Aplikasi Pencocokan Pola pada Asisten Pribadi Cerdas Google Now

Nursyahrina - 13513060
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13513060@std.stei.itb.ac.id

***Abstract*—Pencocokan pola adalah salah satu metode pemecahan masalah yang dibahas pada mata kuliah strategi algoritma. Pencocokan pola adalah metode untuk mencari pola pada serangkaian teks. Asisten pribadi cerdas (*Intelligent Personal Assistant*) contohnya Google Now, menerapkan penggunaan metode pencocokan pola seperti algoritma Boyer-Moore dan Knuth-Morris-Pratt, untuk mengenali perintah dari pengguna aplikasi berdasarkan pola perintah (commands) yang terdefinisi. Aplikasi Google Now selanjutnya akan membuka aplikasi lain atau menampilkan informasi yang dibutuhkan pengguna.**

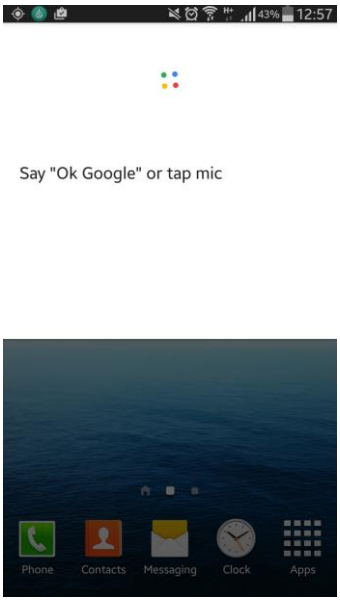
***Keywords*—algoritma; Boyer-Moore; google now; Knuth-Morris-Pratt; KMP; pola; *pattern*; teks;**

I. PENDAHULUAN

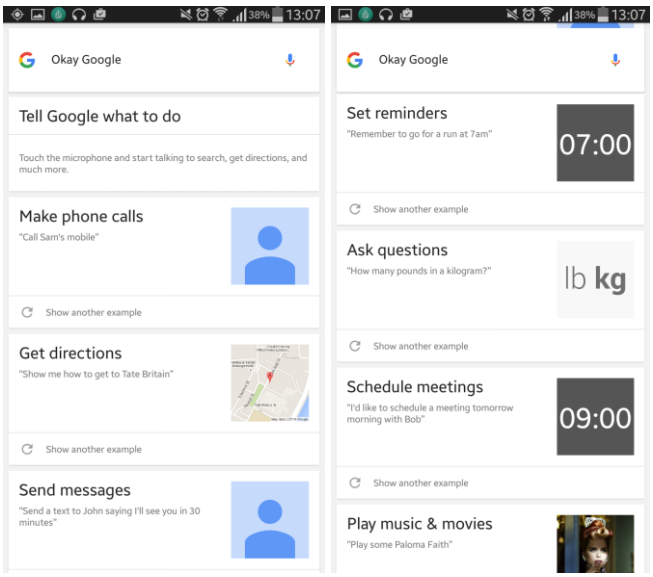
Perkembangan teknologi pada saat ini memungkinkan pengguna gadget, baik handphone, tablet, notebook dan sejenisnya untuk beraktivitas dan melakukan banyak hal dengan teknologi yang ada. Untuk semakin mempermudah pengguna menggunakan gadget , pengembang produk teknologi seperti Google menyediakan *Intelligent Personal Assistant* atau asisten pribadi cerdas yang dapat melakukan banyak hal pada gadget sesuai perintah yang diberikan pengguna melalui voice commands. Produk Google ini bernama Google Now.

Google Now adalah asisten pribadi cerdas (*Intelligent Personal Assistant*) yang dikembangkan oleh Google. Google now dapat digunakan pada Google Search di Android atau iOS maupun pada Google Chrome di PC. Google Now adalah bentuk evolusi dari software Google Search. Idenya sederhana, yaitu Google Now memprediksikan apa saja informasi yang dibutuhkan oleh orang kebanyakan dan mencoba menyuguhkannya dalam format yang mudah dipahami dengan eksekusi yang cerdas.

Diumumkan pada Google I/O ,Juni 2012, Google now kemudian disajikan dalam bentuk “desain kartu (card design)” yang banyak digunakan aplikasi Android pada zaman sekarang. Google Now memahami perintah melalui ucapan, serupa dengan asisten pribadi cerdas lainnya seperti Siri pada perangkat Apple dan S Voice pada perangkat Samsung. Diawali dari Android 4.4 KitKat, Google Now sekarang sudah menjadi bagian dari *home screen launcher* dan dapat dijalankan langsung dengan mengucapkan ‘OK Google’ pada *home screen*[1].



Gambar 1 Tampilan Awal Google Now Di Perangkat Android



Gambar 2 Beberapa contoh perintah yang dikenali Google Now. Dapat dilihat dengan mengucapkan 'Okay Google'.

Gambar 3 Contoh penggunaan Google Now untuk mengetahui prediksi cuaca (hujan)

Selain Bahasa Inggris, Google Now juga memahami Bahasa lain, seperti Bahasa Indonesia. Jika membreikan perintah melalui ucapan dengan Bahasa Indonesia, maka respon dari Google Now juga akan diberikan dengan bahasa yang sama. Berikut adalah beberapa contoh perintah dalam Bahasa Inggris yang dikenali Google Now[2] :

- Open [app name]. Contoh: "Open Gmail."
- Go to [website]. Contoh.: "Go to CNET.com."
- Call [contact name]. Contoh.: "Call Mom."
- Text or Send text to [contact name]. Contoh: "Text Wife I'm running late."
- Set an alarm for [specific time, or amount of time]. Contoh: "Set alarm for 10 a.m." Or "Set alarm for 20 minutes from now."
- Set a timer for [X] minutes.
- Show me my calendar.
- Pencarian sederhana. Contoh: "Mountain bikes."
- When is [special event]. Contoh: "When is the next eclipse?" or "When is Easter in 2018?"
- What sound does [animal] make?
- Directions or Navigate to [address, name, business name, type of business, or other destination].
- Show me the trailer for [movie title].

II. DASAR TEORI

A. Pencocokan Pola (*Pattern Matching*)

Definisi :

1. T: Teks, yaitu string yang panjangnya n karakter.
2. P:*Pattern*, yaitu string dengan panjang m karakter (asumsi m <<< n, m sangat kecil dibanding n) yang akan dicari pada teks.

Pada persoalan pencocokan pola akan ditentukan apakah ada string yang bersesuaian dengan *pattern* di dalam teks. Jika ada maka mengembalikan nilai posisi dari kemunculan *pattern* pada teks.

Aplikasi Algoritma Pencocokan Pola

Pencocokan pola banyak digunakan dalam berbagai program atau aplikasi yang digunakan sehari-hari, seperti fitur pencarian pada editor teks, web search engine seperti google, analisis citra, *bioinformatics* seperti pencocokan rantai asam amino rantai DNA, dll.

Algoritma pencocokan pola yang dibahas pada makalah ini adalah algoritma Knuth-Morris-Pratt dan Boyer-Moore. Berikut adalah penjelasan detail mengenai kedua algoritma tersebut.

1) **Algoritma Knuth-Morris-Pratt**

Pencarian *pattern* di dalam teks dengan algoritma Knuth-Morris-Pratt (KMP) dilakukan dengan pengecekan karakter teks dari kiri ke kanan, seperti pada algoritma brute force. Namun, pergeseran dilakukan dengan cara yang lebih cerdas. Untuk menghindari pencocokan yang tidak penting (karena sudah dapat dipastikan ketidakcocokannya), metode pergeseran yang dilakukan memanfaatkan informasi suffix dan prefix dari *pattern*. Pada pencocokan, *pattern* digeser sebanyak panjang suffix *pattern* yang juga merupakan *prefix* dari *pattern*. Pergeseran ini memanfaatkan fungsi pinggiran.

Fungsi Pinggiran pada KMP

KMP memproses *pattern* untuk mencocokkan *prefix* dengan *suffix pattern* itu sendiri. Jika,

- *j* = posisi ketidakcocokan pada *pattern P*[],

- k = posisi sebelum ketidakcocokan j ($k = j - 1$),
- fungsi pinggiran atau *border function* $b(k)$ adalah ukuran *prefix* terbesar yang juga merupakan *suffix* dari *pattern* $P[1..k]$. *border function* juga sering disebut *failure function* (*fail*). Berikut ini adalah contoh dari *border function* pada *pattern* "ABABABC".

Tabel 1 Contoh Fungsi Pinggiran

j	1	2	3	4	5	6	7
P[j]	A	B	A	B	A	B	C
b[j]	0	0	1	2	3	4	0

Pemanfaatan fungsi pinggiran adalah sebagai berikut :

jika ada ketidakcocokan pada P[j] (P[j]!= T[j] maka

k = j – 1,

j = *b*(*k*) + 1 //nilai j yang baru

Kompleksitas Waktu Algoritma KMP

Kompleksitas waktu untuk menghitung fungsi pinggiran O(m), untuk melakukan pencocokan sepanjang karakter pada teks O(n). Oleh karena itu kompleksitas waktu algoritma KMP adalah O(m+n), lebih cepat bila dibandingkan dengan algoritma brute force.

Kelebihan dan Kelemahan Algoritma KMP

Kelebihan KMP adalah algoritmanya tidak perlu bergerak mundur pada teks masukan, hal ini menyebabkan algoritma ini baik untuk *input* teks dengan memproses *file* yang sangat besar atau membaca *file* dari perangkat eksternal atau melalui stream antar jaringan.

Kelemahannya, algoritma KMP kurang cocok untuk teks dengan jumlah alfabet yang banyak atau karakter alfabet penyusun yang sangat beragam. Hal ini menyebabkan besarnya kemungkinan ketidakcocokan. Ketidakcocokan akan terjadi diawal *pattern*, dan inilah yang menyebabkan pencocokan lebih lambat.

2) **Algoritma Boyer-Moore**

Pencocokan pola dengan algoritma Boyer-Moore menggunakan dua teknik sebagai berikut :

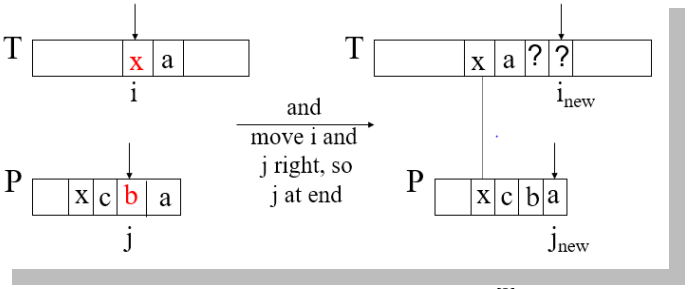
1. The looking-glass technique

   Mencari *pattern* P di dalam teks T dengan pencocokan menelusuri *pattern*, dari karakter terakhir *pattern* sampai karakter pertama (bergerak kebelakang).

2. The character-jump technique

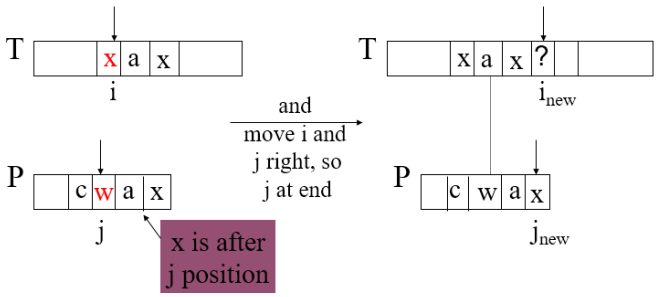
   Ketika terjadi ketidakcocokan antara karakter teks dan karakter pada *pattern*. Berdasarkan ada atau tidaknya karakter teks (yang sedang dicek) pada *pattern*, ada 3 jenis kasus pergeseran, yaitu :

1. Jika karakter teks x yang sedang dicek ada pada *pattern* P, maka P digeser ke kanan, hingga kemunculan terakhir x pada P.



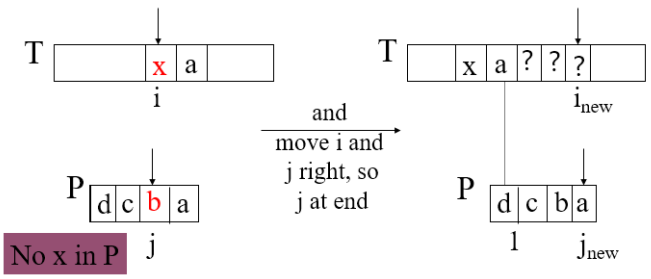
Gambar 4 Kasus ke-1 Pergeseran *Pattern*[3]

2. Jika x ada pada *pattern* P, namun tidak mungkin lagi disejajarkan dengan x pada *pattern* karena x berada pada sisi kanan *pattern*, maka P digeser hanya 1 karakter ke kanan.



Gambar 5 Kasus ke-2 Pergeseran *Pattern*[3]

3. Jika tidak cocok dengan kasus 1 maupun kasus 2, maka P digeser ke kanan sampai karakter pertama *pattern* sejajar dengan karakter setelah x pada teks.



Gambar 6 Kasus ke-3 Pergeseran *Pattern*[3]

Algoritma Boyer-Moore ini menggunakan fungsi *last occurance* (kemunculan terakhir) untuk melakukan pengecekan 3 kasus diatas dan untuk menggeser *pattern*.

Fungsi Kemunculan Terakhir pada Boyer-Moore

Algoritma Boyer-Moore memproses *pattern* P dan huruf-huruf alfabet untuk membentuk fungsi kemunculan terakhir (*last occurance*) L(). L() memetakan huruf-huruf alfabet dengan kemunculan terakhirnya pada *pattern*. Kemunculan terakhir suatu karakter x atau L(x) adalah kemunculan terakhir

atau indeks i terbesar dari posisi x di dalam *pattern*, P[i] == x. Jika karakter alfabet x tidak ada dalam *pattern*, maka fungsi memetakan x dengan nilai -1. Fungsi ini biasanya direpresentasikan sebagai *array* (table) seperti pada contoh.

Misal, A = {a, b, c, d, e} dan P = "membaca", maka fungsi kemuculan terakhir L() nya adalah sebagai berikut,

Tabel 2 Contoh Fungsi Last Ocuurance pada Algoritma Boyer-Moore

x	*a*	*b*	*c*	*d*	*e*
L(x)	7	4	6	-1	2

Kompleksitas Waktu Algoritma Boyer-Moore

Kompleksitas waktu untuk kasus terburuk dari algoritma Boyer-Moore adalah O(mn + A). dengan A adalah himpunan alfabet dari karakter yang ada pada *pattern*.

Kelebihan dan Kelemahan Algoritma Boyer-Moore

Kelebihan algoritma Boyer-Moore adalah algoritma ini sangat cepat untuk jenis alfabet dengan himpunan karakter yang besar seperti alfabet ASCII atau teks dalam bahasa. Kelemahannya algoritma ini tidak cocok untuk alfabet dengan himpunan kecil seperti karakter binary, karena pemrosesannya akan lebih lama. Algoritma Boyer-Moore untuk alfabet yang lebih besar jauh lebih cepat bila dbandingkan dengan algoritma brute force.

III. APLIKASI PENCOCOKAN POLA PADA GOOGLE NOW

Aplikasi Google Now menerima perintah dalam bentuk ucapan. Gelombang suara yang diterima kemudian akan ditransformasikan dalam bentuk teks dengan metode *voice recognition* dan *speech-to-text*. Teks yang dihasilkan inilah yang kemudian akan dicocokkan dengan kata kunci - kata kunci command yang telah didefinisikan oleh Google Now. Agar kemudian perintah tersebut dapat dieksekusi dan memberikan informasi yang tepat untuk pengguna.

A. Pencocokan Pola Pada Google Now

Algoritma pencocokan pola KMP dan Boyer-Moore dapat digunakan pada Google Now untuk mengenali perintah yang telah diubah menjadi bentuk teks. Google Now kemudian akan menjalankan aplikasi atau memberikan informasi sesuai perintah yang telah teridentifikasi tersebut.

Misalkan pengguna ingin mengetahui rute perjalanan dari tempat tinggal di Jalan Cisitu Baru ke Institut Teknologi Bandung. Pengguna dapat mengucapkan perintah "show me the directions to institut teknologi Bandung". Perintah suara kemudian akan dikonversikan menjadi bentuk text dan text akan diproses untuk mengetahui jenis perintah yang diberikan pengguna.

1) Algortima Knuth-Morris-Pratt

Algoritma KMP diimplementasikan dengan bahasa pemrograman dalam dua buah fungsi. Fungsi pertama `computeFail` adalah untuk menghitung fungsi pinggiran pada algoritma KMP dan fungsi kedua `kmpMatch` untuk mencocokkan teks dan *pattern*. Berikut adalah implementasi dari algortima KMP dalam bahasa pemrograman notasi algoritmik.

```
function computefail(input pattern : string) → array of integer
{ fungsi menghitung fungsi pembatas pada pattern dan mengembalikan array integer hasil fungsi pembatas }

KAMUS
        fail : array of integer dengan ukuran sama dengan panjang pattern
        m : integer
        j : integer
        i : integer

ALGORITMA
        fail[0] ← 0
        m ← panjang pattern
        j ← 0
        i ← 1

        while i < m do {
                if karakter pattern indeks ke j sama dengan karakter pattern indeks ke i then
                        { j+1 karakter cocok }
                        fail[i] ← j + 1
                        i ← i + 1
                        j ← j + 1
                else if j > 0 then
                        j ← fail[j-11]
                else { tidak cocok }
                        fail[i] ← 0
                        i ← i + 1
        → fail
```

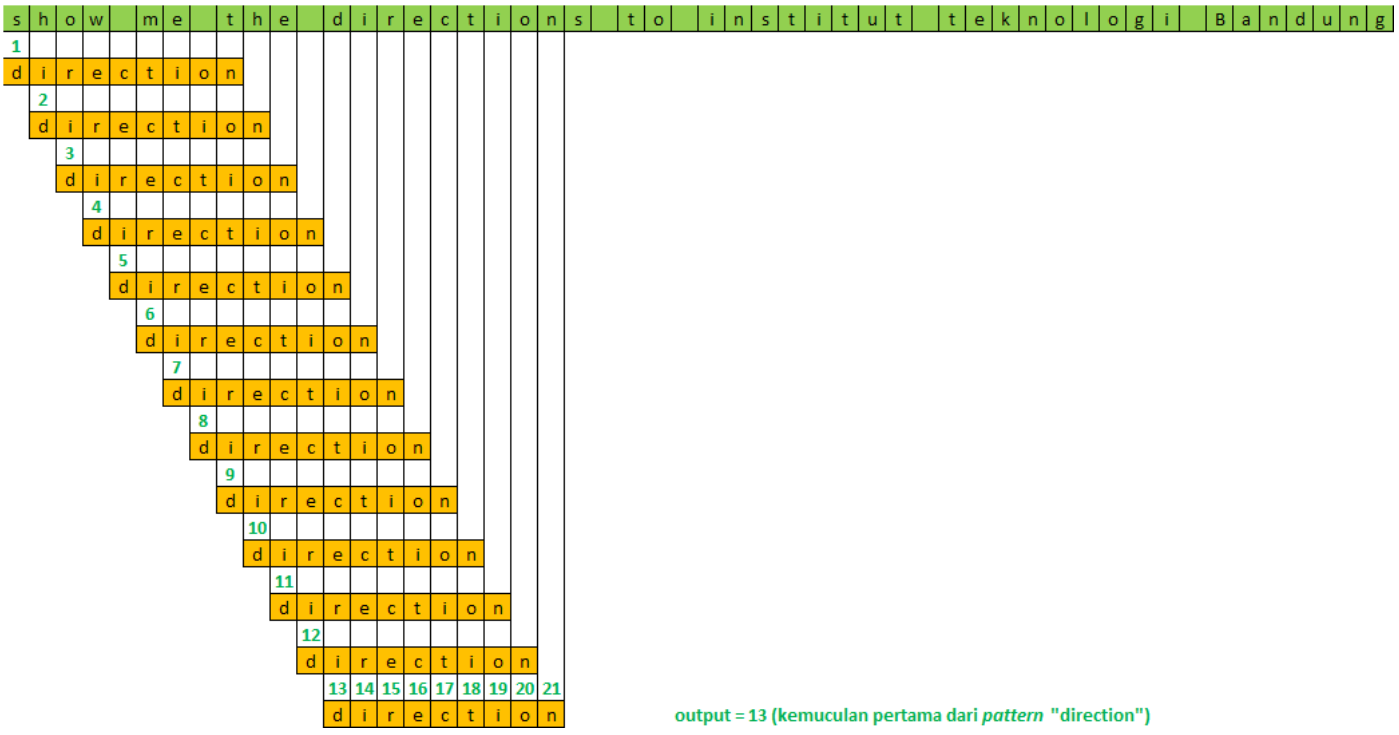
```
function kmpMatch(input text : string,
input pattern : string) → integer
{ fungsi melakukan pencocokan string atau
pola dengan algoritma KMP, mengembalikan
nilai posisi kemunculan pattern di dalam
teks, atau -1 jika pattern tidak ditemukan
}

KAMUS

        n : integer
        m : integer
        fail : array of integer
        i : integer
        j : integer

ALGORITMA

        array fail ← computeFail(pattern)
        n ← panjang teks
        m ← panjang pattern
        i ← 0
        j ← 0

        { Untuk setiap karakter teks }
        { dilakukan pencocokan. }
        while i < n do
                if karakter pada teks indeks
                ke i sama dengan karakter
                pattern indeks ke j then
                        if j = m - 1 then
                                → i - m + 1;
                                {cocok}
                        i ← i + 1
                        j ← j + 1
                else if (j > 0) then
                        j ← fail[j-1]
                else
                        i ← i + 1
        → -1 {tidak ada kecocokan}
```

Untuk pengujian notasi algoritmik ditranlasikan terlebih dahulu ke bahasa pemrograman seperti Java, C, C++, C#, atau bahasa pemrograman lainnya.

Misalkan *pattern* P = "direction" merupakan kata kunci perintah untuk menampilkan peta atau navigasi perjalanan. Teks yang didapatkan dari perintah ucapan adalah T = "show me the directions to institute teknologi Bandung". Pencocokan pola dengan algoritma KMP adalah sebagai berikut :

Fungsi Pembatas untuk *pattern* "direction"

j	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
P[j]	d	i	r	e	c	t	i	o	n
b[j]	0	0	0	0	0	0	0	0	0



Gambar 7 Perhitungan pencocokan pola dengan KMP

Dengan menggunakan algortima KMP, pencocokan pola mengembalikan nilai 13 merupakan indeks posisi kemunculan pertama dari *pattern* "direction". **Jumlah pencocokan adalah 21 kali**, sampai ditemukan kecocokan.

2) Algortima Boyer-Moore

Algoritma Boyer-Moore diimplementasikan dengan bahasa pemrograman dalam dua buah fungsi. Fungsi pertama `buildLast` adalah untuk menghitung fungsi kemunculan terakhir pada algoritma Boyer-Moore dan fungsi kedua `bmMatch` untuk mencocokkan teks dan *pattern*. Berikut adalah implementasi dari algortima Boyer-Moore dalam bahasa pemrograman notasi algoritmik.

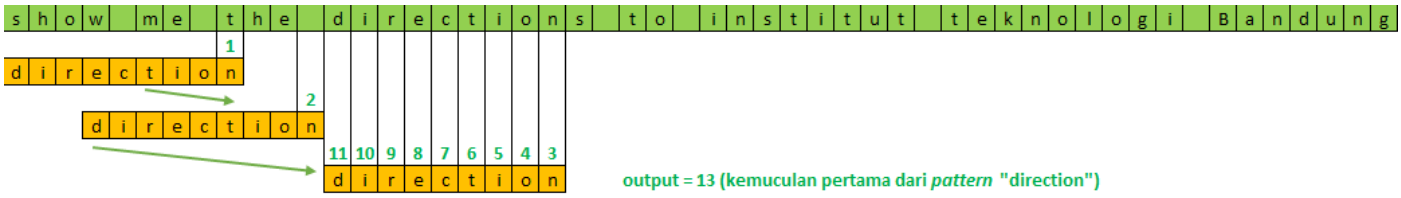
```
function bulidLast(input pattern : string) → array of integer
{ Mengembalikan array integer hasil fungsi last occurance yang memetakan karakter ASCII ke indeks kemunculan terakhirnya pada pattern atau -1 jika tidak ditemukan di pattern }

KAMUS
        last : array of integer
        i : integer

ALGORITMA
        last ← alokasi array integer dengan
               ukuran 128
        { 128 untuk jumlah karakter ASCII }
        inisialisasi elemen array last
        dengan nilai -1

        iterasi i = 0 to i < panjang pattern
          last[karakter ke i pada pattern]←i
        → last
```

```
function bmMatch(input text : string, input pattern : string) → integer
{ fungsi melakukan pencocokan string atau pola dengan algoritma Boyer-Moore, mengembalikan nilai posisi kemunculan pattern di dalam teks, atau -1 jika pattern tidak ditemukan }

KAMUS
        last : array of integer
        n : integer
        m : integer
        i : integer
        j : integer
        lo : integer

ALGORITMA
        last ← buildLast(pattern)
        n ← panjang teks
        m ← panjang pattern
        i ← m – 1 { pencocokan mulai dari }
                  { akhir pattern }
        if i > n-1 then
             → -1
          { tidak cocok karena pattern }
          { lebih panjang dari teks }
        j ← m-1
        do
          if karakter pattern indeks ke
          j = karakter teks indeks ke i then
               if j = 0 then
                  → i { cocok }
               else
                  { looking-glass technique }
                  i ← i + 1
                  j ← j + 1

        else { character jump technique }
               lo ← last[karakter teks pada
               indeks i]
               { kemunculan terakhir }
               i ← i + m – nilai terkecil
               antara j dan 1 + lo
               j ← m – 1
        while i <= n-1

        → -1 { tidak ada kecocokan }
```

Dengan definisi persoalan yang sama dengan persoalan pada bagian algoritma KMP di atas, berikut adalah pencocokan pola dengan algoritma Boyer-Moore:

Fungsi Kemuculan Terakhir untuk *pattern* "direction":

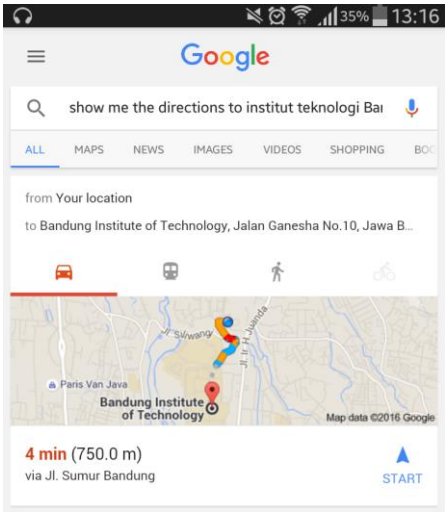
x	*c*	*d*	*e*	*i*	*n*	*o*	*r*	*t*
L(x)	5	1	4	7	9	8	3	6



Gambar 8 Perhitungan pencocokan pola dengan Boyer-Moore

Dengan menggunakan algortima Boyer-Moore, pencocokan pola mengembalikan nilai 13 merupakan indeks posisi kemunculan pertama dari *pattern* "direction". **Jumlah pencocokan adalah 11 kali**, sampai ditemukan kecocokan.

Dengan adanya perintah "direction" pada teks tesebut, maka diidentifikasi bahwa perintah yang dimasukkan pengguna adalah untuk mencari arah atau rute perjalanan dengan tujuan "insitut teknologi Bandung" dari posisi pengguna saat ini, Cisitu Baru, yang diketahui dari GPS pada perangkat Android. Selanjutnya Google Now akan menampilkan hasil pencarian seperti di bawah ini :



Gambar 9 Hasil informasi rute perjalanan dari Cisitu Baru ke Institut Teknologi Bandung

IV. Simpulan

Berdasarkan perhitungan pencocokan pola pada kasus diatas, algoritma pencocokan pola atau pencocokan string seperti algoritma Knuth-Morris-Pratt dan Boyer-Moore dapat digunakan dalam aplikasi Android seperti asisten pribadi cerdas (*Intelligent Personal Assistant*) Google Now untuk mengenali jenis perintah yang diberikan pengguna. Pencocokan dilakukan antara teks yang dihasilkan dari transformasi ucapan perintah pengguna (*speech-to-text*) dan kata kunci perintah (*commands*) yang telah didefinisikan oleh pengembang Google Now. Pada contoh algoritma Boyer-Moore membutuhkan 11 kali pencocokan sampai ditemukan *pattern* pada teks, lebih cepat disbanding algortima KMP, 21 kali pencocokan. Oleh karena itu, untuk penggunaan Bahasa Inggris atau bahasa sehari-hari lainnya pencocokan pola dengan algoritma Boyer-Moore lebih cepat dan efisien bila dibandingkan dengan algoritma KMP.

Referensi

[1] Artikel "*Google serves up information before you even knoe you need it*" pada http://www.androidcentral.com/google-now diakses pada 6 Mei 2016

[2] Artikel "*The complete list of 'OK, Google' commands*" pada http://www.cnet.com/how-to/complete-list-of-ok-google-commands/ diakses pada 8 Mei 2016.

[3] Slide kuliah IF2211 Strategi Algoritma Pencocokan String (2015) di-*update* oleh Dr.Rinaldi Munir, Informatika, STEI, Institut Teknologi Bandung dari *Lecture Note "Pattern Matching*", Dr. Andrew Davidson, WiG Lab (teachers room), CoE, Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University (PSU). Homepage: https://fivedots.coe.psu.ac.th/~ad/

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

Nursyahrina (13513060)