Ekstraksi Informasi dari Teks Alamat dengan Pattern Matching

Helkia Yeremia - 13519056
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519056@std.stei.itb.ac.id

***Abstrak* — Penulisan alamat merupakan bagian penting dari transaksi online. Biasanya alamat ditulis dalam bentuk kalimat dan tidak seluruhnya diperlukan dalam proses pengiriman barang. Pada makalah ini akan dibahas proses ekstraksi informasi dari teks alamat dengan menggunakan pattern matching. Algoritma yang digunakan adalah KMP, Boyer Moore, dan Regular expression. Dengan menggunakan pattern matching informasi dari teks alamat dapat diekstraksi dengan baik namun terbatas pada pola – pola yang didefinisikan.**

Kata Kunci* — *Pattern matching, ekstraksi alamat, regular expression

I. Pendahuluan

Di masa pandemi ini, sebagian besar kegiatan sehari – hari tidak dapat dilakukan dengan normal. Mulai dari kegiatan belajar – mengajar, pekerjaan kantor, sampai kegiatan jual – beli. Untuk itu agar tetap produktif perlu suatu cara agar kegiatan – kegiatan tersebut dapat tetap dilaksanakan. Salah satunya dengan melakukannya dari rumah secara daring.

Salah satu pihak yang paling diuntungkan dengan adanya pandemi ini adalah perusahaan e-commerce seperti shopee dan tokopedia. Berdasarkan berita yang dirilis CNN Indonesia pada 21 Oktober 2020, Transaksi e – commerce naik hampir dua kali lipat saat pandemi. Sampai Agustus 2020, telah terjadi 140 juta transaksi. Sementara pada 2019 hanya terdapat 80 juta transaksi.

Gambar 1 E – commerce di Indonesia

Sumber:(muinton.com)

Salah satu hal yang penting dalam suatu transaksi pembelanjaan online adalah pengisian alamat. Saat melakukan transaksi online, pembeli akan diminta mengisi alamat pengantaran barang. Alamat ini akan digunakan oleh kurir untuk mengantarkan barang ke rumah pemesan. Untuk itu pengisian alamat perlu selengkap mungkin agar barang dapat sampai ke tujuan yang benar.

Sebagian besar kurir yang digunakan oleh perusahaan – perusahaan e-commerce biasanya membutuhkan informasi alamat berupa nama jalan, kelurahan, kecamatan, kota, dan kode pos. Namun seringkali alamat yang di input user berupa kalimat panjang yang berisi informasi – informasi tambahan yang kurang membantu. Untuk itu pada makalah ini akan dicoba melakukan ekstraksi alamat dari sebuah kalimat menjadi informasi – informasi yang dibutuhkan kurir. Ekstraksi alamat pada makalah ini akan dilakukan dengan pattern matching (kombinasi algoritma knuth morris pratt/KMP, algoritma Boyer Moore/BM dan Regular expression).

II. Landasan Teori

A. *Pattern Matching*

Pattern Matching merupakan sebuah persoalan untuk mencari lokasi pertama dari suatu pola di dalam sebuah teks.

Contoh :

T : The rain in spain stays mainly on the plain

P : main

Sumber:(slide kuliah)

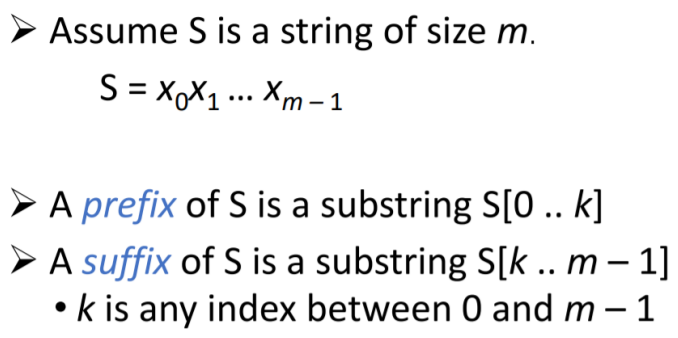
Pada contoh diatas pola main ditemukan pada lokasi ke-25 pada teks. Untuk mendapatkan lokasi pola dapat dilakukan dengan beberapa cara seperti algoritma knuth morris pratt/KMP, algoritma Boyer Moore/BM dan Regular expression.

Terdapat berbagai macam aplikasi dari pattern matching pada kehidupan sehari – hari. Beberapa diantaranya adalah pencarian suatu kata dalam editor text, web search engine, analisis citra dan bioinformatics (pencocokan rantai asam amino pada rantai DNA)

B. *String*

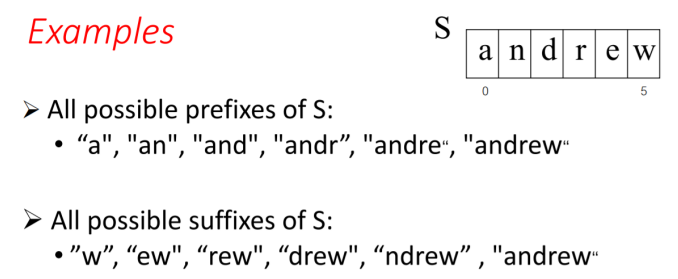
String merupakan kumpulan dari beberapa karakter. String bisa juga dianggap sebagai array of character. Sebuah string

akan memiliki sebuah prefix dan suffix. Prefix merupakan substring yang dimulai dari awal string. Sementara suffix merupakan substring yang dimulai dari akhir string.



Gambar 2.B.1 Prefix dan Suffix

Sumber:(slide kuliah)



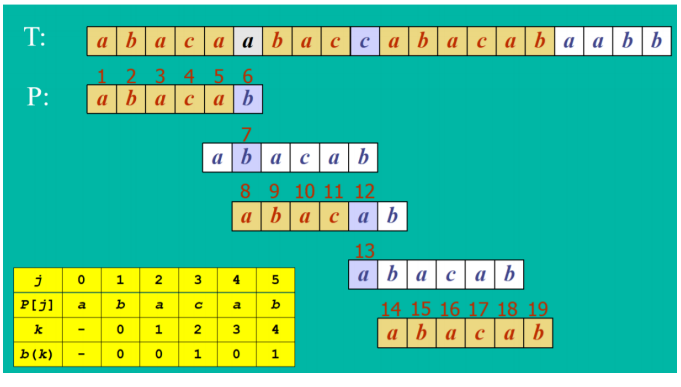
Gambar 2.B.2 contoh Prefix dan Suffix

Sumber:(slide kuliah)

C. *Knuth Morris Pratt algorithm*

Knuth Morris Pratt algorithm atau KMP merupakan salah satu algoritma yang dapat digunakan untuk menyelesaikan persoalan pattern matching. Algoritma ini mencari pola dalam suatu teks dengan bergerak dari kiri ke kanan. Algoritma ini mirip dengan bruteforce namun sedikit lebih baik dalam pergeseran pola.

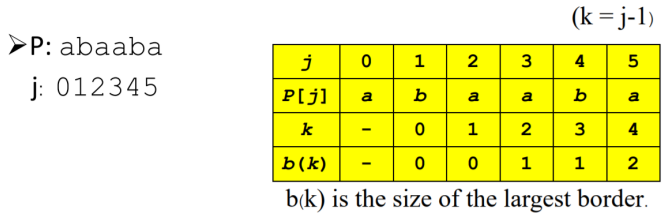
Ide utama dari algoritma ini adalah jika terjadi mismatch (T[i] ≠ P[j]) kita menggeser pola sesedikit mungkin untuk mencegah perbandingan karakter yang tidak diperlukan. Hal ini dapat dilakukan dengan melakukan pergeseran pola sebesar prefix terbesar yang sama dengan suffix.



Gambar 2.C.1 contoh KMP

Sumber:(slide kuliah)

KMP memiliki sebuah properti yang bernama fungsi pinggiran (KMP border function). Fungsi pinggiran ini akan menentukan indeks baru dari pola jika terjadi mismatch. KMP border fungsi akan mencari nilai prefix terbesar yang sama dengan suffix jika terjadi mismatch pada pola indeks ke – j.



Gambar 2.C.2 KMP border function

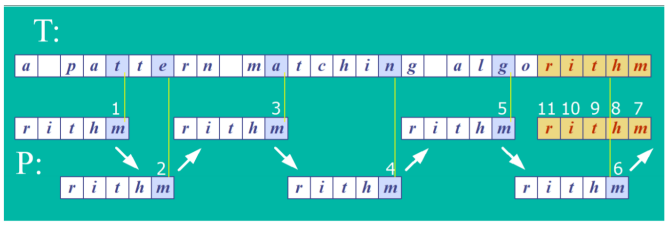
Sumber:(slide kuliah)

Kompleksitas waktu untuk menghitung fungsi pinggiran KMP adalah O(m). Kompleksitas waktu untuk pencarian string adalah O(n). Kompleksitas waktu algoritma KMP secara keseluruhan adalah O(m+n). Kompleksitas waktu ini jauh lebih cepat dibandingkan dengan brute force.

Salah satu kelebihan algoritma KMP adalah algoritma ini tidak pernah bergerak mundur terhadap teks input. Hal ini membuatnya baik untuk memproses file berukuran besar. Walaupun demikian, algoritma ini kurang baik jika alphabet dari pola dan teks sangat bervariasi karena akan menyebabkan banyak mismatch dan mismatch akan terjadi pada awal pattern.

D. *Boyer Moore Algorithm*

Seperti KMP, Boyer Moore/BM juga merupakan salah satu algoritma yang dapat digunakan untuk menyelesaikan persoalan pattern matching. Algoritma Boyer Moore didasarkan pada dua hal utama.



Jumlah perbandingan karakter: 11 kali

Gambar 2.D.1 contoh Booyer Moore

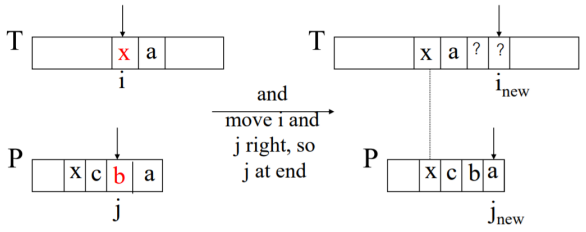
Sumber:(slide kuliah)

Yang pertama adalah looking glass technique yaitu melakukan perbandingan pattern dan text dimulai dari karakter terakhir pattern kemudian bergerak ke kiri.

Yang kedua adalah character jump technique. Yaitu cara melakukan pergeseran saat terjadi mismatch. Terdapat tiga kasus dalam character jump technique.

Case 1

➢If P contains x somewhere, then try to *shift P* right to align the last occurrence of x in P with T[i].

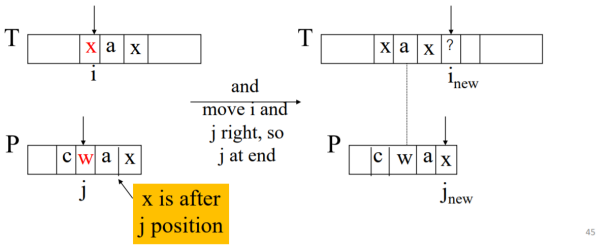


Gambar 2.D.2 case 1 character jump technique

Sumber:(slide kuliah)

Case 2

➢If P contains x somewhere, but a shift right to the last occurrence is *not* possible, then *shift P* right by 1 character to T[i+1].

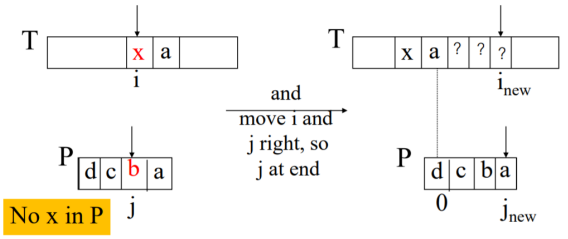


Gambar 2.D.3 case 2 character jump technique

Sumber:(slide kuliah)

Case 3

➢If cases 1 and 2 do not apply, then *shift* P to align P[0] with T[i+1].



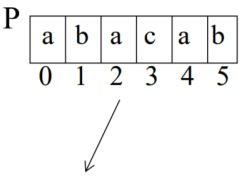
Gambar 2.D.4 case 3 character jump technique

Sumber:(slide kuliah)

Untuk dapat melakukan character jump technique, Algoritma Boyer Moore membutuhkan sebuah fungsi yang disebut last occurrence function. Fungsi ini akan menerima suatu karakter dan mengembalikan indeks terbesar pada pattern saat karakter tersebut muncul. Fungsi ini biasanya dipanggil saat pattern sudah diketahui. Dan nilai indeks dari kemunculan terakhir tiap karakter akan disimpan dalam sebuah array.

L() Example

- A = {a, b, c, d}
- P: "abacab"



x	*a*	*b*	*c*	*d*
L(x)	4	5	3	-1

L() stores indexes into P[]

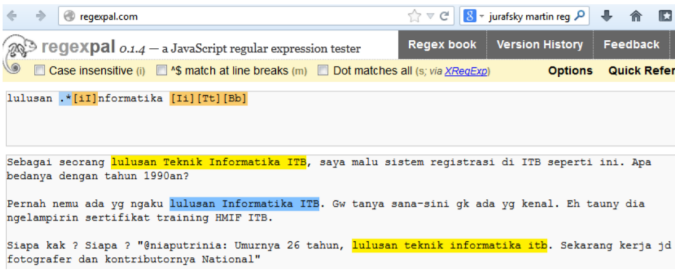
Gambar 2.D.4 last occurence

Sumber:(slide kuliah)

Untuk skenario terburuk, kompleksitas waktu algoritma Boyer Moore adalah O(nm+A). Algoritma ini berfungsi paling baik saat variasi alphabetnya banyak karena akan menghindari banyak perbandingan yang tidak diperlukan. Karena hal tersebut algoritma ini baik digunakan untuk mencari kata dalam suatu teks.

E. Regular Expression

Regular expression merupakan sebuah cara mencari pola dalam suatu teks berdasarkan serangkaian ekspresi. Pencarian pola dengan regular expression dilakukan dengan menyusun sebuah pola berdasarkan aturan – aturan tertentu. Kemudian pola tersebut akan dicari pada teks.



Gambar 2.E.1 Regex

Sumber:(slide kuliah)

Berikut beberapa ekspresi yang biasa digunakan saat melakukan pattern matching menggunakan regular expression.

- The use of the brackets [ ] to specify a disjunction of characters.

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/[abc]/	'a', 'b', *or* 'c'	"In uomini, in soldati"
/[1234567890]/	any digit	"plenty of 7 to 5"

- The use of the brackets [ ] plus the dash – to specify a range.

RE	Match	Example Patterns Matched
/[A-Z]/	an uppercase letter	"we should call it 'Drenched Blossoms'"
/[a-z]/	a lowercase letter	"my beans were impatient to be hoed!"
/[0-9]/	a single digit	"Chapter 1: Down the Rabbit Hole"

- Uses of the caret ^ for negation or just to mean ^

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase letter	"Oyfn pripetchik"
[^Ss]	neither 'S' nor 's'	"I have no exquisite reason for't"
[^\.]	not a period	"our resident Djinn"
[e^]	either 'e' or '^'	"look up ^ now"
a^b	the pattern 'a^b'	"look up a^b now"

- The question-mark ? marks optionality of the previous expression.

RE	Match	Example Patterns Matched
woodchucks?	woodchuck or woodchucks	"woodchuck"
colou?r	color or colour	"colour"

- The use of period . to specify any character

RE	Match	Example Patterns
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

Gambar 2.E.2 contoh notasi Regex

Sumber:(slide kuliah)

III. IMPLEMENTASI

A. *Program Utama*

Pada implementasi, program akan menerima input sebuah kalimat panjang kemudian akan diambil beberapa informasi berikut dari kalimat tersebut :

1. Nama jalan / Alamat utama
2. Nama kelurahan
3. Nama kecamatan
4. Nama kota
5. Kode pos

Untuk mengambil nama jalan/alamat utama, program dapat mengenalinya melalui beberapa pola regular expression. Pola (.*) akan diambil sebagai nama jalan/alamat utama

- (.*) ((kel.?|kelurahan) {})

  tanda {} diisi dengan nama kelurahan yang terdaftar
- (.*) ((kec.?|kecamatan) {})

  tanda {} diisi dengan nama kecamatan yang terdaftar
- (.*) ((kab.?|kabupaten|kot.?|kota) {})

  tanda {} diisi dengan nama kabupaten atau kota yang terdaftar
- (.*) {}

  tanda {} dapat diisi dengan nama kelurahan, kecamatan, kabupaten, atau kota yang terdaftar

Untuk mengambil nama kelurahan digunakan algoritma kmp. Program akan melakukan loop pada daftar nama kelurahan yang terdaftar berdasarkan kecamatan kemudian mencari nama kelurahan yang sesuai dengan teks alamat yang diterima.

Untuk mengambil nama kecamatan digunakan algoritma kmp. Program akan melakukan loop pada daftar nama kecamatan yang terdaftar berdasarkan kota kemudian mencari nama kecamatan yang sesuai dengan teks alamat yang diterima.

Untuk mengambil nama kota/kabupaten digunakan algoritma kmp. Program akan melakukan loop pada daftar nama kota/kabupaten yang terdaftar kemudian mencari nama kabupaten/kota yang sesuai dengan teks alamat yang diterima.

Untuk mengambil kodepos digunakan regular expression dengan pola sebagai berikut (kode pos Indonesia terdiri dari 5 digit). [1-9]{5}

```
   Program Ekstrak Alamat
   kelurahan              : array [1..l] of
string
   kecamatan              : array [1..m] of
string
   kabupatenKota    : array [1..n] of string
   alamatUtamaPattern     : regex pattern
   kodePosPattern   : regex pattern


   input(alamat)


   for i ← 1 to n do
      if        (kmpMatch(kabupatenKota[i],
alamat) = True) then
            tempKota ← kabupatenKota[i]


   for j ← 1 to m do
      if  (kmpMatch(kecamatan[i], alamat) =
True) then
            tempKecamatan ← kecamatan[i]


   for k ← 1 to l do
      if  (kmpMatch(kelurahan[i], alamat) =
True) then
            tempKelurahan ← kelurahan[i]


   tempAlamatUtama                          ←
search(alamatUtamaPattern, alamat)


   tempKodePos    ←    search(kodePosPattern,
alamat)


   output(tempAlamatUtama)
   output(tempKelurahan)
```

```
output(tempKecamatan)
output(tempKota)
    output(tempKodePos)
```

Pseudocode program utama

B. Algoritma KMP

Untuk implementasi algoritma KMP, digunakan dua fungsi yang berbeda. Yang pertama adalah fungsi compute fail. Fungsi ini akan menerima input sebuah pola dan mengembalikan array of integer yang menyimpan besar pergeseran yang harus dilakukan jika terjadi mismatch pada indeks tersebut.

```
function computefail(pattern:string) → array[1..n] of integer

  fail      : array[1..n] of integer
  i,j,m : integer

  for i ← 1 to n do
      fail[i] ← -1
  fail[0] ← 0
  m ← n {length of pattern}
  j ← 0
  i ← 1
  while (i < m) do
      if (pattern[j] = pattern[i]) then
            fail[i] ← j + 1
            i ← i + 1
            j ← j + 1
      else if (j > 0) then
            j ← fail[j-1]
      else then
            fail[i] ← 0
            i ← i + 1
      endif
  endwhile
      → fail
```

Pseudocode border function

Yang kedua adalah fungsi kmpMatch. Fungsi ini akan menerima sebuah pola dan sebuah text. Fungsi ini bertugas mencari pola dari dalam text tersebut. Jika pola ditemukan fungsi akan mengembalikan nilai true, jika sebalika fungsi akan mengembalikan nilai false.

```
function kmpMatch(pattern, text) → boolean

  n   : integer {length of text}
  m   : integer {length of pattern}
  i,j       : integer
  fail      : array[1..m] of integer

  fail ← computefail(pattern)
  i ← 0
  j ← 0

  while (i < n) do
      if (pattern[j] = text[i]) then
            if (j = m -1) then
                  → True
            i ← i + 1
            j ← j + 1
      else if (j > 0) then
            j ← fail[j-1]
      else
            i ← i + 1
      endif
  endwhile
      → False
```

Pseudocode kmpMatch

C. Pengujian

Untuk melakukan pengujian penulis menggunakan sebuah alamat di daerah Tangerang. Namun untuk menguji program penulisannya dibedakan menjadi 3.

- Komplek taman asri blok G7 no.16 kel. gaga kecamatan larangan Tangerang 15154
- Komplek taman asri blok G7 no.16 kecamatan larangan Tangerang 15154
- Komplek taman asri blok G7 no.16 Tangerang 15154

Setelah ketiga input dimasukkan ke program, hasil yang didapat adalah :

- Input 1:

```
Nama jalan          : komplek taman asri blok g7 no.16
Kelurahan           : gaga
Kecamatan           : larangan
Kabupaten/Kota      : tangerang
Kodepos             : 15154
```

- Input 2:

```
Nama jalan          : komplek taman
asri blok g7 no.16
Kelurahan           :
Kecamatan           : larangan
Kabupaten/Kota      : tangerang
Kodepos             : 15154
```

- Input 3

```
Nama jalan          : komplek taman
asri blok g7 no.16
Kelurahan           : gaga
Kecamatan           : larangan
Kabupaten/Kota      : tangerang
Kodepos             : 15154
```

D. Pembahasan

Berdasarkan hasil yang didapat dari poin C, program sudah berhasil mengekstraksi informasi – informasi yang diperlukan dari teks alamat yang diinput. Walaupun pada input kedua dan ketiga ada informasi yang tidak lengkap, program masih berhasil mengekstrak informasi yang dimasukkan user. Biarpun begitu, dari beberapa hasil pengujian penulis, program memerlukan minimal nama jalan, salah satu dari kelurahan, kecamatan, atau kabupaten/kota serta kode pos agar bisa berjalan dengan baik. Urutan input user juga penting karena untuk mengambil nama jalan diperlukan urutan input yang benar.

Menurut penulis kemampuan program juga masih sangat terbatas. Hal ini dikarenakan setiap kemungkinan pola yang muncul masih harus dipikirkan secara manual oleh penulis sehingga program hanya bisa mengekstraksi alamat dengan pola pola tertentu saja.

IV. Kesimpulan

Dengan pencocokan string, proses ekstraksi alamat menjadi lebih cepat dan otomatis. Walaupun begitu, menurut penulis solusi ekstraksi alamat dengan pencocokan kurang optimal. Hal ini disebabkan beragamnya cara penulisan alamat setiap orang yang menyebabkan pembuat program harus membuat banyak pola untuk menangani berbagai kasus.

Video Link at Youtube

Penjelasan tentang makalah ini dapat dilihat pada link berikut : https://youtu.be/QytVmBWRFbY

Acknowledgment

Penulis bersyukur kepada Tuhan Yang Maha Esa yang memberikan anugrah kesehatan bagi penulis untuk bisa membuat makalah ini. Penulis juga berterima kasih kepada Dr. Nur Ulfa Maulidevi, S.T., M.Sc sebagai dosen mata kuliah IF2211 Strategi Algoritma untuk K-02 beserta seluruh tim dosen IF2211 yang telah mengajari penulis dan teman – teman lainnya dengan sabar. Penulis juga mengucapkan terima kasih kepada seluruh pihak yang telah berkontribusi dalam penulisan makalah ini.

References

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf diakses pada 10 Mei 2021

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf diakses pada 10 Mei 2021

[3] https://www.cnnindonesia.com/ekonomi/20201021193353-92-561232/transaksi-e-commerce-naik-nyaris-dua-kali-lipat-saat-pandemi diakses pada 10 Mei 2021

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, 29 April 2020

Helkia Yeremia 13519056