

Penerapan Algoritma Pencocokan String pada Pencarian *Record* Tabel

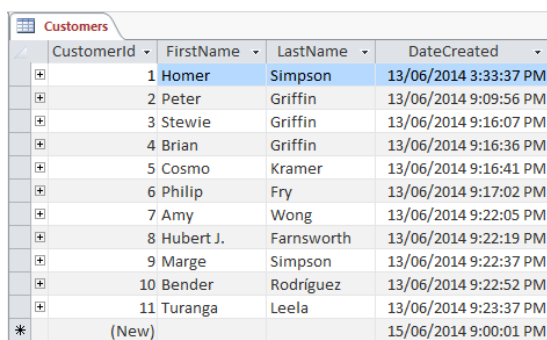
Rezda Abdullah Fachrezzi 13519194
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
Surel: 13519194@std.stei.itb.ac.id

Abstrak—Dalam kehidupan, pasti kita sering bertemu dengan data-data, apapun data itu dan bentuknya. Contoh dari data-data ini adalah data mahasiswa untuk suatu jurusan di kampus tertentu, data dosen untuk suatu kampus, dan banyak lainnya. Salah satu contoh bentuk dari data-data tersebut adalah data tabel. Terkadang, kita ingin mencari data dari suatu kata kunci saja, contohnya pada data tabel kita ingin mencari suatu data mahasiswa dengan nomor induk tertentu. Nomor induk tersebutlah yang kemudian menjadi kata kunci. Pencarian kata kunci ini yang kemudian diselesaikan dengan algoritma pencocokan string. Contoh dari algoritma pencocokan string yang ada adalah Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Ekspresi Regular.

Kata Kunci—algoritma; pencocokan; string; pencarian; tabel

I. PENDAHULUAN

Sebagai mahasiswa, sudah merupakan hal yang lumrah dan wajar apabila kita sering bertemu dengan data. Data ini banyak macamnya, salah satu contohnya adalah data mahasiswa pada suatu program studi pada suatu universitas, data dosen pada suatu universitas, data penderita penyakit COVID-19, dan banyak lainnya. Data ini juga banyak model penyajiannya, salah satunya adalah data tabel.



CustomerId	FirstName	LastName	DateCreated
1	Homer	Simpson	13/06/2014 3:33:37 PM
2	Peter	Griffin	13/06/2014 9:09:56 PM
3	Stewie	Griffin	13/06/2014 9:16:07 PM
4	Brian	Griffin	13/06/2014 9:16:36 PM
5	Cosmo	Kramer	13/06/2014 9:16:41 PM
6	Philip	Fry	13/06/2014 9:17:02 PM
7	Amy	Wong	13/06/2014 9:22:05 PM
8	Hubert J.	Farnsworth	13/06/2014 9:22:19 PM
9	Marge	Simpson	13/06/2014 9:22:37 PM
10	Bender	Rodriguez	13/06/2014 9:22:52 PM
11	Turanga	Leela	13/06/2014 9:23:37 PM
*	(New)		15/06/2014 9:00:01 PM

Gambar 1 Contoh data tabel sederhana yang menyimpan nama pelanggan.

(Sumber: <https://database.guide/what-is-a-table/>)

Umumnya, saat kita sedang mengkaji atau memproses data, kita akan sering menemukan kondisi saat kita harus mencari data yang spesifik dari data tersebut. Sebagai contoh pada gambar 1, kita ingin mencari nama pelanggan dengan nama “Amy”. Maka kita akan tahu bahwa pelanggan dengan nama “Amy” memiliki customerId “7” dan LastName “Wong”.

Tetapi, apakah kita tahu bagaimana proses pencarian tersebut bekerja?

Salah satu pendekatan pencarian ini adalah dengan menggunakan Algoritma Pencocokan String. Pada Algoritma Pencocokan String, terdapat komponen penting yang bernama kata kunci atau *pattern*. Kata kunci atau *pattern* inilah yang kemudian akan digunakan untuk mencocokkan tiap-tiap baris dari data tabel gambar 1. Pada contoh yang sudah disebutkan, kata kunci yang dipakai adalah kata “Amy”, sedangkan data-data yang dicocokkan adalah kolom FirstName pada tiap-tiap baris dari tabel tersebut.

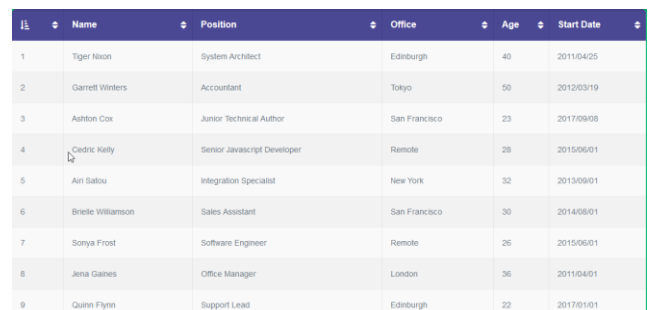
Algoritma Pencocokan String ini ada banyak macamnya, seperti Brute Force, Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Ekspresi Regular. Namun, pada makalah ini, penulis hanya akan mengaplikasikan algoritma KMP, BM, dan Ekspresi Regular saja.

II. LANDASAN TEORI

A. Tabel

1. Definisi

Tabel, menurut Kamus Besar Bahasa Indonesia, adalah daftar berisi ikhtisar sejumlah (besar) data informasi, biasanya berupa kata-kata dan bilangan yang tersusun secara sistematis, urut ke bawah dalam lajur dan deret tertentu dengan garis pembatas sehingga dapat dengan mudah disimak [6].



ID	Name	Position	Office	Age	Start Date
1	Tiger Nixon	System Architect	Edinburgh	40	2011-04-25
2	Garrett Winters	Accountant	Tokyo	50	2012-03-19
3	Ashton Cox	Junior Technical Author	San Francisco	23	2017-09-08
4	Cedric Kelly	Senior Javascript Developer	Remote	28	2015-06-01
5	Ali Salou	Integration Specialist	New York	32	2013-05-01
6	Brielle Williamson	Sales Assistant	San Francisco	30	2014-05-01
7	Sonya Frost	Software Engineer	Remote	25	2015-06-01
8	Jena Jones	Office Manager	London	36	2011-04-01
9	Quinn Flynn	Support Lead	Edinburgh	22	2017-01-01

Gambar 2 Contoh tabel yang menyimpan data.

(Sumber: <https://essential-addons.com/elementor/docs/content-elements/data-table/>)

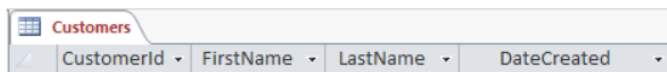
Tabel pada aplikasinya dapat menyimpan berbagai jenis data, salah satu contohnya adalah data mahasiswa pada suatu program studi pada suatu kampus. Bentuk tabel ini banyak macamnya, beberapa contohnya adalah dapat berupa basis data dengan Bahasa SQL, dalam bentuk format fail excel, ataupun dalam bentuk format *comma separated value* atau dikenal dengan CSV.

2. Komponen

Pada tabel, terdapat beberapa komponen dan istilah penting untuk diketahui.

a. Field

Field merupakan kumpulan string yang bermakna yang mewakili kolom yang dibawahinya. Pada gambar 1, field pada tabel tersebut adalah CustomerId, FirstName, LastName, dan DateCreated.



CustomerId	FirstName	LastName	DateCreated
------------	-----------	----------	-------------

Gambar 3 Contoh *field* pada tabel di gambar 1. (Sumber: <https://database.guide/what-is-a-table/>)

b. Record

Record merupakan kumpulan field yang biasanya dalam bentuk baris. Pada gambar 1, salah satu contoh recordnya adalah sebagai berikut.



5	Cosmo	Kramer	13/06/2014 9:16:41 PM
---	-------	--------	-----------------------

Gambar 4 Contoh *record* pada tabel di gambar 1. (Sumber: <https://database.guide/what-is-a-table/>)

Dapat dilihat pada gambar 4, suatu *record* menyimpan masing-masing *value* dari *field* pada tabel. Dari *record* tersebut, dapat diketahui bahwa CustomerId-nya adalah "5", FirstName-nya adalah "Cosmo", LastName-nya adalah "Kramer", dan DateCreated-nya adalah "13/06/2014 9:16:41 PM".

B. Algoritma Pencocokan String

Pencocokan string atau disebut juga *string matching* atau *pattern matching* merupakan istilah dalam dunia Informatika untuk pencarian suatu pola pada suatu string. Persoalan pencocokan string adalah misalnya diberikan suatu string dengan panjang N karakter dan suatu pola dengan panjang M karakter dengan $N \geq M$ yang kemudian akan dicari di dalam string. Algoritma pencocokan string adalah sebuah pendekatan dalam mengaplikasian hal tersebut.

Algoritma pencocokan string adalah sebuah metode yang dapat menyelesaikan persoalan pencocokan string. Algoritma pencocokan string umumnya digunakan pada pencarian kata pada dokumen, pencarian pada mesin pencari, analisis citra, dan bioinformatika, misalnya pencocokan rantai DNA.

dalam strategi **algoritma** (baik < menyelesaikan suatu masalah. ambar kontribusimu, jadi tidak .ebih bagus lagi jika makalah yang dibuat). Spesifikasi teknis nerlu dilampirkan dengan kode

Gambar 5 Contoh pencocokan string pada pencarian kata di dokumen (kata yang diwarnai adalah hasil pencarian). (Sumber: penulis)

Pada string, terdapat sebuah komponen yang pasti ada, yaitu prefiks dan sufiks. Misalnya untuk string "abcd", prefiksnya dapat berupa "a", "ab", "abc", dan "abcd", sedangkan sufiksnya dapat berupa "d", "cd", "bcd", "abcd".

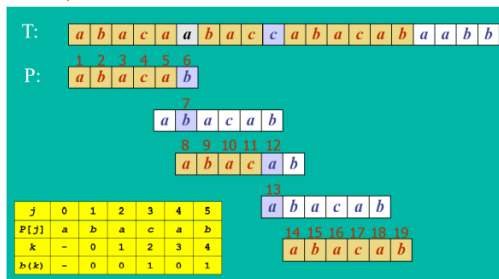
Algoritma pencocokan string umumnya membutuhkan dua komponen penting, yaitu string atau teks dan kata kunci atau *pattern* atau pola. Saat ini, terdapat beberapa contoh dari algoritma pencocokan string yang memerlukan kedua komponen tersebut, beberapa contohnya adalah Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Ekspresi Regular.

C. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt atau disingkat menjadi KMP adalah salah satu algoritma pencocokan string yang bersifat *exact matching*. *Exact matching* berarti pencocokan pola dengan string secara tepat dengan susunan dan karakter yang sama persis, baik urutan maupun jumlahnya pada pola tersebut. Algoritma ini pertama kali dikembangkan oleh Donald Knuth pada tahun 1967 dan James Morris bersama Vaughan Pratt pada tahun 1966, lalu dipublikasikan pada tahun 1977.

Algoritma ini mencari suatu bagian dari string bernama *pattern* dengan urutan pencarian dari kiri ke kanan. Pencarian ini mirip dengan algoritma brute force tetapi lebih "pintar". Jika pada algoritma brute force dilakukan perpindahan *pattern* dan mengecek satu persatu pada tiap karakter dari *pattern*, algoritma KMP melakukan perpindahan dengan metode tertentu.

Pada algoritma KMP, terdapat sebuah fungsi pinggiran yang mengatur perpindahan *pattern* tersebut pada saat melakukan pencocokan. Nama lain dari fungsi ini adalah *border function* atau *failure*. Fungsi ini menghasilkan tabel yang berisi ukuran terbesar prefiks [0..k] dengan sufiks [1..k] pada suatu *pattern*.



Gambar 6 Ilustrasi cara kerja algoritma KMP.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)

Algoritma KMP beserta fungsi pinggirannya dapat digambarkan dengan pseudocode berikut dengan P adalah pattern dengan panjang string m serta S adalah string dengan panjang string n.

```

Pinggiran(P)
i = 1
j = 0
fail[0] = 0 // fungsi pinggiran berupa larik

while i < m do
  if P[j] = P[i] then
    fail[i] = j + 1
    i = i + 1
    j = j + 1
  else if j > 0 then
    j = fail[j - 1]
  else
    fail[i] = 0
    i = i + 1
return fail
  
```

```

KMPSearch(S, P)
m = S.length
n = P.length

fail = Pinggiran(P)
i = 0; j = 0

while i < n do
  if P[j] = S[i] then
    if j = m - 1 then return i - m + 1
    j = j + 1
    i = i + 1
  else
    if j > 0 then
      j = fail[j - 1]
    else
      i = i + 1
return -1
  
```

D. Algoritma Boyer-Moore (BM)

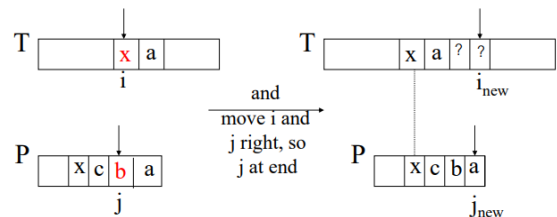
Algoritma Boyer-Moore atau disingkat menjadi BM adalah salah satu algoritma pencocokan string yang sifatnya sama dengan algoritma KMP, yaitu *exact matching*. Algoritma ini pertama dikembangkan oleh Robert S. Boyer dan J, Strother Moore dan dipublikasikan pada tahun 1977.

Berbeda dengan algoritma KMP yang urutan pencariannya dari kiri, algoritma BM melakukan pencarian dari kanan. Pada algoritma BM, terdapat dua teknik yang digunakan, yaitu *looking glass* dan *character jump*. *Looking Glass* merupakan pencarian yang dilakukan dari akhir, serta *character jump*, yaitu ketika suatu ketidakcocokan ditemukan saat string $T[i] = x$, maka *pattern* $P[j]$ tidak sama dengan $T[i]$.

Terdapat tiga kasus yang berbeda saat perpindahan *pattern* atau *character jump* dilakukan saat algoritma BM dijalankan.

1. Kasus pertama

Jika *pattern* P mengandung karakter x di suatu indeks, maka P akan digeser ke kanan sampai karakter x tersebut sejajar dengan x di string T dengan posisi x pada T merupakan indeks terakhir ditemukannya x pada P.



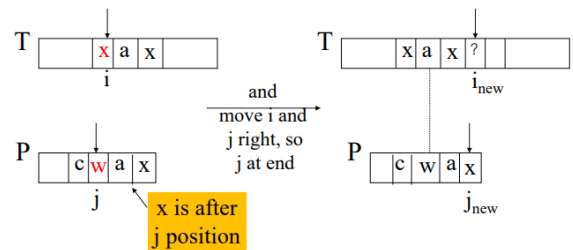
Gambar 7 Ilustrasi kasus pertama *character jump technique*.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)

2. Kasus kedua

Jika *pattern* P mengandung karakter x di suatu indeks tetapi perpindahan *pattern* P tidak dimungkinkan karena kemunculan terakhir x pada P di luar batas, maka dilakukan pergeseran *pattern* sebesar satu kali ke kanan.



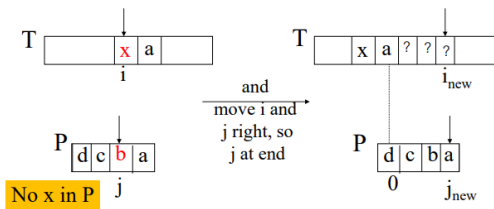
Gambar 8 Ilustrasi kasus kedua *character jump technique*.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)

3. Kasus ketiga

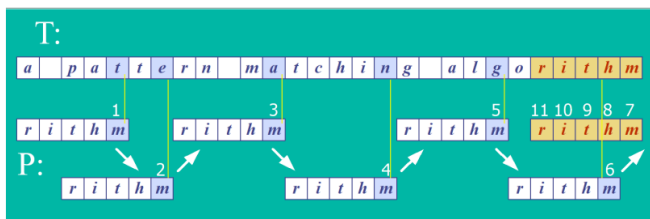
Jika kasus pertama dan kedua tidak ditemukan (karakter x tidak ada pada *pattern*), maka *pattern* P akan disejajarkan dengan karakter berikutnya pada T.



Gambar 9 Ilustrasi kasus ketiga *character jump technique*.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)



Gambar 10 Ilustrasi cara kerja algoritma BM.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)

Algoritma BM dapat digambarkan dengan *pseudocode* berikut dengan P adalah *pattern* dengan panjang string m serta T adalah string dengan panjang string n.

```

BM(T,P)
// mencari last occurrence / kemunculan terakhir pada P
last = int[128]

for i in range (128) do
    last[i] = -1

for i in range (P.length) do
    last[P.charCodeAt(i)] = i

// algoritma BM
m = P.length
n = T.length
i = m - 1

if i > n - 1 then
    return -1 // length berbeda

j = m-1
while i < n do
    if P[j] = T[i] then
        if j = 0 then return i
        else i = i - 1; j = j - 1
    else
        lo = last [P.charCodeAt(i)]

```

```

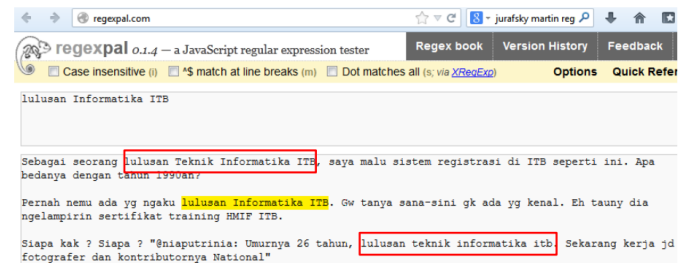
i = i + m - Min(j, 1 + lo)
j = m - 1

return -1

```

E. Ekspresi Regular

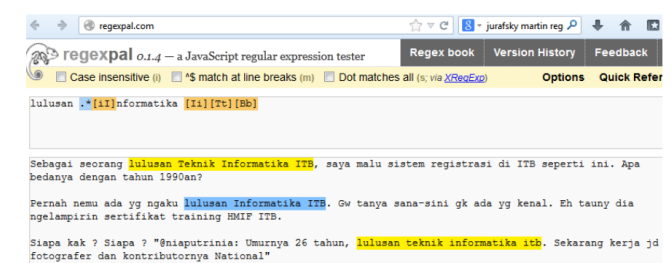
Ekspresi regular, atau dikenal dengan *Regular Expression* atau *Regex* dan disingkat RE, merupakan salah satu algoritma pencocokan string yang sangat populer. Algoritma ini lebih *powerful* daripada algoritma KMP dan BM. Alasannya adalah *Regex* tidak hanya dapat melakukan *exact matching*, tetapi juga dapat melakukan *pattern matching* atau pengenalan pola.



Gambar 11 *Exact matching* dengan *Regex*.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)

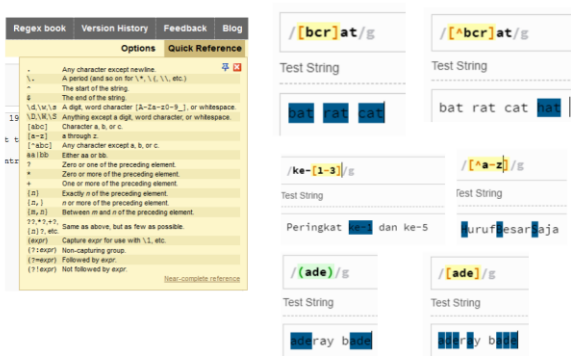


Gambar 12 *Pattern matching* dengan *Regex*.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)

Dalam *Regex*, terdapat beberapa notasi yang dapat digunakan, baik itu untuk mengenali pola tertentu seperti mencari angka ataupun karakter, mengenali jumlah karakter tertentu, mendeteksi karakter pertama dan terakhir. Masih banyak hal lainnya yang dapat dilakukan dengan *Regex*. Berikut adalah beberapa notasi yang umum digunakan.



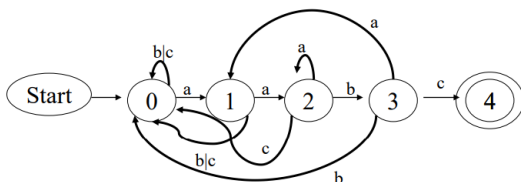
Gambar 13 Notasi yang umum digunakan pada Regex.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)

Untuk cara kerja Regex sendiri, Regex menggunakan Finite Automata sebagai pemrosesannya. Singkatnya, Finite State Automata adalah mesin komputasi yang menerima string sebagai input yang mengeluarkan sebuah output berupa *yes* atau *no*. Singkatnya, mesin ini meng-“accept” atau meng-“reject” masukan. Pada setiap notasi Regex yang dipakai, Regex akan membentuk state-state yang kemudian dapat memproses masukan. Jika state-state tersebut tidak mengarah ke solusi, finite state automata akan meng-“reject” string tersebut dan menghasilkan kesimpulan bahwa pola untuk suatu string tersebut tidak ditemukan.

- Alphabet {a,b,c}
- Pattern “aabc”
- String: aaaaaaaaaabcccccccccccccc



Gambar 14 Contoh FSM untuk *pattern* “aabc”.

(Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>)

III. IMPLEMENTASI DAN PENGUJIAN

Pada makalah ini, penulis melakukan implementasi pencarian *record* dari tabel data dan pengujian menggunakan Bahasa Python. Penulis mengimplementasikan tiga algoritma pencocokan string atau pola, yakni algoritma KMP, algoritma BM, dan ekspresi regular.

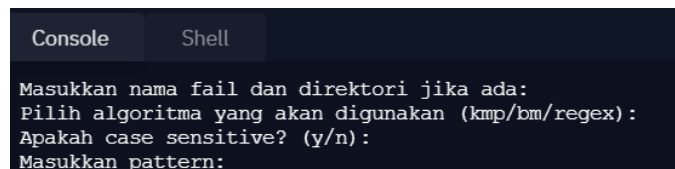
A. Implementasi

Penulis menggunakan tabel data dalam format CSV sebagai tabel yang akan digunakan saat pengujian pada program. Implementasi algoritma KMP dan algoritma BM menyesuaikan dengan materi yang diajarkan pada kuliah IF2211 [1] serta yang tertulis pada Landasan Teori (Bab 2).

Sedangkan implementasi Ekspresi Regular memanfaatkan kaskas **re** dari Python.

Program yang dirancang memiliki alur kerja sebagai berikut.

1. Program akan meminta nama direktori dan fail CSV.
2. Program akan meminta nama algoritma yang akan digunakan.
3. Program akan menanyakan apakah *pattern* yang akan dimasukkan nanti *case sensitive* atau tidak.
4. Program akan meminta *pattern* untuk kemudian dicocokkan dengan tiap record dari tabel.
5. Program kemudian akan mencari *record* pada tabel yang cocok dengan *pattern* tersebut.
6. Jika kecocokan ditemukan, program akan menampilkan indeks *record* serta *field* lainnya. Program dapat menampilkan lebih dari satu *record*.
7. Jika kecocokan tidak ditemukan, program akan menampilkan pesan yang memberi tahu bahwa kecocokan tidak ditemukan.
8. Program akan menampilkan jumlah waktu yang dihabiskan dalam detik (opsional).



Gambar 15 Masukan-masukan yang diminta program.

(Sumber: Penulis)

Komponen-komponen yang terdapat pada algoritma pencocokan string adalah teks atau string serta pola atau *pattern*. *Pattern* merupakan masukan dari pengguna yang kemudian akan dicocokkan dengan *record-record* yang ada pada tabel.

Pendekatan pencarian yang dilakukan adalah pertama-tama program akan membaca fail yang dimasukkan oleh pengguna. Fail itu kemudian akan diubah menjadi bentuk *list of records*. Dari *list of records* tersebut, masing-masing *records* kemudian diubah menjadi sebuah string, mencakup keseluruhan *field*-nya.

Sebagai contoh pada *record* dari gambar 4, *record* tersebut akan dikonversikan menjadi string dalam bentuk:

5 – Cosmo – Kramer – 13/06/2014 9:16:41 PM

String hasil konversi inilah yang menjadi komponen pertama pada algoritma pencocokan string. *List of records* hasil pembacaan fail tadi kemudian dikonversikan seluruhnya menjadi sebuah *list of string* yang mengandung seluruh *records* tabel dalam bentuk string.

Langkah selanjutnya setelah mengonversikan *list of records* menjadi *list of string* adalah mencocokkan *pattern* yang sudah dimasukkan pengguna dengan *list of string* tersebut. *List*

tersebut diiterasikan dan dicocokkan satu persatu dengan *pattern* dengan algoritma dan kondisi sesuai masukan dari pengguna. Jika terdapat kecocokan, string tersebut akan dimasukkan ke larik baru yang kemudian akan diperlihatkan ke pengguna. Setelah selesai dicocokkan semuanya, program juga akan menghitung lamanya waktu eksekusi dalam detik yang kemudian dapat ditampilkan ke pengguna.

B. Pengujian

Pengujian dilakukan dengan menggunakan tabel data daftar mahasiswa ASYNC Himpunan Mahasiswa Informatika Institut Teknologi Bandung (HMIF ITB). Tabel yang digunakan memiliki *field* NIM dan Nama Lengkap. Tabel memiliki 340 baris *record* termasuk baris pertama yang merupakan nama-nama *field*. Pengujian dilakukan di peladen awan Repl dengan sistem operasi Ubuntu 18.04.5 LTS dengan *memory* 500 Megabytes.

Sebelum dilakukan pengujian, berikut merupakan beberapa catatan penting terkait program yang dirancang.

1. Program mengasumsikan bahwa semua masukan pengguna adalah valid. Valid memiliki artian yakni:
 - a. Fail dengan nama dan direktori terkait ada.
 - b. Fail yang dimasukkan merupakan fail CSV.
 - c. Seluruh masukan yang diminta adalah sesuai dengan permintaan program.
2. Fail CSV yang dimasukkan memiliki baris pertama berupa nama-nama *field*.
3. Seluruh algoritma yang dipakai merupakan *exact matching* (termasuk RegEx).

Setelah hal-hal tersebut diperhatikan, selanjutnya program dapat langsung diujikan. Langkah pengujian yang dilakukan adalah sebagai berikut.

1. Menjalankan fail program
2. Mengisi masukan-masukan yang diminta oleh program sesuai dengan jenis pengujian.
3. Mengamati hasil.

Berdasarkan langkah-langkah pengujian yang sudah ditulis sebelumnya, selanjutnya adalah melakukan pengujian berdasarkan jenis-jenis pengujian. Berikut adalah jenis-jenis pengujian yang dilakukan serta hasilnya.

1. Pengujian 1 – Algoritma KMP

Melakukan pengujian algoritma KMP dengan *pattern* “Rez” dengan menguji dua kondisi, yaitu *case sensitive* dan *case insensitive*.

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): kmp
Apakah case sensitive? (y/n): y
Masukkan pattern: Rez

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi
284 - 18219064 - Muhammad Reza Nur Fauzi

=====
Menghabiskan 0.00775 detik
```

Gambar 16 Pengujian 1 – *Case sensitive*.
(Sumber: Penulis)

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): kmp
Apakah case sensitive? (y/n): n
Masukkan pattern: Rez

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi
256 - 18219036 - Zachrandika Alif Syahreza
278 - 18219058 - Afif Fahreza
284 - 18219064 - Muhammad Reza Nur Fauzi

=====
Menghabiskan 0.00914 detik
```

Gambar 17 Pengujian 1 – *Case insensitive*.
(Sumber: Penulis)

2. Pengujian 2 – Algoritma BM

Melakukan pengujian algoritma BM dengan *pattern* “Rez” dengan menguji dua kondisi, yaitu *case sensitive* dan *case insensitive*.

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): bm
Apakah case sensitive? (y/n): y
Masukkan pattern: Rez

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi
284 - 18219064 - Muhammad Reza Nur Fauzi

=====
Menghabiskan 0.00558 detik
```

Gambar 18 Pengujian 2 – *Case sensitive*.
(Sumber: Penulis)

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): bm
Apakah case sensitive? (y/n): n
Masukkan pattern: Rez

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi
256 - 18219036 - Zachrandika Alif Syahreza
278 - 18219058 - Afif Fahreza
284 - 18219064 - Muhammad Reza Nur Fauzi

=====
Menghabiskan 0.00395 detik
```

Gambar 19 Pengujian 2 – *Case insensitive*.
(Sumber: Penulis)

3. Pengujian 3 – Ekspresi Regular

Melakukan pengujian Ekspresi Regular dengan *pattern* “Rez” dengan menguji dua kondisi, yaitu *case sensitive* dan *case insensitive*.

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): regex
Apakah case sensitive? (y/n): y
Masukkan pattern: Rez

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi
284 - 18219064 - Muhammad Reza Nur Fauzi

Menghabiskan 0.00137 detik
```

Gambar 20 Pengujian 3 – *Case sensitive*.
(Sumber: Penulis)

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): regex
Apakah case sensitive? (y/n): n
Masukkan pattern: Rez

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi
256 - 18219036 - Zachrandika Alif Syahreza
278 - 18219058 - Afif Fahreza
284 - 18219064 - Muhammad Reza Nur Fauzi

Menghabiskan 0.00112 detik
```

Gambar 21 Pengujian 3 – *Case insensitive*.
(Sumber: Penulis)

4. Pengujian 4 – Pencarian berdasarkan *field* tertentu

Melakukan pengujian ketiga algoritma tetapi pencarian berdasarkan *field* tertentu. Dalam pengujian ini akan diuji dengan *field* NIM dengan *pattern* “13519194”.

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): kmp
Apakah case sensitive? (y/n): n
Masukkan pattern: 13519194

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi

Menghabiskan 0.0062 detik
```

Gambar 22 Pengujian 4 – Algoritma KMP.
(Sumber: Penulis)

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): bm
Apakah case sensitive? (y/n): n
Masukkan pattern: 13519194

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi

Menghabiskan 0.00331 detik
```

Gambar 23 Pengujian 4 – Algoritma BM. (Sumber: Penulis)

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): regex
Apakah case sensitive? (y/n): n
Masukkan pattern: 13519194

=====HASIL=====
indeks - [field]
193 - 13519194 - Rezda Abdullah Fachrezzi

Menghabiskan 0.00133 detik
```

Gambar 24 Pengujian 4 – Ekspresi Regular.
(Sumber: Penulis)

5. Pengujian 5 – Pencarian tidak ditemukan

Melakukan pengujian ketiga algoritma tetapi *pattern* yang dimasukkan dibuat sedemikian supaya pencarian tidak menemukan hasil.

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): kmp
Apakah case sensitive? (y/n): y
Masukkan pattern: asdjlfads

=====HASIL=====
indeks - [field]
Tidak ditemukan kecocokan pada record manapun

Menghabiskan 0.00925 detik
```

Gambar 25 Pengujian 5 – Algoritma KMP.
(Sumber: Penulis)

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): bm
Apakah case sensitive? (y/n): n
Masukkan pattern: asdjlfads

=====HASIL=====
indeks - [field]
Tidak ditemukan kecocokan pada record manapun

Menghabiskan 0.00359 detik
```

Gambar 26 Pengujian 5 – Algoritma BM.
(Sumber: Penulis)

```
Masukkan nama fail dan direktori jika ada: async.csv
Pilih algoritma yang akan digunakan (kmp/bm/regex): regex
Apakah case sensitive? (y/n): n
Masukkan pattern: asdjlfads

=====HASIL=====
indeks - [field]
Tidak ditemukan kecocokan pada record manapun

Menghabiskan 0.00132 detik
```

Gambar 27 Pengujian 5 – Ekspresi Regular.
(Sumber: Penulis)

IV. ANALISIS

Dari lima jenis pengujian yang sudah dilakukan. Semua algoritma, baik itu KMP, BM, ataupun Regex, dalam dua kondisi, yaitu *case sensitive* dan *case insensitive*, dapat melakukan pencarian *record* pada tabel secara tepat. Hasil pencarian dari seluruh jenis pengujian mengeluarkan hasil yang sama persis. Namun, terdapat perbedaan pada waktu eksekusi yang dihabiskan masing-masing algoritma.

Tabel 1 Waktu dari hasil pengujian dalam detik.

Algoritma	Pengujian 1,2,3		Pengujian 4	Pengujian 5	Rerata
	cs	ci			
KMP	0,00775	0,00914	0,0062	0,00925	0,008085
BM	0,00558	0,00395	0,00331	0,00359	0,0041075
Regex	0,00137	0,00112	0,00113	0,00132	0,001235

Pada tabel 1, dapat dilihat bahwa Algoritma Regex memiliki waktu eksekusi yang paling kecil daripada kedua algoritma lainnya. Hal ini menunjukkan bahwa Ekspresi Regular memiliki kompleksitas waktu yang lebih kecil dan efisien daripada kedua algoritma lainnya. Sehingga dapat disimpulkan Ekspresi Regular adalah algoritma yang paling cocok dalam pencarian *record* pada tabel.

V. KESIMPULAN

Algoritma pencocokan string adalah algoritma yang dapat menyelesaikan persoalan pencocokan string. Algoritma ini dapat menyelesaikan persoalan tersebut dengan berbagai algoritma, beberapa contohnya adalah *Brute Force*, KMP, BM, atau RegEx. Contoh dari persoalan tersebut adalah pencarian kata pada dokumen, analisis citra, pencarian di mesin pencari, ataupun pencocokan rantai DNA.

Salah satu pemanfaatan algoritma pencocokan string adalah seperti yang penulis implementasikan dalam makalah ini. Algoritma pencocokan string dapat menjadi alternatif solusi dalam mencari *record* pada suatu tabel. Dari pengujian yang sudah dilakukan juga didapat bahwa Ekspresi Regular merupakan algoritma yang paling cepat saat mencari *record* pada tabel daripada algoritma KMP dan BM.

Dengan adanya program yang dibuat oleh penulis ini, penulis berharap program ini dapat mempermudah seseorang dalam melakukan pencarian pada suatu tabel. Selain itu, penulis juga berharap program ini dapat dijadikan media untuk mempelajari algoritma pencocokan string.

TAUTAN PROGRAM

Program dapat diunduh langsung melalui tautan <https://github.com/raf555/CSV-RecordFinder-StringMatching>.

UCAPAN TERIMA KASIH

Penulis berterima kasih kepada Allah S.W.T, Tuhan Semesta Alam, karena limpahan Rahmat dan Nikmat-Nya penulis dapat menyelesaikan makalah ini dengan lancar dan tanpa hambatan. Penulis juga berterima kasih kepada jajaran dosen pengampu mata kuliah IF2211 Strategi Algoritma, khususnya kepada Dr. Ir. Rinaldi Munir, MT. karena sudah memberikan kuliah ini kepada penulis dengan sabar dan tulus. Sehingga penulis dapat mengerti dan memahami mata kuliah ini.

Penulis juga berterima kasih kepada orang tua penulis dan teman-teman penulis yang sudah mendukung penulis dalam menjalani perkuliahan ini, karenanya penulis bisa menyelesaikan makalah ini dengan lancar. Walaupun kondisi saat ini sedang sulit, tanpa mereka, penulis tidak akan bisa menyelesaikan perkuliahan serta makalah ini dengan baik.

DAFTAR PUSTAKA

- [1] Munir, Rinaldi. 2021. Pencocokan String 2021. Diakses pada tanggal 10 Mei 2021 dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>
- [2] N, Sora. 14 Desember 2014. Pengertian Field, Record, Table, File, Data Dan Basis Data Lengkap. Diakses pada tanggal 10 Mei 2021 dari <http://www.pengertianku.net/2014/12/pengertian-field-record-table-file-data-dan-basis-data-lengkap.html>
- [3] geeksforgeeks.org. 24 Maret 2021. KMP Algorithm for Pattern Searching. Diakses pada tanggal 10 Mei 2021 dari <https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/>
- [4] geeksforgeeks.org. 14 April 2021. Boyer Moore Algorithm for Pattern Searching. Diakses pada tanggal 10 Mei 2021 dari <https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching>
- [5] w3schools.com. 2021. Python RegExp. Diakses pada tanggal 10 Mei 2021 dari https://www.w3schools.com/python/python_regex.asp
- [6] tabel. 2021. KBBi Daring. Diakses pada tanggal 10 Mei 2021 dari <https://kbbi.kemdikbud.go.id/entri/tabel>

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.

Jakarta, 10 Mei 2021



Rezda Abdullah Fachrezzi
13519194