Konversi Romaji ke Hiragana dengan Algoritma Pencocokan String

Venny Larasati Ayudiani – 13513025
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
vennylaras@students.itb.ac.id

***Abstrak*—Romaji merupakan romanisasi ke alfabet latin dari sistem penulisan Bahasa Jepang, baik kanji maupun kana. Romanisasi dari karakter Bahasa Jepang dibutuhkan untuk menuliskan kata Bahasa Jepang kepada orang yang tidak bisa membaca kanji ataupun kana, contohnya adalah pelajar yang baru mempelajari Bahasa Jepang. Pada makalah ini penulis hendak merumuskan algoritma pencocokan string sederhana untuk mengonversi romaji menjadi salah satu silabis kana, yaitu hiragana.**

***Kata kunci*— *pattern matching*, pencocokan string, konversi, romaji, hiragana.**

I. PENDAHULUAN

Pada era globalisasi ini, menguasai Bahasa Inggris saja tidak cukup karena Bahasa Inggris sudah bukan menjadi nilai tambah, melainkan sebuah kebutuhan. Salah satu Bahasa yang umum dipelajari oleh pelajar di Indonesia diantaranya adalah Bahasa Jepang. Bahasa Jepang memiliki sistem penulisan yang berbeda dengan Bahasa Indonesia maupun Bahasa Inggris yang menggunakan alfabet latin. Bahasa Jepang menggunakan karakter kanji dan kana dalam sistem penulisannya. Karakter-karakter ini dapat terasa asing apabila dipelajari untuk pertama kalinya. Untuk membaca karakter-karakter tersebut, dibutuhkan romanisasi dari kanji dan kana menjadi alfabet latin yang disebut sebagai romaji.

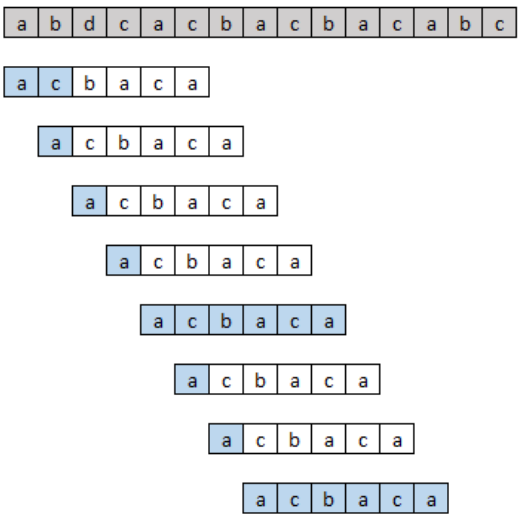
Selain dibutuhkannya romanisasi dari kanji dan kana, dibutuhkan juga untuk mempelajari kanji dan kana dari romajinya. Karena selain mengerti arti kata, hendaknya para pelajar yang memperlajari Bahasa Jepang juga bisa membaca dan menulis dalam sistem penulisan Bahasa Jepang. Pada awal pembelajaran Bahasa Jepang, biasanya yang pertama dipelajari adalah kana karena kana merupakan dasar. Kata yang dituliskan dalam kanji juga bisa dituliskan dengan kana. Oleh karena itu, penulis membuat algoritma sederhana untuk mengonversi romaji menjadi salah satu dari silabis kana yaitu hiragana.

II. LANDASAN TEORI

Pencarian string dalam sebuah teks biasa disebut dengan pencocokan string (*string matching* atau *pattern matching*). Dalam pencocokan string, tulisan di mana pencarian dilakukan dinamakan sebagai teks dengan panjang n. Lalu, string yang dicari disebut sebagai *pattern* dengan panjang m, di mana *pattern* memenuhi syarat panjang m kurang dari panjang n (m<n). Terdapat beberapa algoritma yang dapat digunakan untuk melakukan pencocokan string yaitu algoritma Brute Force, algoritma Knuth-Morris-Pratt, dan algoritma Boyer-Moore.

2.1 Algoritma Brute Force

Brute force merupakan sebuah pendekatan dalam pemecahan masalah yang penyelesaiannya lempeng (*straight forward*)[1]. Pencocokan string *dengan algoritma* brute force dapat dilakukan dengan mencocokkan satu per satu karakter yang ada pada *pattern* dengan karakter yang ada di teks. Jika ditemukan ketidakococokan, geser *pattern* sejauh satu karakter, lalu ulangi lagi pencocokan dari karakter pertama pada *pattern*.



Gambar 1. Ilustrasi pencocokan string dengan algoritma *brute force.*

Gambar di atas merupakan ilustrasi dari pencocokan string menggunakan algoritma Boyer-Moore. Karakter dengan warna biru merupakan karakter pada *pattern* yang dibandingkan dengan karakter pada teks.

Kasus terburuk pada pencocokan string menggunakan algoritma brute force adalah ketika karakter awal pada

pattern cocok dengan teks, namun karakter terakhirnya tidak cocok dan kecocokan baru ditemukan pada karakter terakhir dari teks. Kompleksitas algoritmanya pada kasus terburuk adalah O(mn).

Contoh:

teks	: aaaaaaaaaaaaaaaah
pattern	: aah

Kasus terbaik dapat dicapai pada kondisi karakter pertama pada *pattern* tidak pernah mengalami kecocokan dengan teks sebelum *pattern* ditemukan. Komplekstitas algoritma pada kasus ini adalah O(n).

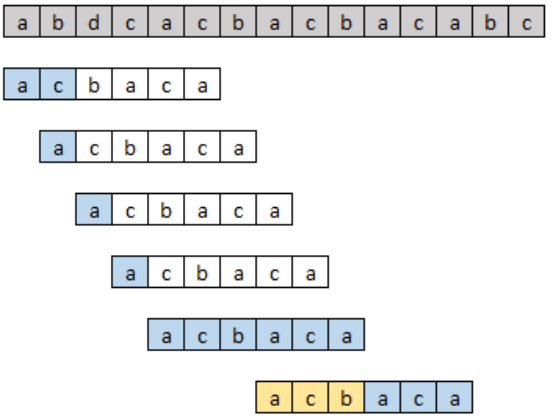
Contoh:

teks	: Pada hari Minggu saya membeli kacang
pattern	: kacang

Kasus yang paling sering terjadi (kasus rata-rata) yaitu ketikaada beberapa karakter yang cocok pada *pattern* da nada beberapa yang cocok. Kompleksitas algoritmanya adalah O(m+n). Algoritma *brute force* akan optimal ketika jumlah alfabet besar dan akan kurang optimal ketika jumlah alfabet kecil.

2.2 Algoritma Knuth-Morris-Pratt

Pencocokan string pada algoritma Knuth-Morris-Pratt (KMP) dilakukan dari kiri. Algoritma KMP merupakan perbaikan dari algoritma *brute force*[1]. Perbaikan dalam algoritma ini terdapat pada penggeseran *pattern* jika terjadi ketidakcocokan. Pergeseran *pattern* ditentukan oleh sebuah fungsi yang dinamakan fungsi pinggiran. Fungsi pinggiran merupakan fungsi yang mengembalika jumlah awalan (*prefix*) dari *pattern* yang juga merupakan *suffix*.



Gambar 2. Ilustrasi pencocokan string dengan algoritma Knuth-Morris-Pratt.

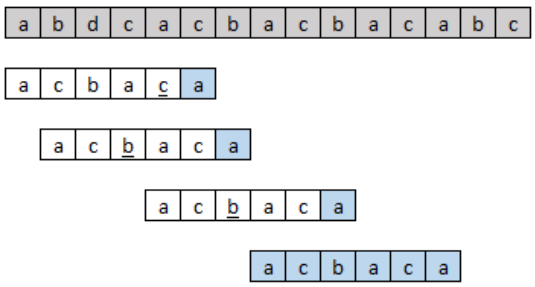
Gambar di atas merupakan ilustrasi dari pencocokan string menggunakan algoritma KMP. Karakter dengan warna biru merupakan karakter pada *pattern* yang dibandingkan dengan karakter pada teks. Karakter dengan warna kuning merupakan karakter yang sudah tidak perlu dibandingkan lagi karen adanya fungsi pinggiran.

Kompleksitas yang dibutuhkan untuk mencari fungsi pinggiran adalah O(m) dan kompleksitas yang dibutuhkan untuk pencocokan string adalah O(n). Sehingga, kompleksitas dari algoritma KMP ini adalah O(m+n). Kasus terbaik akan terjadi jika jumlah alfabet pada teks kecil.

2.3 Algoritma Boyer-Moore

Algoritma Boyer-Moore melakukan pencocokan string dari kanan (dari belakang). Terdapat dua karakeristik dari algoritma Boyer-Moore[1]. Pertama yaitu teknik *looking-glass* yang merupakan teknik pencocokan yang dilakukan dari karakter paling belakang pada *pattern*. Kedua, yaitu teknik character-jump yang dilakukan apabila terjadi ketidakcocokan pada karakter. Terdapat tiga kondisi yang menentukan *character jump*:

1. Karakter pada teks ditemukan di sisi kiri karakter pada *pattern*. Jika hal ini terjadi, maka geser pattern sampai karakter yang sama yang ada *pattern* bersesuaian posisinya dengan karakter pada teks
2. Karakter pada teks ditemukan di sisi kanan karakter pada *pattern*. Apabila hal ini terjadi, maka geser *pattern* sejauh satu karakter.
3. Karakter pada teks tidak ditemukan pada *pattern*. Jika konsisi ini terjadi, geser *pattern* hingga melewati karakter yang tidak cocok pada teks.



Gambar 3. Ilustrasi pecocokan string dengan algoritma Boyer-Moore.

Gambar di atas merupakan ilustrasi dari pencocokan string menggunakan algoritma Boyer-Moore. Karakter dengan warna biru merupakan karakter pada *pattern* yang dibandingkan dengan karakter pada teks. Karakter yang digaris bawahi merupakan karakter pada *pattern*yang sama dengan karakter pada teks saat terjadi ketidakcocokan antara *pattern* dengan teks.

Kasus terburuk pada algoritma Boyer-Moore dicapai ketika jumlah alfabet besar, sehingga algoritma ini baik digunakan untuk melakukan pencocokan string pada tulisan biasa dan kurang cocok untuk digunakan pada pencocokan string bilangan biner.

III. SISTEM PENULISAN BAHASA JEPANG

Sistem penulisan Bahasa jepang merupakan gabungan dari dua jenis tipe karakter yaitu kanji dan kana. Kanji merupakan karakter logografis digunakan untuk menuliskan kata-kata asli Jepang atau kata-kata yang memiliki asal-usul dari Tiongkok. Kata-kata tersebut termasuk sebagian besar kata benda, kata kerja, dan nama tempat maupun nama orang.

Kana merupakan karakter silabis yang terdiri dari hiragana dan katakana. Hiragana biasa digunakan untuk kata-kata dari Bahasa Jepang asli maupun komponen-komponen tata bahasa. Sedangkan, katakana biasa digunakan untuk kata-kata dari bahasa asing seperti nama orang asing, nama tempat, istilah dalam dunia sains maupun kata yang diturunkan dari Bahasa Inggris.

Gambar 4. Kanji untuk beberapa kata dalam Bahasa Jepang dengan artinya dalam Bahasa Inggris.[2]

Selain kanji dan kana, terdapat satu lagi sistem penulisan Bahasa Jepang, yaitu *Romanized Japanese* atau yang disebut sebagai romaji. Romaji biasa digunakan oleh orang asing yang sedang belajar Bahasa Jepang maupun oleh orang Jepang asli sebagai masukan ke komputer.

3.1 Hiragana

Hiragana merupakan salah satu komponen dasar dalah sistem penulisan Bahasa Jepang. Karakter hiragana dikategorikan sebalai aksara silabis dan bukan sebagai alfabet karena setiap karakternya merepresentasikan sebuah suku kata dan bukan sebuah konsonan (kecuali untuk ん "n"**)**.

Seperti yang telah disebutkan sebelumnya, hiragana biasa digunakan untuk menuliskan kata-kata asli Bahasa Jepang. Kata yang memiliki karakter kanji dapat dituliskan pula dengan hiragana, tergantung dengan preferensi penulisnya. Selain itu, hiragana biasa digunakan sebagai *furigana*, yaitu ejaan fonetik kanji yang ditulis di bawah karakter kanji sebagai alat bantu untuk belajar. Tabel di bawah ini merupakan karakter dasar dalam hiragana.

Tabel 1. Karakter dasar hiragana[3]

	a	**i**	**u**	**e**	**o**
Ø	あ	い	う	え	お
k	か	き	く	け	こ
s	さ	し	す	せ	そ
t	た	ち	つ	て	と
n	な	に	ぬ	ね	の
h	は	ひ	ふ	へ	ほ
m	ま	み	む	め	も
y	や		ゆ		よ
r	ら	り	る	れ	ろ
w	わ	ゐ		ゑ	を
	ん **(N)**				
Functional marks and diacritics					
っ	ゝ		゛	゜	

Karakter dasar hiragana dapat dimodifikasi dengan cara berikut:

1. Dengan menambah tanda *dakuten* (゛), konsonan k menjadi g, s menjadi z, t menjadi d, ch/sh menjadi j dan h menjadi b.
2. Dengan menambah tanda *hadakuten* (゜), konsonan h menjadi p.
3. Dengan menambah *sokuon* (っ), konsonan setelahnya akan digandakan. Namun, sokuon tidak digunakan untuk konsonan n. Jika ingin menggandakan n, cukup dengan menuliskan ん.
4. Dengan menambah versi kecil dari ya, yu, atau yo (や, ゆ atau よ) setelah hiragana dengan akhirkan i, vokalnya akan diganti dengan ya, yu, atau yo. Contohnya ki (き) ditambah ya kecil (や) menjadi kya (きゃ).

3.2 Romaji

Romaji merupakan karakter Bahasa Jepang yang diromanisasi ke alphabet latin. Seperti yang telah diuraikan di atas, romaji biasa digunakan oleh pelajar asing untuk mempelajari Bahasa Jepang, Selain itu, romaji juga biasa digunakan untuk menuliskan kata dalam Bahasa Jepang yang ditujukan untuk orang yang tidak dapat membaca kanji ataupun kana seperti pada plang nama tempat wisata di Jepang yang biasa dikunjungi oleh turis.

Contoh:
たべまん = tabemasen
ありが゛とう　ございます = arigatou gozaimasu
さようなら = sayounara

Romanisasi disesuaikan dengan bagaimana karakter dibaca sehingga romanisasi dilakukan secara fonetik. Berikut merupakan beberapa pengecualian romanisasi hiragana ke romaji:

1. し dibaca shi, bukan si
2. ち dibaca chi, bukan ti

3. つ dibaca tsu, bukan tu
4. ふ dibaca fu, bukan hu

IV. KONVERSI ROMAJI KE HIRAGANA

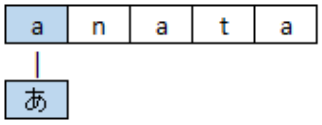
Berdasarkan landasan teori yang telah dipaparkan pada bagian sebelumnya, konversi karakter dari romaji ke hiragana dapat dilakukan dengan menggunakan algoritma pencocokan string. Algoritma yang akan digunakan untuk masalah ini adalah algoritma *brute force*. Walaupun algoritma *brute force* memiliki kompleksitas yang lebih tinggi dari algoritma KMP dan Boyer-Moore, algoritma brute force lebih mudah dimengerti dan lebih mudah untuk diimplementasi.

Menurut penulis, algoritma brute force merupakan solusi yang tepat karena panjang *pattern* tergolong pendek, sesuai dengan panjang romaji per karakter hiragana, yaitu antara satu sampai tiga karakter. Penggunaan algoritma KMP tidak akan efektif karena *pattern* tidak memiliki awalan (*prefix*) dan akhiran (*suffix*) yang sama. Hal ini menyebabkan fungsi pinggiran pada algoritma KMP tidak akan memiliki kegunaan yang signifikan. Penggunaan algoritma Boyer-Moore juga kurang tepat karena pada masalah ini pencocokan karakter lebih sesuai jika dilakukan dari depan ke belakang. Penggunaan algoritma *brute force* memungkinkan pencocokan per karakter secara kondisional dimulai dari karakter paling awal.

Proses konversi romaji ke hiragana memiliki tiga kondisi, yaitu untuk pattern yang memiliki panjang satu karakter (vokal atau konsonan ‘n’), dua karakter (konsonan-vokal), atau tiga karakter (konsonan-konsonan-vokal). Langkah-langkah untuk mengetahui panjang romaji dari setiap hiragana adalah sebagai berikut:

1. Mengecek karakter pertama.
   (i) Jika karakter pertama merupakan huruf vokal, maka *pattern* romaji memiliki panjang 1 karakter. Karakter berikutnya merupakan hiragana baru.

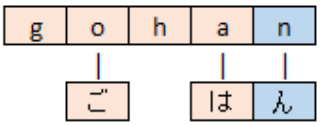
   Contoh:

   (ii) Jika karakter pertama merupakan konsonan ‘n’ lanjut ke langkah kedua.
   (iii) Jika karakter pertama merupakan konsonan selain ‘n’, lanjut ke langkah ketiga.

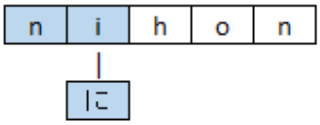
2. Mengecek karakter kedua jika karakter pertama sama dengan ‘n’.
   (i) Jika karakter kedua merupakan spasi, blank, atau konsonan, maka *pattern* romaji memiliki panjang 1 karakter dan karakter hiragananya adalah ん. Karakter berikutnya (karakter kedua) merupakan hiragana baru.

   Contoh:

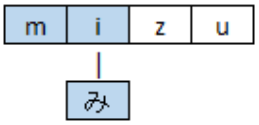
   (ii) Jika karakter kedua merupakan huruf vokal, maka *pattern* romaji memiliki panjang 2 karakter. Karakter berikutnya (karakter ketiga) merupakan hiragana baru.

   Contoh:

3. Mengecek karakter kedua jika karakter pertama tidak sama dengan ‘n’.
   (i) Jika Jika karakter kedua merupakan huruf vokal, maka *pattern* romaji memiliki panjang 2 karakter. Karakter berikutnya (karakter ketiga) merupakan hiragana baru.

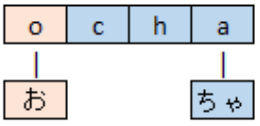
   Contoh:

   (ii) Jika karakter kedua merupakan konsonan, lanjut ke langkah keempat.

4. Mengecek kesamaan konsonan.
   (i) Jika konsonan pada karakter kedua berbeda dengan konsonan pada karakter pertama, cek karakter ketiga. Panjang *pattern* adalah 3 karakter. Karakter berikutnya (karakter ke-empat) merupakan hiragana baru.

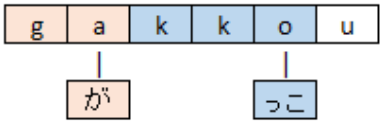
   Contoh:

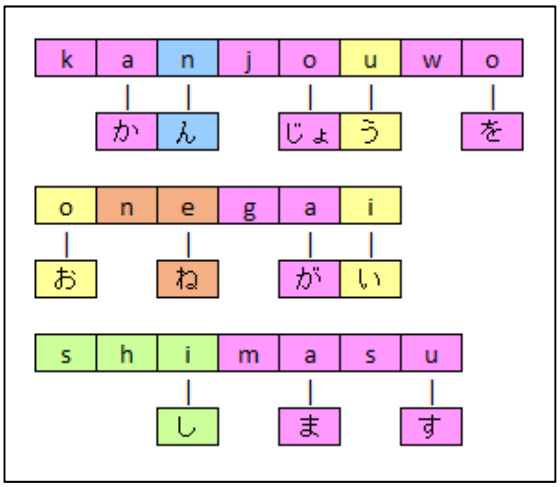
   (ii) Jika konsonan pada karakter kedua sama dengan konsonan pada karakter pertama, maka panjang *pattern* adalah 3 karakter dengan karakter hiragananya adalah tanda *sokuon* (っ) dan karakter konsonan-vokal yang sesuai. Karakter berikutnya (karakter keempat)

merupakan hiragana baru.

Contoh:



Di bawah ini terdapat contoh konversi romaji ke hiragana dengan algoritma yang telah diuraikan di atas.

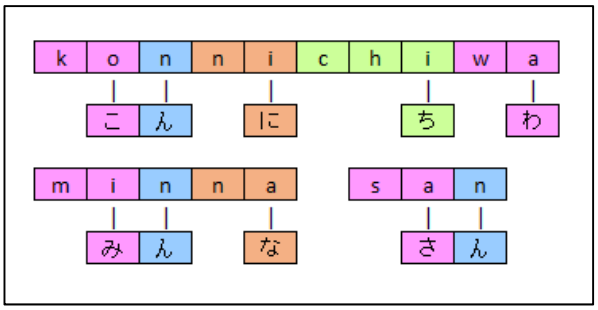


Gambar 5. Contoh 1 konversi romaji ke hiragana

Pada contoh di atas, setiap hiragana dengan kondisi yang berbeda memiliki warna yang berbeda:

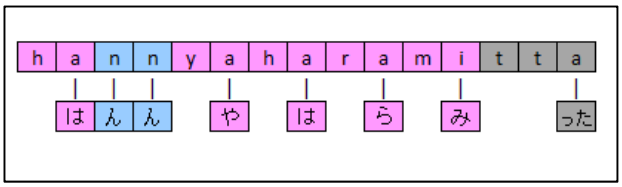
- Warna kuning artinya panjang romaji satu karakter dan merupakan huruf vokal.
- Warna biru artinya panjang romaji satu karakter dan merupakan karakter ‘n’.
- Warna ungu berarti bahwa panjang romaji dua karakter dengan karakter pertama konsonan bukan ‘n’ dan karakter kedua vokal.
- Warna jingga berarti bahwa panjang romaji dua karakter dengan karakter pertama konsonan ‘n’ dan karakter kedua vokal.
- Warna abu-abu artinya panjang romaji tiga karakter dengan karakter pertama dan kedua merupakan konsonan yang sama.
- Warna hijau artinya panjang romaji tiga karakter dengan karakter pertama dan kedua konsonan yang berbeda.

Pada contoh pertama di Gambar 5, teks romaji yang akan di konversi adalah *“kanjouwo onegai shimasu”*. Hasil dari konversi romaji ke hiragananya adalah かんじょうを おねがい します. Hasil konversi tersebut telah sesuai dengan hasil konversi yang semestinya.



Gambar 6. Contoh 2 konversi romaji ke hiragana

Pada contoh ke-2 yang dapat dilihat pada Gambar 6, teks romaji yang akan dikonversi adalah “*konnichiwa minna san*”. Hasil konversi yang didapatkan dari algoritma di atas adalah こんにちわ みんな さん. Hasil konversi tersebut tidak sesuai dengan hasil yang seharusnya, yaitu こんにちは みんな さん[4]. Terdapat perbedaan pada hasil konversi pada suku kata ‘wa’ di kata ‘konnichiwa’. Suku kata ‘wa’ pada kalimat ini merupakan partikel pada aturan Bahasa Jepang sehingga penulisannya menggunakan は dan bukan わ. Algoritma yang penulis rumuskan tidak memperhatikan aspek tata bahasa, sehingga untuk kalimat berpartikel, hasil konversi kurang tepat.



Gambar 7. Contoh 3 konversi romaji ke hiragana

Pada contoh ke-3 yang dapat dilihat pada Gambar 7, teks romaji yang akan dikonversi adalah “*hannyaharamitta*”. Hasil konversi yang didapatkan dari algoritma di atas adalah はんんやはらみった. Hasil konversi tersebut tidak sesuai dengan hasil yang seharusnya, yaitu はんにゃはらみった[5]. Terdapat perbedaan pada hasil konversi pada suku kata ‘nya’. Pada beberapa kata, suku kata ‘nya’ ditulis dengan んや, contohnya pada kata hinyari (ひんやり)[6]. Pada beberapa kata lainnya, suku kata ‘nya’ ditulis dengan にゃ seperti pada kata di contoh ke-3 di atas. Algoritma yang dirumuskan oleh penulis tidak memperhatikan aspek penulisan asli dari kata yang akan dikonversi sehingga hasil konversi kurang tepat untuk kasus-kasus tertentu.

V. KESIMPULAN

Konversi romaji ke hiragana dapat dilakukan dengan menggunakan algoritma *brute force* untuk pencocokan string. Algoritma *brute force* dinilai sesuai untuk memecahkan masalah ini jika dibandingkan dengan algoritma KMP dan Boyer-Moore karena *pattern* tidak

memiliki awalan (*prefix*) dan akhiran (*suffix*) yang sama dan pencocokan lebih sesuai jika dilakukan dari depan *pattern*. Pencocokan dilakukan dengan mengecek masing-masing karakter dan mengklasifikasikannya ke kondisi yang sesuai untuk mengetahui panjang *pattern* romaji untuk satu hiragana. Namun, algoritma yang dirumuskan oleh penulis belum optimal karena hasil konversi yang dihasilkan masih kurang tepat untuk beberapa kasus tertentu. Hal ini disebabkan karena penulis tidak mempertimbangkan aspek tata bahasa maupun aspek penulisan asli dari setiap kata.

VI. Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Tuhan yang Maha Esa, karena atas karunia-Nya makalah ini dapat diselesaikan. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi selaku dosen dari mata kuliah IF2211 Strategi Algoritma yang telah membantu dan memudahkan dalam pembuatan makalah ini.

Referensi

[1] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika STEI ITB.
[2] https://faicchi.wordpress.com/2014/09/18/kanji-the-beautiful-hiragana-the-curvaceous-and-katakana-the-evil/ diakses pada 3 Mei 2015, pukul 22:59 WIB.
[3] http://en.wikipedia.org/wiki/Hiragana diakses pada 4 Mei 2015, pukul 15:58 WIB.
[4] http://en.wiktionary.org/wiki/konnichi_wa diakses pada 4 Mei 2015, pukul 17.35 WIB.
[5] http://www.romajidesu.com/dictionary/meaning-of-hannyaharamitta.html diakses pada 4 Mei, pukul 18.05 WIB.
[6] http://www.romajidesu.com/dictionary/meaning-of-hinyari.html diakses pada 4 Mei, pukul 18.07 WIB.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 4 Mei 2015

Venny Larasati Ayudiani
NIM. 13513025