Aplikasi Algoritma Pencocokan String dan Algoritma Runut Balik dalam Konversi Romaji ke Hangul

Denita Hanna Widiastuti - 13514008
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514008@std.stei.itb.ac.id

***Abstract*—Hangul merupakan karakter Bahasa Korea dan alfabet latin merupakan alfabet yang biasa digunakan sehari-hari pada Bahasa Indonesia. Sedangkan romaji merupakan romanisasi dari karakter selain alfabet latin ke alfabet latin. Romanisasi karakter Hangul banyak digunakan dalam pembelajaran karatker Hangul dan Bahasa Korea untuk membantu pelajar membaca dan melafalkan karakter Hangul. Penulis akan merumuskan algoritma untuk mengonversi romaji menjadi karakter Hangul dengan algoritma pencocokan string sederhana, yaitu algoritma brute force dan algoritma runut balik.**

Keywords—brute force, runut balik, string, hangul, romaji, jamo awal, jamo tengah, jamo akhir, pencocokan string, blok, konsonan, vokal.

I. PENDAHULUAN

Globalisasi memaksa negara dan manusia untuk mampu berkembang agar tidak kalah dalam persaingan global. Keberhasilan suatu negara untuk berkembang di era globalisasi ini salah satunya ditandai dengan meningkatnya popularitas budaya negara tersebut ke berbagai belahan dunia. Salah satu negara yang berhasil mengalahkan globalisasi adalah negara Korea Selatan. Dengan budaya seni modern, bernama Hallyu, yang dimilikinya, Korea Selatan mampu membuat banyak orang tertarik untuk mempelajari budaya asli negara ginseng tersebut, terutama karakter asli Korea yang disebut Hangul.

Hangul mempunyai keunikan tersendiri yang akan terlihat asing untuk orang yang belum pernah mempelajarinya. Hangul mempunyai keunikan karena satu blok kata bisa terdiri dari 2 hingga 4 karakter dan cara penulisannya yang ditulis bertingkat sesuai jenis jamonya. Berbeda dengan karakter hiragana atau katakana yang berasal dari Jepang, yang setiap satu blok kata sudah pasti adalah satu suku kata.

Untuk memudahkan membaca Hangul, dibuatlah konversi huruf Hangul ke alfabet latin atau yang disebut dengan romaji, maupun konversi dari romaji ke karakter Hangul. Romaji membantu bagi para pemula untuk membaca dan mempelajari Hangul. Oleh karena itu, penulis membuat algoritma untuk memkonversi romaji ke Hangul dengan menggunakan kolaborasi dari algoritma pencocokan string brute force dan algoritma runut balik.

II. TEORI DASAR

2.1 Algoritma Pencocokkan String

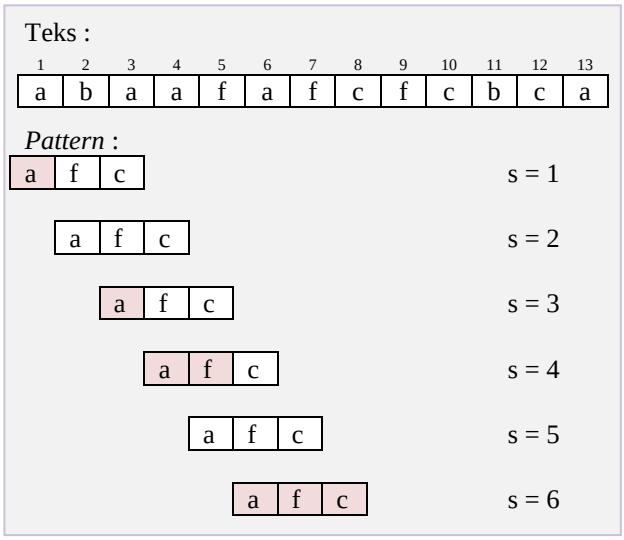
Pencocokan string (pattern matching *atau* string matching) merupakan suatu persoalan pencarian *string* di dalam teks. Pada persoalan pencarian *string* diberikan :

- Teks, merupakan *long string* yang memiliki panjang *n* karakter.
- *Pattern*, yaitu *string* dengan panjang *m* karakter ($m < n$) yang akan dicari keberadaanya dalam teks.

Pencocokkan akan mencari lokasi pertama di dalam teks yang bersesuaian dengan *pattern.* Ada beberapa metode dalam algoritma pencocokkan *string*, diantaranya algoritma *brute force*, algoritma KMP, dan algoritma *Boyer Moore.*

2.1.1 Algoritma Brute Force

Algoritma *brute force* merupakan algoritma yang memiliki pendekatan penyelesaian yang paling sederhana dibandingkan algoritma lainnya. Algoritma *brute force* mencocokkan karakter pertama *pattern* (indeks *j*) dengan karakter pertama pada teks (indeks *i*). Jika sama, pencocokkan akan dilakukan pada karakter selanjutnya, dan seterusnya. Jika karakter pada *pattern* tidak cocok dengan karakter pada teks, maka akan dilakukan pergeseran pada indeks i_{awal} sebanyak 1 karakter ke kanan. Sedangkan indeks *j*, akan dikembalikan ke karakter pertama *pattern*.



Gambar 1. Contoh penggunaan algoritma *brute force.*

Kesimpulan dari gambar 1 adalah *pattern* `afc` ditemukan pada posisi indeks ke 6 dari awal teks. Kasus terbaik terjadi jika karakter pertama *pattern* tidak pernah sama dengan karakter teks yang dicocokkan, kecuali pada pencocokkan yang terakhir. Kasus terbaik memiliki kompleksitas *O(n)*. Contoh kasus terbaik :

Teks : `Algoritma Pencarian Yang Unik`
Pattern : `Unik`

Sedangkan kasus terburuk terjadi jika karakter pertama hingga indeks *m*-1 sama dengan karakter indeks *n*, namun berbeda pada indeks *m*, dan pencocokkan dilakukan hingga mencapai karakter terakhir dari teks. Kasus terburuk membutuhkan *m(n-m+1)* perrbandingan yang kompleksitasnya adalah *O(mn)*. Contoh kasus terburuk :

Teks : `aaaaaaaaaaaaaaaaaaaaaaaaah`
Pattern : `aaaaah`

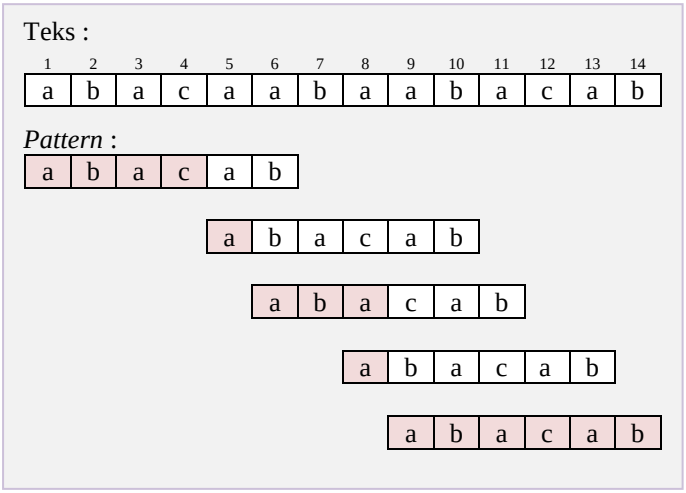
2.1.2 Algoritma Knuth-Morris-Pratt (KMP)

Algoritma KMP merupakan algoritma pencocokkan *string* dengan prinsip *left-to-right* yang mirip dengan algoritma *brute force*. Perbedaannya adalah jumlah indeks pada teks yang digeser jika ditemukan ketidakcocokkan antara karakter di teks dengan karakter di *pattern*. Jumlah indeks yang digeser pada algoritma KMP memiliki perhitungan tersendiri dan bersifat signifikan, sehingga dapat meningkatkan kemangkusan dari proses pencocokkan *string*.

Pada algoritma KMP terdapat 2 jenis *upastring* yang dikenal dengan istilah *suffix* dan *prefix*. Jika suatu *string* S mempunyai indeks 1..*n*, maka :

- *Prefix* dari S adalah *upastring* S[1 .. *k*-1]
- *Suffix* dari S adalah *upastring* S[*k*-1 .. *n*]

Jumlah indeks yang digeser pada algoritma KMP adalah jumlah maksimal *prefix* dari S[1.. *n*-1] yang merupakan *suffix* dari S[1 .. *n*-1].



Gambar 2. Contoh penggunaan algoritma KMP

Pada gambar 2, terlihat pencocokkan kedua dilakukan dengan menggeser indeks ke kanan sebanyak 3 (melewati karakter `bac`). Tiga karakter dalam indeks teks tersebut tidak perlu dicek karena sudah dapat dipastikan berbeda dengan karakter pada *pattern*. Kepastian ini didapat setelah dihitung fungsi pinggiran (*border function*).

Tabel 1. Tabel Fungsi Pinggiran

j	1	2	3	4	5	6
P[*j*]	a	b	a	b	a	a
b(*j*)	0	0	1	2	3	1

j : indeks
P[j] : karakter pada indeks ke-*j*
b(j) : fungsi pinggiran

Untuk menghitung fungsi pinggiran dibutuhkan waktu *O(m)*, sedangkan pencarian *string* membutuhkan waktu *O(n)*. Sehingga, kompleksitas waktu algoritma KMP adalah *O(m+n)*.

2.1.3 Algoritma Boyer Moyer

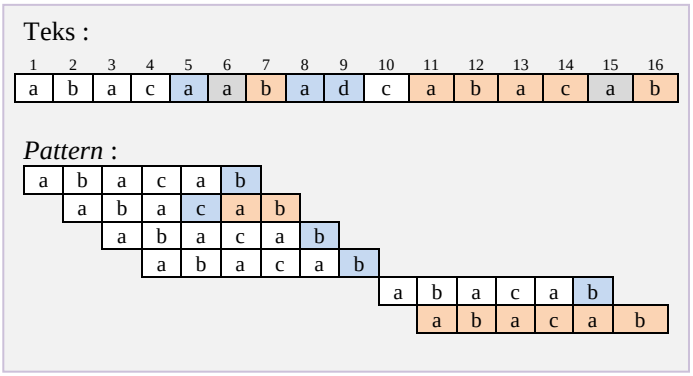
Algoritma *Boyer Moore* memiliki 2 teknis yang berbeda dengan algoritma lain,

1. Pencocokkan dimulai dari belakang (kanan ke kiri)
2. Teknik *character-jump*. Jika ketidakcocokkan muncul pada teks T[*i*] = x, maka karakter pada *pattern* P[*j*] tidak sama dengan karakter pada teks T[*i*].

Untuk menentukan letak *character-jump* apabila ditemukan ketidakcocokkan, perhatikan 3 kemungkinan

kasus yang mungkin terjadi,

1. Jika *pattern* P memiliki karakter x di indeks sebelah kiri dari *j*, maka geser P ke kanan sedemikian sehingga indeks P sejajar dengan indeks *i* dimana x terdekat berada.
2. Jika *pattern* P memiliki karakter x bukan di indeks sebelah kiri dari *j*, maka geser P ke kanan sebanyak 1 karakter.
3. Jika yang terjadi tidak termasuk kasus 1 maupun kasus 2, geser *pattern* sehingga P[1] sejajar dengan T[*i*+1].



Gambar 3. Contoh penggunaan algoritma *Boyer Moore*

Kasus terbaik terjadi jika karakter indeks *j* pada teks dan indeks *i* pada *pattern* tidak pernah sama dengan karakter teks yang dicocokkan, kecuali pada pencocokkan yang terakhir. Kasus terbaik memiliki kompleksitas *O(n)*. Contoh kasus terbaik :

Teks : `Unik Sangat Meja Itu`
Pattern : `Unik`

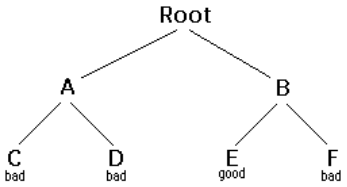
Sedangkan kasus terburuk algoritma *Boyer* Moore adalah saat karakter yang dilakukan pencocokkan selalu sama kecuali karakter terakhir. Sehingga pencarian dilakukan pada seluruh karakter pada teks. Contoh kasus terburuk :

Teks : `haaaaaaaaaaaaaaaaaaaaaaaaa`
Pattern : `haaaaa`

2.2 Algoritma Runut Balik

Algoritma runut balik, atau *backtracking*, adalah algoritma penyelesaian masalah yang mencari solusi secara sistematis berdasarkan ruang solusi, namun tudak semua ruang solusi diperiksa, hanya yang mengarah pada solusi yang akan diperiksa.

Algoritma runut balik merupakan algoritma lanjutan yang berbasis pada algoritma pencarian melebar atau DFD (*Depth First Search)*. Algoritma ini secara secara garis besar akan ‘mundur’ jika pada suatu pencarian tidak ditemukan solusi dan tidak ada kemungkinan solusi lain yang selevel, sehingga pencarian harus ‘mundur’ untuk mencari solusi pada level sebelumnya.



Gambar 4. Ilustrasi Pohon Pencarian Solusi

Pencarian solusi *good* untuk pohon pada gambar 4 dengan menggunakan algoritma runut balik terdiri dari tahap :

1. Dimulai dari Root, pilih A
2. Di A, pilih C.
3. C berisi *bad*, mundur ke A.
4. Di A, pilih C.
5. C berisi *bad*. Mundur ke A.
6. Di A, semua pilihan sudah ditelusuri dan tidak ditemukan solusi. Mundur ke Root.
7. Di Root, pilih B.
8. Di B, pilih E.
9. E berisi *good*. Solusi ditemukan

III. SISTEM PENULISAN HANGUL

Hangul adalah alphabet yang digunakan untuk menulis Bahasa Korea. Hangul atau Hangeul diciptakan oleh Raja Sejong (1397-1450) pada tahun 1443 di masa Dinasto Jeoseon. Hangul kini digunakan sebagai alfabet resmi di Korea Selatan. Hangul juga dapat digunakan untuk menulis bahasa selain Bahasa Korea.

3.1 Huruf Vokal dan Konsonan

Sama seperti alphabet romaji, hangul dibagi menjadi huruf vokal dan huruf konsonan. Fungsi huruf vokal dan konsonan pada hangul sama dengan fungsi huruf vokal dan konsonan pada alfabet latin.

Vocal	Romanisasi
ㅏ	a
ㅑ	ya
ㅓ	eo
ㅕ	yeo
ㅗ	o
ㅛ	yo
ㅜ	u
ㅠ	yu
ㅡ	eu
ㅣ	i

Vocal Ganda	Romanisasi
ㅐ	ae
ㅒ	yae
ㅔ	e
ㅖ	ye
ㅘ	wa
ㅙ	wae
ㅚ	oe
ㅝ	we
ㅞ	weo
ㅟ	ui
ㅢ	eui, ee

Gambar 5. Huruf Vokal Tunggal dan Ganda

Masing –masing huruf vokal dan konsonan pada hangul dibagi menjadi dua jenis, tunggal dan ganda.

Konsonan Ganda	Nama	Romanisasi
ㄲ	ssang kiyeok	kk
ㄸ	ssang digeut	dd
ㅃ	ssang bieup	pp
ㅆ	ssang siot	ss
ㅉ	ssang jieut	jj

Konsonan	Nama	Romanisasi
ㄱ	giyeok	g,k
ㄴ	nieun	n
ㄷ	digeut	d
ㄹ	rieul	r,l
ㅁ	mieum	m
ㅂ	bieup	b,p
ㅅ	siot	s
ㅇ	ieung	ng
ㅈ	jieut	j
ㅊ	chiet	ch
ㅋ	kieuk	k^
ㅌ	tieut	t
ㅍ	pieup	p
ㅎ	hieut	h

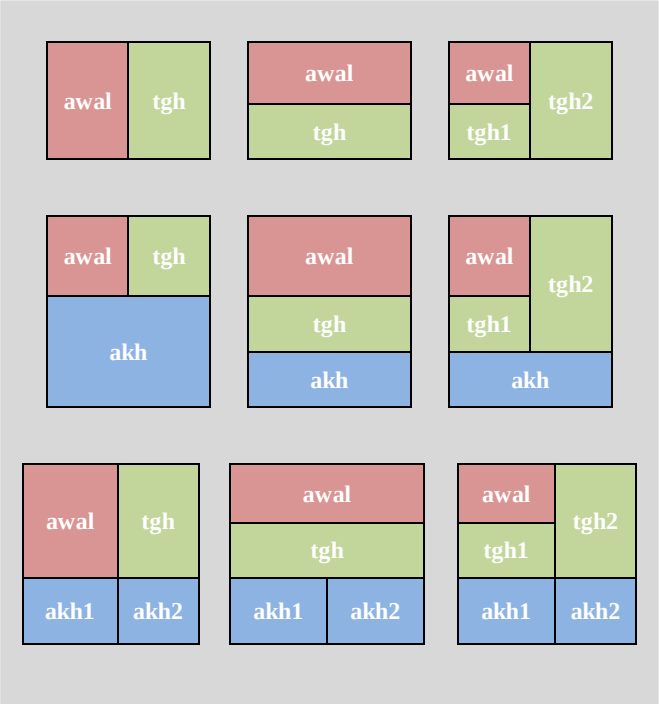
Gambar 6. Huruf Konsonan Tunggal dan Ganda

3.2 Jamo

Jamo adalah blok *unicode* yang mengandung bentuk posisi dari susunan huruf konsonan dan vokal pada Hangul. Jamo hangul dibagi menjadi 3 jenis :

1. Jamo awal.
   Dapat berupa huruf konsonan maupun huruf vokal, baik jenis ganda maupun tunggal.
2. Jamo tengah
   Dapat berupa huruf konsonan maupun huruf vokal, baik jenis ganda maupun tunggal.
3. Jamo akhir.
   Merupakan huruf konsonan, baik jenis ganda maupun tunggal.

Dalam penulisan suatu blok dengan hangul, posisi huruf harus sesuai dengan aturan jamo yang berlaku.



Gambar 7. Aturan Peletakan Huruf Hangul Sesuai Joma

Contoh penulisan hangul beserta romajinya antara lain:

아프다 (Apeuda) = sakit

죽다 (Jukta) = mati

꿈 (Kkum) = dream

자다 (Ja da) = to sleep

찌다 (Jji da) = mengukus

졸음 (Jor eum) = kantuk

젊다 (Jeol da) = muda

IV. ANALISIS

Untuk mengkonversi romaji menjadi hangul, dapat dilakukan dengan menerapkan beberapa metode pencocokkan *string*. Diantara tiga jenis algoritma yang sudah penulis paparkan sebelumnya, menurut penulis algoritma *brute force* merupakan algoritma yang paling sangkil dan mangkus untuk mengkonversi romaji menjadi hangul.

Hangul mempunyai aturan dalam bentuk joma dalam penulisannya. Menurut joma, setiap blok suku kata dapat terdiri dari beberapa kemungkinan posisi huruf. Sehingga penulis membagi kemungkinan tersebut menjadi 4 :

1. huruf vokal
2. huruf vokal dan huruf konsonan
3. huruf konsonan dan huruf vokal
4. huruf konsonan, huruf vokal, dan huruf konsonan.

Sebagian kemungkinan di atas menyebabkan proses konversi juga memerlukan algoritma runut balik.

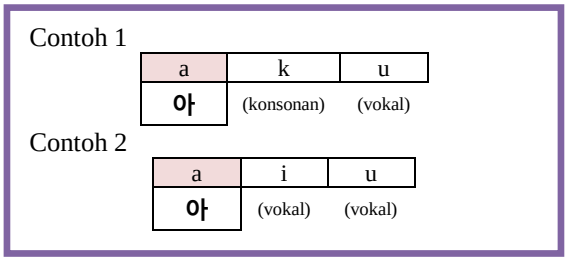
Algoritma runut balik diperlukan untuk mengecek, apakah blok suku kata tersebut mempunyai joma akhir atau tidak. Mekanisme lebih lanjut akan diterangkan pada penjelasan di subbab dibawah.

Proses konversi di bawah tidak memperhatikan apakah jenis huruf konsonan dan vokal termasuk jenis tunggal atau ganda. Sehingga bila terdapat jenis ganda, maka akan dianggap sebagai satu huruf konsonan atau satu huruf vokal.

4.1 Kondisi 1

Suatu blok suku kata dikatakan masuk kondisi 1 jika setelah ditemukan huruf vokal, ditemukan huruf :

- konsonan diikuti huruf vokal setelahnya
- vokal diikuti huruf vokal lagi setelahnya

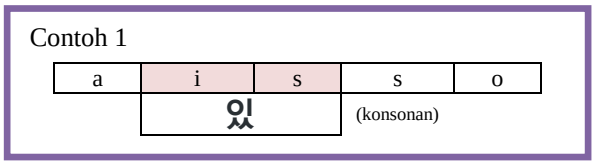


Gambar 7. Contoh Konversi Pada Kondisi 1

4.2 Kondisi 2

Suatu blok suku kata dikatakan masuk kondisi 2 jika setelah ditemukan huruf vokal dan konsonan, ditemukan huruf :

- konsonan

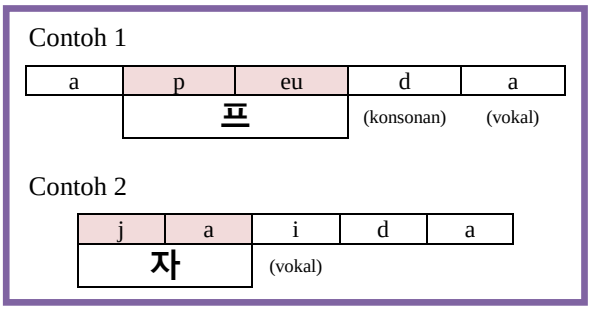


Gambar 8. Contoh Konversi Pada Kondisi 2

4.3 Kondisi 3

Suatu blok suku kata dikatakan masuk kondisi 3 jika setelah ditemukan huruf konsonan dan vokal, ditemukan huruf :

- konsonan diikuti huruf vokal setelahnya
- vokal

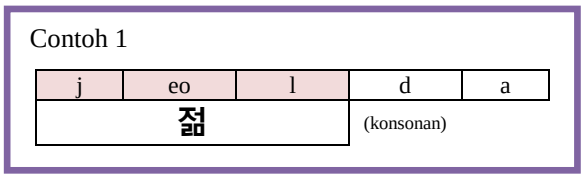


Gambar 8. Contoh Konversi Pada Kondisi 3

4.4 Kondisi 4

Suatu blok suku kata dikatakan masuk kondisi 4 jika setelah ditemukan huruf konsonan, vokal, dan konsonan lagi, ditemukan huruf :

- konsonan



Gambar 9. Contoh Konversi Pada Kondisi 4

V. KESIMPULAN

Konversi romaji ke hangul dapat dilakukan dengan menggunakan algoritma pencocokkan *string* sederhana, yaitu *brute force*. Algoritma ini lebih baik digunakan dibandingkan algoritma pencocokkan *string* lain, seperti KMP dan *Boyer Moore*, karena dalam proses konversi ini, dibutuhkan pembacaan karakter satu per satu, sehingga algoritma *brute force* lah yang paling tepat untuk digunakan.

Selain penggunaan algoritma pencocokkan *string*, strategi runut balik juga diperlukan dikarenakan aturan penulisan dalam hangul yang terdiri dari beberapa variasi kondisi berdasarkan joma. Hal ini mau tidak mau, membuat sistem harus melihat jenis huruf di depan suatu blok suku kata untuk menentukan variasi kondisi yang tepat.

Dalam makalah ini, masih terdapat beberapa kekurangan dikarenakan tidak menjelaskan secara rinci mengenai cara penulisan hangul. Makalah ini hanya berfokus mengenai cara untuk membaca romaji ke hangul. Sehingga, proses penulisan blok suku kata dalam hangul tidak dijelaskan.

VI. APPENDIX

Pertama, saya mengucapkan terima kasih kepada Allah SWT karena berkat rahmat dan izin-Nya lah makalah ini dapat selesai tepat waktu. Tak lupa shalawat serta salam saya haturkan ke junjungan Nabi Muhammad SAW, karna atas usahanya lah, umat manusia bisa terlepas dari zaman kebodohan menuju zaman yang penuh ilmu pengetahuan.

Saya juga mengucapkan terima kasih kepada orangtua di rumah yang tidak henti-hentinya mendoakan dan mendidik saya sehingga saya dapat menempuh ilmu di Institut Teknologi Bandung. Tak lupa ucapan terima kasih untuk Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi selaku dosen mata kuliah IF2211 Strategi Algoritma atas ilmu yang telah diberikan selama perkuliahan, sehingga saya mampu menyelesaikan makalah ini. Terima kasih juga saya ucapkan kepada teman-teman yang telah membantu dalam penulisan makalah ini.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika STEI ITB.

[2] https://id.wikipedia.org/wiki/Hangul_Jamo_(blok_Unicode). Diakses pada 7 Mei 2016.

[3] Putra, Desi Sasadara. *Otodidak Belajar Korea*. Yogyakarta: Pustaka Widyatama. 2011.

[4] https://www.cis.upenn.edu/~matuszek/cit594-2012/Pages/backtracking.html. Diakses 8 Mei 2016.

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, 8 Mei 2016

Denita Hanna Widiastuti
13514008