Sains dan Komputer (SAINTIKOM)
Vol.18, No.2, Agustus 2019, pp. 139~146
P-ISSN: 1978-6603 E-ISSN : 2615-3475

Perancangan Aplikasi Terjemahan Sandi Morse Dengan Menerapkan Algoritma Brute Force

Azlan, Hafizah, Tugiono
Sistem Informasi, STMIK Triguna Dharma

Article Info

Article history:
Received May 31th, 2019
Revised June 12th, 2019
Accepted Augs 05th, 2019

Keyword:
Pramuka
Terjemahan Sandi Morse
Visual Basic .Net 2008
String Matching
Brute Force

ABSTRACT

Gerakan pramuka Indonesia merupakan wadah para generasi muda untuk mengembangkan minat, bakat, keterampilan dan kedisiplinan. Banyak kegiatan yang dilakukan dalam gerakan pramuka, selain kegiatan baris berbaris ada juga kegiatan dalam menghafal kode-kode pramuka yang dikenal dengan sandi. Sandi pramuka bermacam-macam diantaranya adalah sandi morse. Banyak generasi muda yang sudah mempelajari sandi morse tetapi masih sulit untuk memahami dan menerjemahkan sandi morse yang berisikan kalimat-kalimat panjang. Untuk itu dibutuhkan sebuah aplikasi yang dapat membantu dalam menerjemahkan sandi morse tersebut.Aplikasi tersebut berupa aplikasi komputer yang mampu menerjemahkan sandi morse dalam bentuk teks, dengan adanya aplikasi ini mudah-mudahan dapat berguna dalam memahami sandi morse dan mampu memantapkan serta kejelian dalam menteremahkan sandi morse. Dalam skripsi ini akan dibahas tentang cara menciptkan aplikasi terjemahan sandi morse dengan algoritma string matching. Aplikasi dibangun dengan bahasa pemrograman microsoft visual studio 2008.

First Author
Nama :Azlan
Kantor :STMIK Triguna Dharma
Program Studi :SistemInformasi
E-Mail :Azlansaja19@gmail.com

1. PENDAHULUAN

Gerakan pramuka Indonesia merupakan wadah para generasi muda untuk mengembangkan minat, bakat, keterampilan dan kedisiplinan. Pengaruh gerakan pramuka bagi generasi muda sangat besar dalam pengembangan bakat, dan minat serta keterampilan dan juga kedisiplinannya secara langsung atau tidak langsung. Jika dahulu kala, pramuka identik dengan kegiatan baris berbaris, maka sekarang pramuka sudah mulai membentuk image bahwa pramuka adalah kelompok pemuda dan pelajar yang belajar untuk mandiri oleh dirinya sendiri. Salah satu kegiatan penting dari pramuka adalah menghafal kode-kode atau biasanya dikenal dengan istilah sandi.

Sandi pramuka ini bermacam-macam seperti sandi morse, sandi semaphore, sandi rumput, sandi tangan dan sandi-sandi lain. Saat ini generasi muda yang sudah mempelajari dan menghapal sandi morse, akan tetapi masih sulit untuk menerjemahkan sandi morse yang berisikan kalimat-kalimat panjang, oleh karena itu untuk menerjemahkan sandi morse harus memiliki konsentrasi dan harus fokus. Seperti yang di ketahui sandi morse terdiri dari tanda titik dan tanda garis yang membuat bingung membedakannya, seperti huruf A yang bertanda ._ dan huruf N dengan tanda _., kedua huruf ini merupakan kebalikan dari masing-masing sandi morse A dan N, huruf A diawali tanda titik dan diakhiri tanda garis, begitu juga sebaliknya huruf N diawali tanda garis dan diakhiri tanda titik. Untuk menerjemahkan sandi morse yang berisikan kalimat-kalimat panjang dibutuhkan sebuah aplikasi terjemahan sandi morse.

Aplikasi terjemahan sandi morse dalam menerjemahkan sandi morse ke dalam bentuk teks maka dilakukan pencarian dan pencocokan karakter atau string sandi morse yang di input atau yang ingin diterjemahkan dengan karakter sandi morse yang ada pada listing program, adapaun pencarian dan pencocokan karakter tersebut hanya terdiri dari tanda garis dan tanda titik. Sebagi contoh menerjemahkan sandi morse dengan karakter -. yaitu sebagai karakter yang di input dan sandi morse dengan karakter -.- -. --

sebagai karakter yang ada pada listing program. Setelah pencarian dan pencocokan karakter menghasilkan output makan hasil pencarian dan pencocokan di konversi menjadi alfabet.

Ada beberapa algoritma yang dapat digunakan dalam pencarian dan pencocokan string diantaranya adalah brute force. Algoritma brute force merupakan algoritma pencarian dan pencocokan string yang sangat detail, kerena semua teks yang dicari akan di telusuri karakter perkarakter tanpa ada yang terlewatkan. Proses pencarian dan pencocokan string algoritma brute force dilakukan mulai dari kiri ke kanan. Oleh karena itu penulis menerapkan algoritma brute force untuk merancang aplikasi terjemahan sandi morse.

2. METODE PENELITIAN

Dalam mengembangkan aplikasi ini digunakan metodologi RUP (Rational Unified Process). Rational Unified Process merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai praktek terbaik yang terdapat dalam industri pengembangan perangkat lunak. Untuk membuat integrasi baru dalam bahasa pemodelan antar tool dan proses dalam Rational Unified Process ini menggunakan UML (Unified Modeling Language). Adapun 4 tahapan kerja dari Rational Unified Process sebagai berikut:

1. Inception (tahap analisis), pada tahap ini dilakukan perencanaan sistem yang akan dibangun dengan cara menentukan terlebih dahulu permasalahan yang dihadapi oleh pengguna, menentukan batasan ruang lingkup permasalahan dan kemudian dilakukan identifikasi kebutuhan apa saja yang diperlukan oleh pengguna terhadap permasalahan yang dialami dalam menunjang pengembangan sistem ini.
2. Elaboration (tahap desain), merupakan tahap bagi para pengembang untuk melakukan desain secara lengkap berdasarkan hasil analisis di tahap inception.
3. Construction (tahap implementasi dan pengujian), pada tahap ini penulis melakukan pemeriksaan kembali dari tahap inception dan elaboration, dan kemudian akan diimplementasikan hasil desain dan melakukan pengujian hasil implementasi.
4. Preparation Of Report (tahap Penyusunan Laporan), pada tahap ini dilakukan penyusunan hasil penelitian yang telah dilakukan ke dalam sebuah bentuk laporan.

3. ANALISA DAN HASIL

3.1. Analisa Penerjemahan Sandi Morse dengan Menggunakan Komputer

Untuk menerjemahkan sandi morse ke dalam bentuk teks maka dilakukan pencarian dan pencocokan karakter sandi morse, dimana setiap karakter akan dipisah dengan adanya tanda spasi yang brearti tanda spasi akan dijadikan sebagai kode pemisah karakter. Pencarian dan pencocokan akan selesai dan akan mencetak output setiap menjumpai spasi. Jika pencarian dan pencocokan hanya menemukan tanda titik dan tanda garis maka pencarian terus dilanjutkan tanpa harus mencetak output hingga menemukan tanda spasi.

Misalnya terdapat sandi morse "… .- -.-- .-", dimana terdapat empat karakter dan tiga tanda spasi. Maka pencarian dan pencocokan akan dilakukan sebanyak empat kali menurut banyak karakter yang telah dipisah tanda spasi, pencarian dan pencocokan dilakukan dari sebelah kiri kekanan. Dan pencarian dan pencocokan pertama sandi morse "…" akan menghasilkan alfabet S, sandi morse ".-" akan menghasilkan alfabet A, sandi morse "-.--" akan menghasilkan alfabet Y, sandi morse ".-" akan menghasilkan alfabet A. Dari hasil pencarian dan pencocokan sandi morse "… .- -.-- .-" maka mendapatkan output "SAYA".

Sedangkan tanda garis miring (/) didalam sandi morse akan mencetak output tanda spasi didalam teks, sebagai contoh sandi morse ".- -… / -.-." akan mencetak output "AB C". jadi kesimpulannya tanda spasi didalam sandi morse merupakan pemisah karakter untuk menerjemahkan sandi morse ke teks dan tanda garis miring didalam sandi morse merupakan spasi didalam teks.

3.2. Analisa Implementasi Algoritma Brute Force untuk Menerjemahkan Sandi Morse

Aplikasi terjemahan sandi morse yang menerjemahkan sandi morse dengan menggunakan komputer akan dirancang menggunakan algoritma *brute force* untuk menyelesaikan masalah pencarian dan pencocokan karakter. Pencarian dan pencocokan karakter yang dilakukan yaitu antara sandi morse yang di *input* dengan sandi morse pada *listing* program yang berbentuk matriks. Setelah pencarian dan pencocokan dijumpai maka hasil pencarian dan pencocokan akan di *konversi* kedalam alfabet. Pada *listing* program terdapat satu matriks yang berordo 2x48 dimana didalam matriks terdapat 2 kolom dan 48 baris. Kolom pertama berisikan sandi morse, sedangkan kolom kedua berisikan huruf, tanda baca dan angka.

Untuk menyelesaikan permasalah pencarian dan pencocokan sandi morse ini maka algoritma *brute force* diharapkan dapat menyelesaikan permasalahan tersebut, dimana algoritma *brute force* akan memudahkan pencarian dan pencocokan sandi morse pada aplikasi terjemahan sandi morse. Dalam algoritma pencarian *string* termasuk algoritma *brute force* dikenal dengan yang namanya teks dan *pattern.* Teks dalam pencarian *string* dilambangkan dengan "y" dan *pattern* dilambangkan dengan "x". Pada perancangan aplikasi

terjemahan sandi morse dengan menggunakan algoritma *brute force*, yang digunakan sebagai *pattern* adalah kata yang ingin dicari pada *form* pengetikan kata atau *input* dan yang digunakan sebagai teks adalah matriks kolom pertama yang ada pada *listing* program. Panjang *pattern* selalu harus lebih kecil dari teks. Berikut ini adalah cara kerja dari algortima *brute force* adalah sebagai berikut:

1. Langkah algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.
2. Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :
   a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
   b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah ke-2 sampai *pattern* yang berada di ujung teks.

Contoh penggunaan algoritma *Brute Force* untuk pencarian *pattern* dalam teks martiks berordo 2x3 :

Teks = $\begin{bmatrix} .- & \mathbf{A} \\ -. & \mathbf{N} \\ -.- & \mathbf{K} \end{bmatrix}$

Pattern = -.- .- -. .- -.

Pattern dipisahkan terlebih dahulu dengan adanya tanda spasi, dimana terdapat lima *pattern* yaitu *Pattern* 1 adalah (-.-), *Pattern* 2 adalah (.-), *Pattern* 3 adalah (-.) , *Pattern* 4 adalah (.-) dan *Pattern* 5 adalah (-.).

Penyelesaian :

a. Penyelesaian *Pattern* 1:
   1. Teks (Kolom 1 Baris 1) dan *Pattern* 1 (-.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Teks	.	-	
Pattern 1	-	.	-
Indeks	1	2	3

   Dikarena *pattern* lebih panjang dari teks maka pencarian dihentikan, dan dilanjutkan ke teks (kolom 1 baris 2) dan *Pattern* 1 (-.-).
   2. Teks (Kolom 1 Baris 2) dan *Pattern* 1 (-.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Teks	-	.	
Pattern 1	-	.	-
Indeks	1	2	3

   Dikarena *pattern* lebih panjang dari teks maka pencarian dihentikan, dan dilanjutkan ke teks (kolom 1 baris 3) dan *Pattern* 1 (-.-).
   3. Teks (Kolom 1 Baris 3) dan *Pattern* 1 (-.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	-
Pattern 1	-	.	-
Indeks	1	2	3

   Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :
   a.Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
   b.Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.
   Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi b yaitu semua *pattern* dan teks cocok dan *pattern* juga sudah berada diujung teks serta teks yang ditelusuri juga sudah habis maka pencarian dan pencocokan dihentikan. Selanjutnya pencarian dan pencocokan *pattern* 2.

b. Penyelesaian *Pattern* 2:
   1. Teks (Kolom 1 Baris 1) dan *Pattern* 2 (.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	.	-	
Pattern 2	.	-	
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi b yaitu semua *pattern* dan teks cocok.

2. Teks (Kolom 1 Baris 2) dan *Pattern* 2 (.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	
Pattern 2	.	-	
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi a yaitu *pattern* dan teks tidak cocok.

3. Teks (Kolom 1 Baris 3) dan *Pattern* 2 (.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	-
Pattern 2	.	-	
Indeks	1	2	3

Pencarian dan Pencocokan Ke-2			
Teks	-	.	-
Pattern 1		.	-
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 dan ke-2 ini memenuhi kondisi a yaitu *pattern* dan teks tidak cocok dan *pattern* juga sudah berada diujung teks serta teks yang ditelusuri juga sudah habis maka pencarian dan pencocokan dihentikan. Selanjutnya pencarian dan pencocokan *pattern* 3.

c. Penyelesaian *Pattern* 3:
   1. Teks (Kolom 1 Baris 1) dan *Pattern* 3 (-.) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	.	-	
Pattern 3	-	.	
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi a yaitu *pattern* dan teks tidak cocok.

2. Teks (Kolom 1 Baris 2) dan *Pattern* 3 (-.) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	
Pattern 3	-	.	
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi b yaitu semua *pattern* dan teks cocok.

3. Teks (Kolom 1 Baris 3) dan *Pattern* 3 (-.) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	-
Pattern 3	-	.	
Indeks	1	2	3

Pencarian dan Pencocokan Ke-2			
Teks	-	.	-
Pattern 3		-	.
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 dan ke-2 ini memenuhi kondisi a yaitu *pattern* dan teks tidak cocok dan *pattern* juga sudah berada diujung teks serta teks yang ditelusuri juga sudah habis maka pencarian dan pencocokan dihentikan. Selanjutnya pencarian dan pencocokan *pattern* 4.

d. Penyelesaian *Pattern* 4:

1. Teks (Kolom 1 Baris 1) dan *Pattern* 4 (.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	.	-	
Pattern 4	.	-	
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi b yaitu semua *pattern* dan teks cocok.

2. Teks (Kolom 1 Baris 2) dan *Pattern* 4 (.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	
Pattern 4	.	-	
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi a yaitu *pattern* dan teks tidak cocok.

3. Teks (Kolom 1 Baris 3) dan *Pattern* 4 (.-) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	-
Pattern 4	.	-	
Indeks	1	2	3

Pencarian dan Pencocokan Ke-2			
Teks	-	.	-
Pattern 4		.	-
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 dan ke-2 ini memenuhi kondisi a yaitu *pattern* dan teks tidak cocok dan *pattern* juga sudah berada diujung teks serta teks yang ditelusuri juga sudah habis maka pencarian dan pencocokan dihentikan. Selanjutnya pencarian dan pencocokan *pattern* 5.

e. Penyelesaian *Pattern* 5:
   1. Teks (Kolom 1 Baris 1) dan *Pattern* 5 (-.) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	.	-	
Pattern 5	-	.	
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi a yaitu *pattern* dan teks tidak cocok.

2. Teks (Kolom 1 Baris 2) dan *Pattern* 5 (-.) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	
Pattern 5	-	.	
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 ini memenuhi kondisi b yaitu semua *pattern* dan teks cocok.

3. Teks (Kolom 1 Baris 3) dan *Pattern* 5 (-.) dimana algoritma *brute force* dimulai dengan mencocokkan *pattern* dari awal teks.

Pencarian dan Pencocokan Ke-1			
Teks	-	.	-
Pattern 5	-	.	
Indeks	1	2	3

Pencarian dan Pencocokan Ke-2			
Teks	-	.	-
Pattern 5		-	.
Indeks	1	2	3

Algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter yang ada pada teks yang berkesesuaian mulai dari kiri ke kanan, sampai salah satu kondisi berikut terpenuhi :

a. Karakter yang ada pada *pattern* dan pada teks yang dibandingkan tidak cocok.
b. Semua karakter pada *pattern* cocok. Kemudian algoritma akan memberitahu penemuan di posisi ini.

Algoritma kemudian terus menggeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah diatas sampai *pattern* yang berada di ujung teks. Dengan demikian pada pencarian dan pencocokan ke-1 dan ke-2 ini memenuhi kondisi a yaitu *pattern* dan teks tidak cocok dan *pattern* juga sudah berada diujung teks serta teks yang ditelusuri juga sudah habis maka pencarian dan pencocokan dihentikan.

Hasil dari pencarian dan pencocokan yang sudah dilakukan pada matriks berordo 2x3 sebagai teks dan -.- .- -. .- -. sebagai *pattern* ditemukan pencocokan pada *pattern* 1 dan teks kolom 1 baris 3, *pattern* 2 dan teks kolom 1 baris 1, *pattern* 3 dan teks kolom 1 baris 2, *pattern* 4 dan teks kolom 1 baris 1 serta *pattern* 5 dan teks kolom 1 baris 2 . Setelah pencarian dan pencocokan temukan maka *output* yang akan didapat adalah sebagai berikut :

1. Hasil pencarian dan pencocokan pada *pattern* 1 dan teks kolom 1 baris 3 adalah -.- dikonversikan ke

alfabet yaitu nilai dari kolom 2 baris 3 yang dijadikan sebagai *output* adalah K.
2. Hasil pencarian dan pencocokan pada *pattern* 2 dan teks kolom 1 baris 1 adalah .- dikonversikan ke alfabet yaitu nilai dari kolom 2 baris 1 yang dijadikan sebagai *output* adalah A.
3. Hasil pencarian dan pencocokan pada *pattern* 3 dan teks kolom 1 baris 2 adalah -. dikonversikan ke alfabet yaitu nilai dari kolom 2 baris 2 yang dijadikan sebagai *output* adalah N.
4. Hasil pencarian dan pencocokan pada *pattern* 4 dan teks kolom 1 baris 1 adalah .- dikonversikan ke alfabet yaitu nilai dari kolom 2 baris 1 yang dijadikan sebagai *output* adalah A.
5. Hasil pencarian dan pencocokan pada *pattern* 5 dan teks kolom 1 baris 2 adalah -. dikonversikan ke alfabet yaitu nilai dari kolom 2 baris 2 yang dijadikan sebagai *output* adalah N.

Setelah hasil konversi didapatkan maka dilakukan penggabungan *output* dimana hasil akhir dari *pattern* -.- .- -. .- -. Adalah KANAN.

4. KESIMPULAN

Kesimpulan dari suatu penelitian merupakan penjelasan tentang hasil akhir yang menguraikan pencapaian dari tujuan penelitian. Dari hasil penulisan dan analisa dari bab-bab sebelumnya, maka dapat diambil kesimpulan-kesimpulan, dimana kesimpulan-kesimpulan tersebut kiranya dapat berguna bagi para pembaca, sehingga penulisan skripsi ini dapat lebih bermanfaat. Adapun kesimpulan-kesimpulan tersebut adalah sebagai berikut :

1. Sandi morse dapat diterjemahkan dengan menggunakan komputer yaitu dengan cara membandingkan karakter sandi morse yang di input dengan sandi morse yang ingin diterjemahkan.
2. Algoritma brute force dapat diterapkan pada aplikasi terjemahan sandi morse untuk mempermudah proses pencarian dan pencocokan karakter sandi morse.
3. Aplikasi terjemahan sandi morse telah selesai dirancang dengan menggunakan bahasa pemrograman Microsoft Visual Studio 2008 dan telah dapat dijalankan untuk menerjemahkan sandi morse.

REFERENSI

[1] Ahmad Syukri dan Amran, Pengurusan Teknologi, University Teknologi Malaysia, Malaysia, 2005
[2] HM. Jogiyanto, Sistem Teknologi Informasi, Andi, Yogyakarta, 2003
[3] Kadir Abdul, Pengenalan Sistem Informasi, Andi, Yogyakarta, 2003
[4] Singgih Daru Kuncara., M.R. Nababan., Sri Samiati.(2013). Analisa Terjemahan Tindak Tutur Direktif Pada Novel The Godfather dan Terjemahannya Dalam Bahasa Indonesia, 1, 1-20.
[5] Trianto Juliatmojo., Eko Aribowo.(2013). Pembelajaran Sandi Morse dan Sandi Semaphore Dalam Bentuk Simulasi Berbasis Multimedia, 1, 2338-5197.
[6] Antonius Rachmat C, Algoritmadan Pemrograman dengan Bahasa C, Andi, Yogyakarta, 2010
[7] Riyanarto Sarno, Yeni Anistyasari, dan Rahimi Fitri, Semantic Search, Andi, Yogyakarta, 2012
[8] A.S. Rosa dan M. Shalahuddin, Rekayasa Perangkat Lunak, Modula, Yogyakarta, 2011

Ketut Darmayuda, Pemograman Aplikasi Database dengan Microsoft Visual Basic.NET 2008, Informatika, Bandung, 2010