	15	16	17	18	...
Teks	K	I		M	A	S	K	A	R	A		B	A	**L**	A	D		P	A	...
Pattern								S	U	N	J	A	Y	**A**						

Pergeseran 7 (ztBc[A][L])

Dari hasil percobaan ke-2, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter L pada sumber teks. Artinya, pada percobaan ke-2 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh tujuh karakter. Pergeseran tujuh karakter ini diambil dari tabel 3.43 ztBc dengan baris A dan kolom L.

Percobaan ke-3:

I	14	15	16	17	18	19	**20**	21	22	23	24	25	26	27	28	29	30	31	...
Teks	A	D		P	A	J	**U**	A	N	G		T	I		P	R	I	A	...
Pattern	S	U	N	J	A	Y	**A**												

Pergeseran 7 (ztBc[J][U])

Dari hasil percobaan ke-3, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter U pada sumber teks. Artinya, pada percobaan ke-3 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh tujuh karakter. Pergeseran karakter ini diambil dari tabel 3.43 ztBc dengan baris J dan kolom U.

Percobaan ke-4:

I	14	15	16	17	18	19	20	21	22	23	24	25	26	**27**	28	29	30	31	...
Teks	A	D		P	A	J	U	A	N	G		T	I		P	R	I	A	...
Pattern								S	U	N	J	A	Y	**A**					

Pergeseran 7 (ztBc[I][Sp])

Dari hasil percobaan ke-4, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter Sp pada sumber teks. Artinya, pada percobaan ke-4 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh tujuh karakter. Pergeseran karakter ini diambil dari tabel 3.43 ztBc dengan baris I dan kolom Sp.

Percobaan ke-5:

I	28	29	30	31	32	**33**	**34**	35	36	37	38	39	40	41	42	43	44	45	...
Teks	P	R	I	A	N	**G**	**A**	N	,		B	A	P	A	N	A		S	...
Pattern	S	U	N	J	A	**Y**	**A**												

Pergeseran 1 (ztGs[A])

Dari hasil percobaan ke-5, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter A pada sumber teks. Artinya, pada percobaan ke-5 terjadi kecocokan, maka pencocokan dilakukan terhadap karakter sebelum karakter A, yaitu karakter Y pada *pattern*, karakter Y sejajar dengan karakter G pada teks, terjadi ketidakcocokkan. Karena pada perbandingan pertama terjadi kecocokkan, maka nilai pergeseran selanjutnya di ambil dari tabel 3.54 ztGs dengan karakter A, maka pergeseran sejauh satu karakter.

Percobaan ke-6:

I	28	29	30	31	32	33	34	**35**	36	37	38	39	40	41	42	43	44	45	...
Teks	P	R	I	A	N	G	A	**N**	,		B	A	P	A	N	A		S	...
Pattern		S	U	N	J	A	Y	**A**											

Pergeseran 7 (ztBc[A][N])

Dari hasil percobaan ke-6, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter N pada sumber teks. Artinya, pada percobaan ke-6 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh tujuh karakter. Pergeseran karakter ini diambil dari tabel 3.43 ztBc dengan baris A dan kolom N.

Percobaan ke-7:

I	28	29	30	31	32	33	34	35	36	37	38	39	40	41	**42**	43	44	45	...
Teks	P	R	I	A	N	G	A	N	,		B	A	P	A	**N**	A		S	...
Pattern									S	U	N	J	A	Y	**A**				

Pergeseran 7 (ztBc[A][N])

Dari hasil percobaan ke-7, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter N pada sumber teks. Artinya, pada percobaan ke-7 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh tujuh karakter. Pergeseran karakter diambil dari tabel ztBc dengan baris A dan kolom N.

Percobaan ke-8:

I	43	44	45	46	47	**48**	**49**	50	51	52	53	54	55	56	57	58	59	60	...
Teks	A		S	U	N	**J**	**A**	Y	A		A	T	A	W	A		A	K	...
Pattern	S	U	N	J	A	**Y**	**A**												

Pergeseran 1 (ztGs[A])

Dari hasil percobaan ke-8, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter A pada sumber teks. Artinya, pada percobaan ke-8 terjadi kecocokan, maka pencocokan dilakukan terhadap karakter sebelum karakter A, yaitu karakter Y pada *pattern*, karakter Y sejajar dengan karakter J pada teks, terjadi ketidakcocokkan. karena pada perbandingan pertama terjadi kecocokkan maka nilai pergeseran selanjutnya di ambil dari tabel 3.54 ztGs dengan karakter A, maka pergeseran sejauh satu karakter.

Percobaan ke-9:

I	43	44	45	46	47	48	49	**50**	51	52	53	54	55	56	57	58	59	60	...
Teks	A		S	U	N	J	A	**Y**	A		A	T	A	W	A		A	K	...
Pattern		S	U	N	J	A	Y	**A**											

Pergeseran 1 (ztBc[A][Y])

Dari hasil percobaan ke-9, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter Y pada sumber teks. Artinya, pada percobaan ke-9 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh 1 karakter. Pergeseran karakter ini diambil dari tabel 3.43 ztBc dengan baris A dan kolom Y.

Percobaan ke-10:

I	43	44	**45**	**46**	**47**	**48**	**49**	**50**	**51**	52	53	54	55	56	57	58	59	60	...
Teks	A		**S**	**U**	**N**	**J**	**A**	**Y**	**A**		A	T	A	W	A		A	K	...
Pattern			**S**	**U**	**N**	**J**	**A**	**Y**	**A**										

Pergeseran 7 (ztGs[S])

Dari hasil percobaan ke-10, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter A pada sumber teks. Artinya, pada percobaan ke-10 terjadi kecocokan, maka pencocokan dilakukan terhadap karakter sebelum karakter A, yaitu karakter Y, A, J, N, U, S pada *pattern*, bandingkan dengan karakter yang sejajar, terjadi kecocokkan pada semua karakter. *Pattern* ditemukan kemudian algoritma menandai lokasi penemuan *pattern*, dan untuk melanjutkan pencocokan gunakan pergeseran tujuh karakter diambil dari 3.54 ztGs karakter S.

Percobaan ke-11:

I	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	**58**	59	60	...
Teks	A		S	U	N	J	A	Y	A		A	T	A	W	A		A	K	...
Pattern										S	U	N	J	A	Y	**A**			

Pergeseran 7 (ztBc[A][Sp])

Dari hasil percobaan ke-11, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter Sp pada sumber teks. Artinya, pada percobaan ke-11 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh 7 karakter. Pergeseran karakter ini diambil dari tabel 3.43 ztBc dengan baris A dan kolom Sp.

Percobaan ke-12:

I	59	60	61	62	63	64	**65**	66	67	68	69	70	71	72	73	74	75
Teks	A	K	I	N	A		**N**	Y	I		R	O	D	I	A	H	.
Pattern	S	U	N	J	A	Y	**A**										

Pergeseran 7 (ztGs[Sp][N])

Dari hasil percobaan ke-12, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter N pada sumber teks. Artinya, pada percobaan ke-12 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh 7 karakter. Pergeseran karakter ini diambil dari tabel 3.43 ztBc dengan baris Sp dan kolom N.

Percobaan ke-13:

I	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
Teks	A	K	I	N	A		N	Y	I		R	O	D	**I**	A	H	.
Pattern								S	U	N	J	A	Y	**A**			

Pergeseran 7 (ztGs[Sp][N])

Dari hasil percobaan ke-13, dapat dilihat bahwa karakter akhir *pattern* yang dicari yaitu karakter A sejajar dengan karakter I pada sumber teks. Artinya, pada percobaan ke-13 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh 7 karakter. Pergeseran karakter ini diambil dari tabel 3.43 ztBc dengan baris D dan kolom I.

Karena jumlah pergeseran lebih besar dibandingkan dengan sisa karakter pada teks, maka pencocokan selesai.

Kesimpulan

1. Pada tahap pencocokan dengan algoritma *Zhu-Takaoka* di atas, pencocokan dilakukan sebanyak 13 iterasi. Ditemukan pada iterasi ke-10.
2. Jumlah indeks pada teks adalah 75, *pattern* ditemukan pada indeks ke 45-51.
3. Jumlah perbandingan adalah 73, *pattern* ditemukan pada perbandingan ke 45.

3.1.5 Analisis Kompleksitas

Pada analisis kompleksitas ini akan dijelaskan parameter-parameter apa saja yang ingin diketahui dalam membandingkan performansi antara kedua algoritma untuk mendapatkan hasil performansi mana yang lebih optimal.

3.1.5.1 Analisis Kompleksitas Algoritma Reverse Colussi

Pada algoritma *Reverse Colussi* ini parameter yang ingin diketahui yaitu kompleksitas pencocokan kata yang diinput dengan kata yang ada di database.

Untuk mendapatkan hasil dari kompleksitas pencocokan kata yang diinputkan dengan kata yang ada di database yaitu menggunakan pencocokan dari tabel rcBc dan rcGs. Berikut *Pseudocode* algoritma pencocokan *Reverse Colussi*.

```
Procedure RC(input *x: array of char, input m: integer,
             input *y: array of char, input n: integer){
{ IS : pencocokan String dengan algoritma Reverse Colussi
  FS : keluaran yang diharapkan hasil dari pencocokan }

Kamus
  i, j, s : integer
  rcBc : array[0..ASIZE][0..XSIZE] of integer
  rcGs : array[0..XSIZE] of integer

Algoritma
  /* Preprocessing */
  preRc(x, m, h, rcBc, rcGs);

  /* Searching */
  j = 0;                                                O(1)
  s = m;                                                O(1)
```

```
  while (j <= n - m)                                          O(n)
    while (j <= n - m and x[m - 1] != y[j + m - 1])           O(n)
      s = rcBc[y[j + m - 1]][s];                              O(1)
      j += s;                                                 O(1)
    endwhile
    for (i=1; i<m and x[h[i]]==y[j + h[i]]; ++i);             O(n)
    endfor
    if (i >= m)                                               O(1)
      OUTPUT(j);                                              O(1)
    endif
    s = rcGs[i];                                              O(1)
    j += s;                                                   O(1)
  endwhile
endprocedure
```

Tabel 3.55 Perhitungan Big-O algoritma *Reverse Coulussi*

Pseudocode	Nilai Big-O
`j = 0;`	O(1)
`s = m;`	O(1)
`while (j <= n - m)`	O(n)
`while (j <= n - m and x[m-1] != y[j + m-1])`	O(n)
`s = rcBc[y[j + m - 1]][s];`	O(1)
`j += s;`	O(1)
`for (i = 1; i < m and x[h[i]] == y[j + h[i]]; ++i);`	O(n)
`if (i >= m)`	O(1)
`OUTPUT(j)`	O(1)
`s = rcGs[i];`	O(1)
`j += s;`	O(1)
Jumlah	**$O(n^3)$**

Berdasarkan hasil perhitungan performansi algoritma *Reverse Colussi* menggunakan notasi Big-O didapat kompleksitas waktu dengan **$O(n^3)$** dengan n adalah ukuran inputan. Yang mempengaruhi nilai kompleksitas waktu dari algoritma *Reverse Colussi* adalah nilai n karena n berpangkat 3.

3.1.5.2 Analisis Kompleksitas Algoritma Zhu-Takaoka

Pada algoritma *Zhu-Takaoka* ini, parameter yang ingin diketahui yaitu kompleksitas waktu yang dibutuhkan dalam satu kali proses.

Untuk mendapatkan hasil dari kompleksitas waktu proses yang dibutuhkan dalam satu kali proses dilakukan perhitungan Notasi *Big-O*. Berikut *Pseudocode* algoritma pencocokan *Zhu-Takaoka*.

```
Procedure ZT(input x : array of char, input m :
             integer,input y : array of char,
             input n : integer)
{IS : pencocokan String dengan algoritma zhu-takaoka
 FS : keluaran yang diharapkan hasil dari pencocokan}

Kamus
  i, j  : integer
  ztBc  : array[0..ASIZE][0..ASIZE] of integer
  bmGs  : array[0..XSIZE] of integer
algoritma
  {preprocessing}
  preZtBc(x, m, ztBc);
  preBmGs(x, m, bmGs);

  {pencocokan}
  j = 0;  O(1)
  while (j <= n - m)                              O(n)
    i ← m - 1;                                    O(1)
    while (i < m and x[i] = y[i + j]; --i)        O(n)
      if (i < 0)                                  O(1)
        OUTPUT(j);                                O(1)
        j ← j + bmGs[0];                          O(1)
      endif
      else if
        j ← j + MAX(bmGs[i],ztBc[y[j + m -
        2]][y[j + m -1]]);                        O(1)
      endif
    endwhile
  endwhile
endprocedure
```

Berdasarkan *pseudocode* dari analisis kasus pencocokan algoritma *Zhu-Takaoka*, maka berikut tabel perhitungan *Big-O* nya:

Tabel 3.56 Perhitungan Big-O algoritma *Zhu-Takaoka*

Pseudocode	Nilai Big-O
j = 0;	O(1)
`while (j <= n - m)`	O(n)
`i ← m - 1;`	O(1)
`while (i < m and x[i] = y[i + j]; --i)`	O(n)
`if (i < 0)`	O(1)
`OUTPUT(j);`	O(1)
`j ← j + bmGs[0];`	O(1)
`j ← j + MAX(bmGs[i],ztBc[y[j + m – 2]][y[j + m -1]]);`	O(1)
Jumlah	**O(n^2)**

Berdasarkan hasil perhitungan performansi algoritma *Zhu-Takaoka* menggunakan notasi Big-O didapat kompleksitas waktu dengan **O(n^2)** dengan n adalah ukuran inputan. Yang mempengaruhi nilai kompleksitas waktu dari algoritma *Zhu-Takaoka* adalah nilai n karena n berpangkat 2.

3.1.6 Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak yang akan dibangun berdasarkan kebutuhan pengguna pada jurnal dan artikel serta hasil observasi. Sepsifikasi kebutuhan perangkat lunak akan dibagi ke dalam dua bagian yaitu SKPL-F (Spesifikasi Kebutuhan Perangkat Lunak Fungsional) dan SKPL-NF (Spesifikasi Kebutuhan Perangkat Lunak Non-Fungsional) berikut ini adalah tabel spesifikasi kebutuhan perangkat lunak pencocokan *string*.

Tabel 3. 57 Spesifikasi Kebutuhan Perangkat Lunak Fungsional

Kode	Kebutuhan
SKPL-F001	Perangkat lunak dapat melakukan pencocokan *string* menggunakan algoritma *Reverse Colussi* dan *Zhu Takaoka* untuk menampilkan waktu yang diperlukan oleh kedua algoritma tersebut menggunakan data yang ada pada sistem.
SKPL-F002	Perangkat lunak dapat melakukan pencocokan *string* menggunakan algoritma *Reverse Colussi* dan *Zhu Takaoka* untuk menampilkan waktu yang diperlukan oleh kedua algoritma tersebut menggunakan data masukan *user*.

Tabel 3. 58 Spesifikasi Kebutuhan Perangkat Lunak Non-Fungsional

Kode	Kebutuhan
SKPL-NF001	Perangkat keras yang digunakan adalah komputer dengan spesifikasi minimal Processor Intel Pentium 4, 1.5 GHz, memori 1024 MB, keyboard, dan mouse.
SKPL-NF002	Bahasa pemrograman yang digunakan adalah c++
SKPL-NF003	Pengguna atau *user* yang menggunakan perangkat lunak ini adalah *user* yang ingin melakukan pencocokan.

3.1.6.1 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional adalah sebuah langkah dimana seorang pembangun perangkat lunak (*software develover*) menganalisis perangkat keras dan perangkat lunak yang dimiliki agar sesuai dengan kebutuhan, sehingga dapat ditentukan kompabilitas aplikasi yang dibangun terhadap sumber daya yang ada.

Setelah melakukan analisis non fungsional, maka dilanjutkan ke langkah berikutnya yaitu menentukan kebutuhan fungsional sistem yang akan dibangun untuk disesuaikan dengan fakta yang ada. Apabila kebutuhan tidak terpenuhi maka sistem yang dibangun tidak akan berjalan dengan baik sesuai yang diharapkan.

Analisis kebutuhan non fungsional yang dilakukan dibagi menjadi dua tahap, yaitu Analisis kebutuhan perangkat keras (*hardware*) dan Analisis kebutuhan perangkat lunak (*software*).

3.1.6.1.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras minimum yang dibutuhkan untuk membangun aplikasi Perbandingan Algoritma *Reverse Colussi* dan *Zhu-Takaoka* ini memiliki spesifikasi sebagai berikut:

1. Processor minimal 1.5 Ghz
2. Harddisk minimal 50 Gb
3. Ram minimal 1 Gb
4. VGA minimal 64 MB
5. Monitor 14”
6. Mouse dan Keyboard

3.1.6.1.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk membangun aplikasi Perbandingan Algoritma *Reverse Colussi* dan *Zhu-Takaoka* ini adalah sebagai berikut:

1. Sistem Operasi Windows 7, Ubuntu 16.04
2. Tools Pendukung: Virtual box, Terminal
3. IDE: Qt Creator, Dev C++
4. Code Editor: Sublime Text 3, Gedit, Notepad++.
5. Web Browser: Mozilla firefox, Chrome.

3.1.6.1.3 Analisis Perangkat pikir

Analisa dan spesifikasi kebutuhan diperlukan agar kemampuan perangkat lunak yang dibangun menjadi jelas. Beberapa analisa dan kebutuhan yang berkaitan dengan perangkat lunak yang akan dibangun nanti yaitu analisa dan kebutuhan pengguna. Adapun karkateristik pengguna pada perangkat lunak pencocokan *string* yang akan dibangun yaitu sebagai berikut.

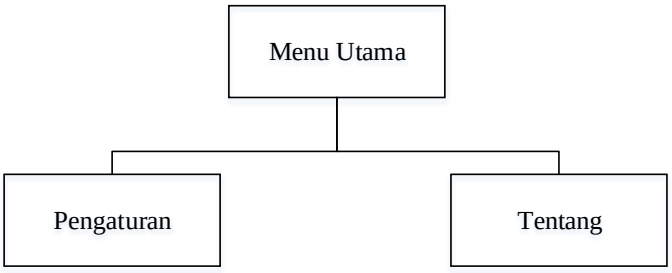
1. Pengguna dapat mengggunakan komputer, minimal mampu menggunakan keyboard sebagai sarana penginputan data dan kata kunci pencocokan. Dan mouse untuk memberikan aksi.
2. Pengguna dapat membaca.

3.2 Perancangan Sistem

Perancangan merupakan penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam suatu kesatuan yang utuh. Tahapan ini meliputi konfigurasi perangkat lunak dan perangkat keras dari suatu sistem. Tahap perancangan desain bertujuan untuk mencari bentuk yang optimal dari aplikasi yang akan dibangun dengan pertimbangan faktor-faktor permasalahan dan kebutuhan yang ada pada sistem seperti yang telah ditetapkan dengan cara mengkombinasikan penggunaan teknologi perangkat keras dan perangkat lunak yang tepat sehingga diperoleh yang optimal dan mudah diimplementasikan. Adapun perancangan sistem dari aplikasi Perbandingan Algoritma *Reverse Colussi* dan *Zhu-Takaoka* ini sebagai berikut.

3.2.1 Perancangan Struktur Menu

Struktur menu adalah bentuk umum dari suatu rancangan program untuk memudahkan pemakai dalam menjalankan program komputer sehingga saat menjalankan program, user atau pengguna aplikasi tidak mengalami kesulitan dalam memilih menu-menu yang diinginkan. Pada perancangan struktur menu aplikasi ini dibuat seperti gambar 3.15.



Gambar 3.7 Struktur Menu

Keterangan:

1. Menu Utama

   Memberikan fasilitas untuk melakukan pencocokan *string* menggunakan algoritma yang akan diuji.

2. Menu Pengaturan

   Untuk memasukan folder master.

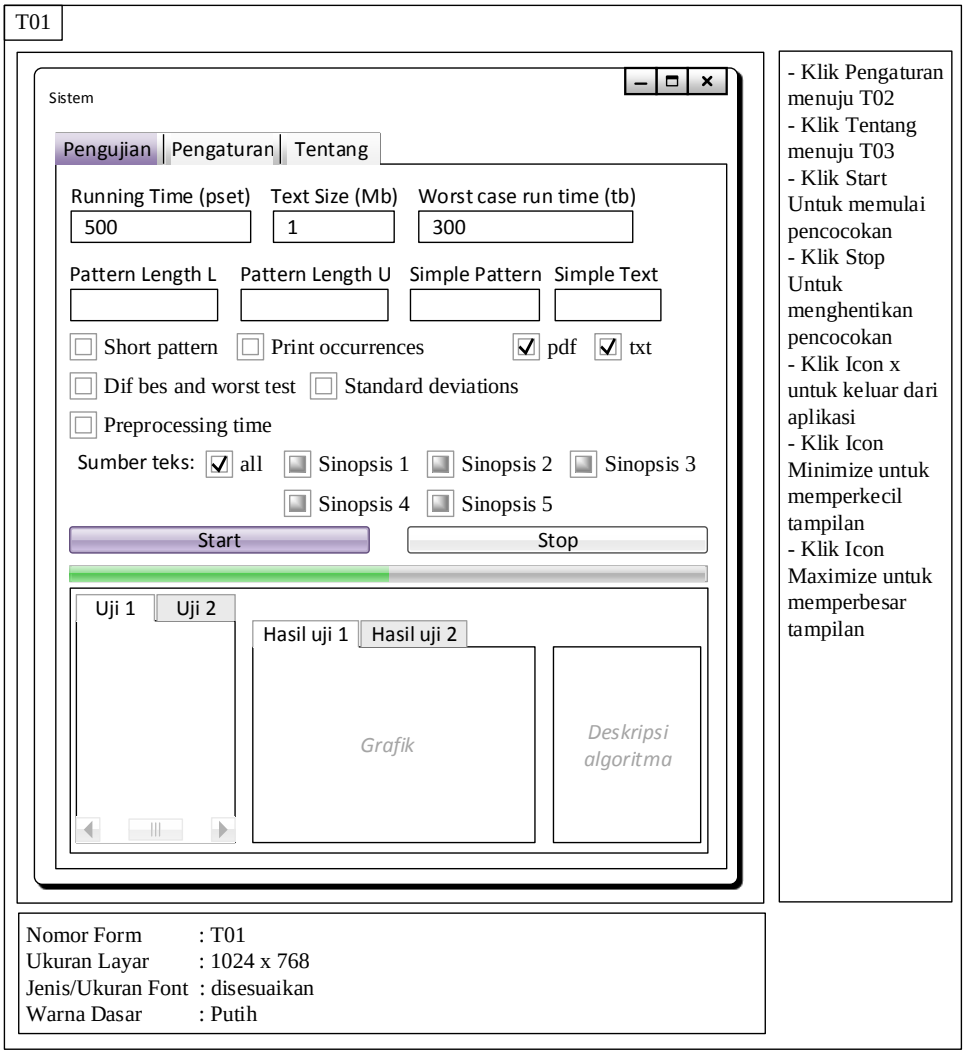
3. Menu Tentang

   Menampilkan informasi tentang profil pembuat.

3.2.2 Perancangan Antarmuka

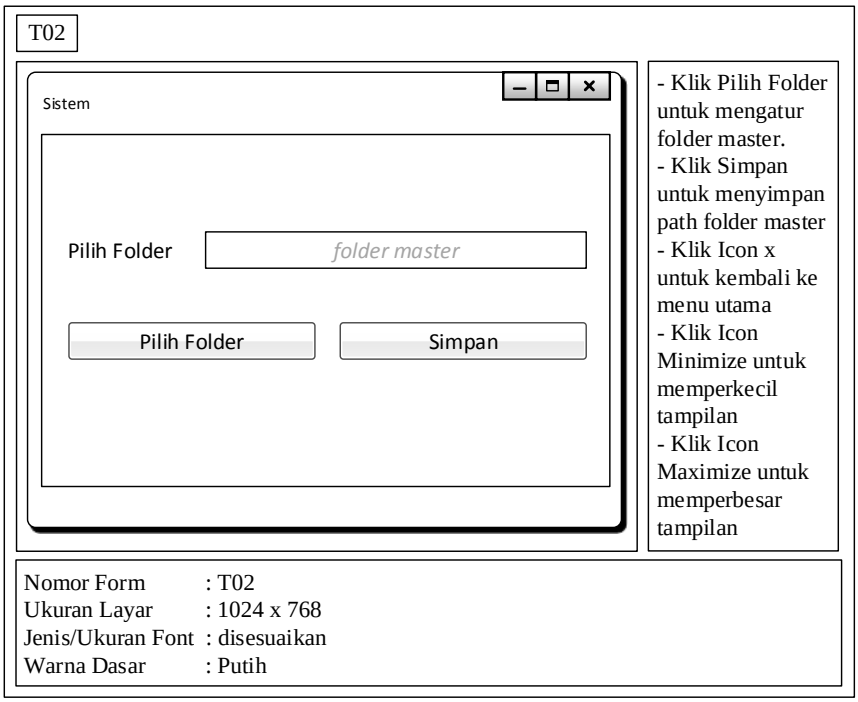
Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi. Pada bagian ini akan dijelaskan perancangan antarmuka mengenai :

1. Perancangan Antarmuka Menu Utama



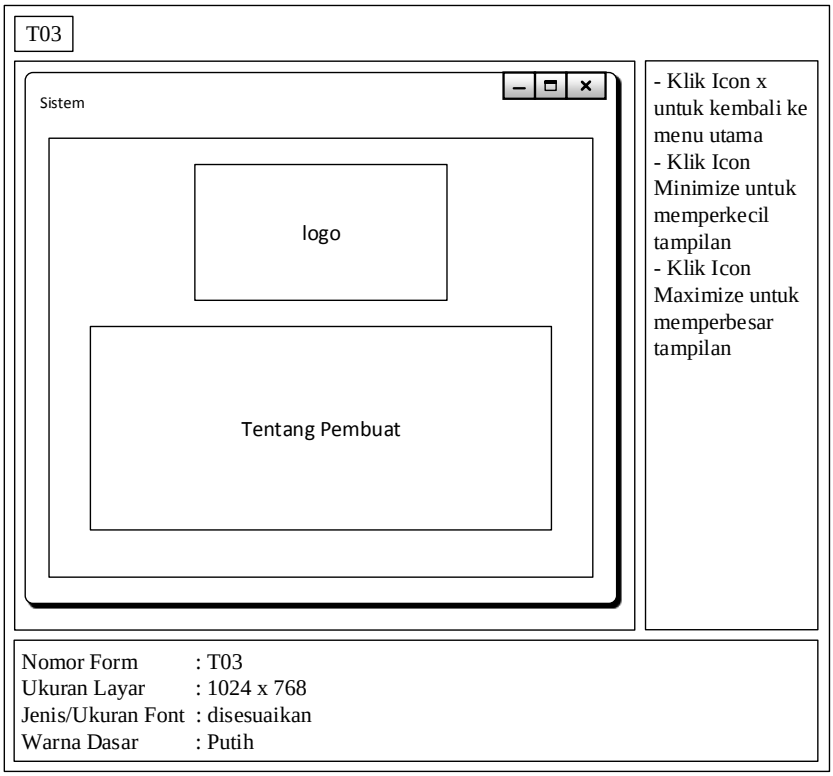
Gambar 3.8 Perancangan Antarmuka Menu Utama

2. Perancangan Antarmuka Menu Pengaturan



Gambar 3. 9 Perancangan Antarmuka Menu Pengaturan

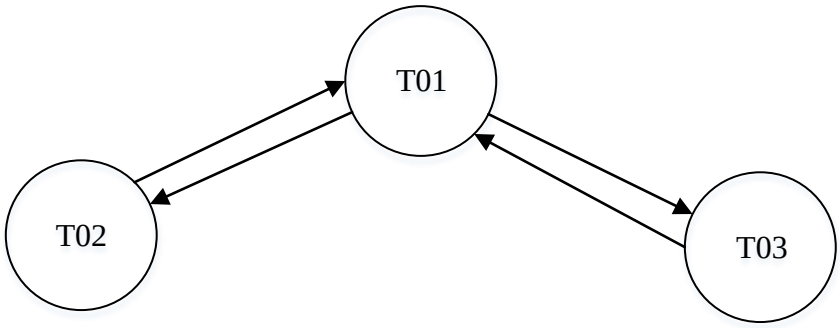
3. Perancangan Antarmuka Menu Tentang



Gambar 3. 10 Perancangan Antarmuka Menu Tentang

3.2.3 Jaringan Semantik

Jaringan Semantik merupakan gambaran pengetahuan grafis yang menunjukan hubungan antar berbagai objek, terdiri dari lingkaran-lingkaran yang dihubungkan dengan anak panah yang menunjukan objek dan informasi tentang objek-objek disebut jaringan semantik. Jaringan semantik pada aplikasi Perbandingan Algoritma *Reverse Colussi* dan *Zhu-Takaoka* bisa dilihat pada *Gambar Jaringan Semantik.*

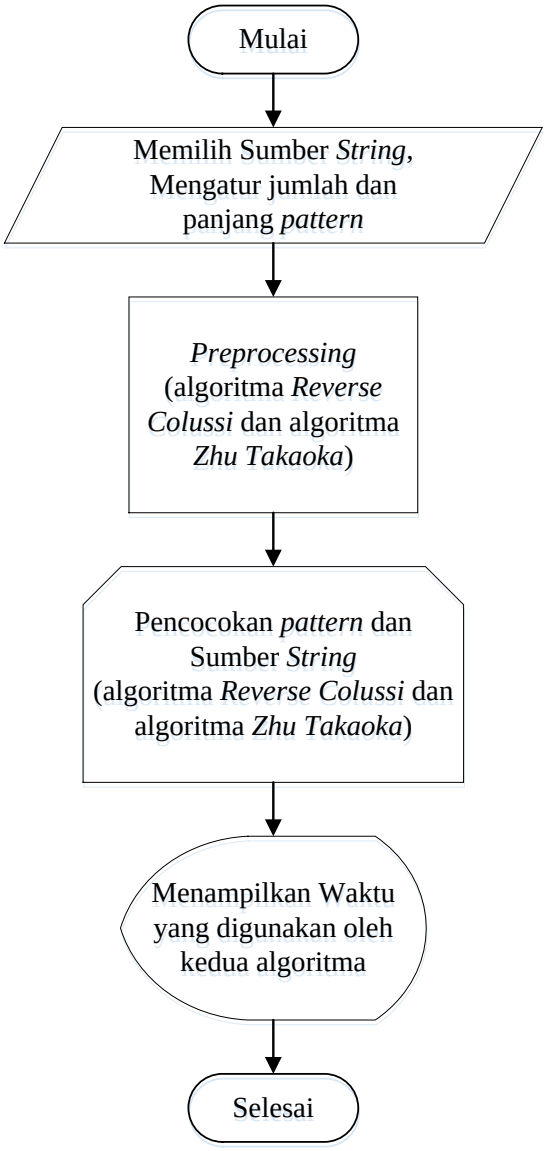


Gambar 3.11 Jaringan Semantik

3.2.4 Perancangan Prosedural

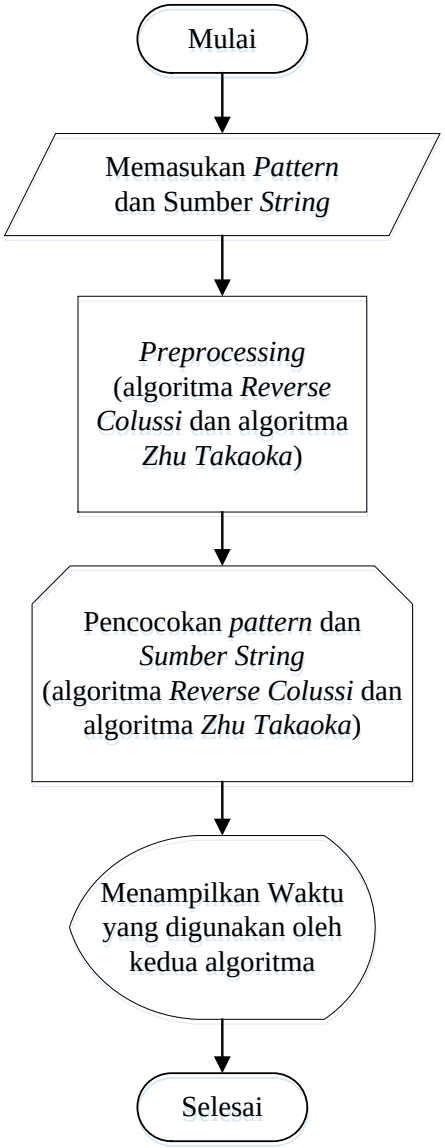
Perancangan prosedural adalah tata cara atau urutan langkah – langkah untuk melakukan suatu proses. Perancangan prosedural menggambarkan algoritma dari prosedur – prosedur yang terdapat pada perangkat lunak yang dibangun. Adapun perancangan prosedural pada perangkat lunak pencocokan *string* adalah sebagai berikut.

1. Prosedur Mencocokan Pola *String* (Data pada sistem)



Gambar 3.12 Prosedur Mencocokan Pola *String* (Data pada sistem)

2. Prosedur Mencocokan Pola *String* (Data masukan user)



Gambar 3.13 Prosedur Mencocokan Pola String (Data masukan user)