Penerapan Algoritma Regex dan Boyer moore pada Pengisian Formulir Online

Muhammad Fahmi Alamsyah - 13519077
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519077@std.stei.itb.ac.id

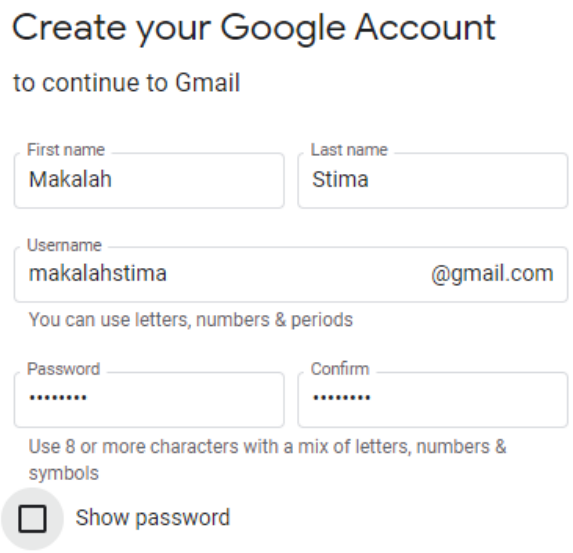
***Abstract*— Di era sekarang ini era 4.0 banyak sekali bermunculan teknologi-teknologi berbasis internet. Dengan berkembangnya aplikasi berbasis web dan android banyak sekali layanan-layanan yang tawarkan untuk membantu pekerjaan umat manusia. Layanan-layanan tersebut berkembang dan memanfaatkan data cloud sebagai media penyimpanannya. Oleh karena itu dibutuhkan pengisian formulir data diri untuk mendaftar layanan sebelum menggunakannya. String matching dengan regex dan Boyer-Moore digunakan untuk memvalidasi masukkan pengguna saat pengisian formulir data diri.**

Keywords—regex, Boyer-Moore, string matching, formulir

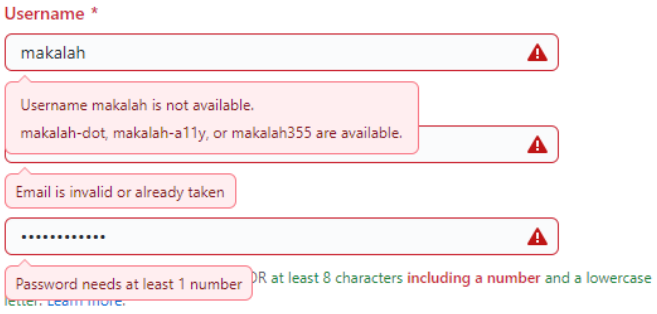
I. PENDAHULUAN

Di era sekarang ini era 4.0 banyak sekali bermunculan teknologi-teknologi berbasis internet. Dengan berkembangnya aplikasi berbasis web dan android banyak sekali layanan-layanan yang tawarkan untuk membantu pekerjaan umat manusia. Layanan-layanan tersebut berkembang dan memanfaatkan data cloud sebagai media penyimpanannya. Oleh karena itu, dibutuhkan pengisian formulir data diri untuk mendaftar layanan sebelum menggunakannya.

Pada saat mendaftar, seorang pendaftar biasanya akan diminta untuk mengisi informasi mengenai username, email, password, nama lengkap, tempat dan tanggal lahir, dan umur. Semua informasi data diri tersebut akan diolah dan dimasukkan ke dalam suatu database cloud. Namun, terkadang seorang pendaftar mengisikan formulir data diri tidak sesuai dengan format-format tertentu seperti dalam pengisian email tidak menggunakan keyword "@gmail.com". Selain itu juga, terkadang dari pihak layanan menambahkan peraturan-peraturan tertentu untuk menjaga keamanan data pengguna seperti peraturan pengisian password yang di wajibkan minimum 7 karakter dan terdapat huruf alphabet dan angka di dalamnya. Selanjutnya pemikiran muncul, bagaimana suatu sistem komputer dapat mendeteksi input masukkan dengan segala peraturan-peraturan yang telah ada. Jawabannya adalah penggunaan teknologi Regex dan algoritma Boyer-Moore sebagai metode pencocokan otomatis antara masukkan pengguna dan peraturan yang telah dibuat. Teknologi Regex dan algoritma Boyer-Moore akan mencocokan karakter-karakter masukkan input pengguna dengan peraturan-peraturan pengisian formulir yang telah dibuat. Tentunya kemunculan teknologi dan algoritma ini memberikan dampak yang sangat besar terutama pada perkembangan bidang IT.



Gambar 1. Teknologi Regex dan algoritma Boyer-Moore pada pengisian formulir online

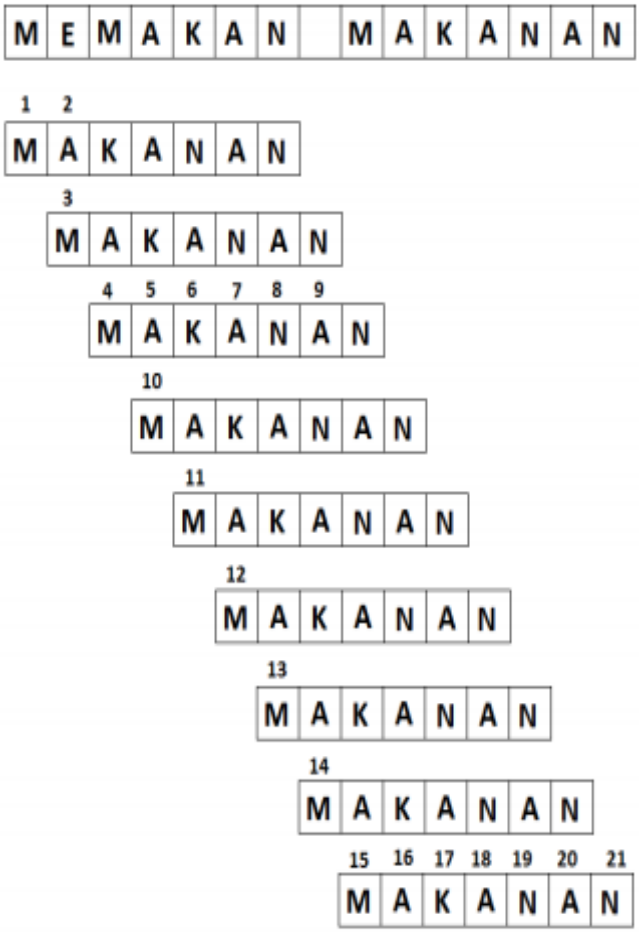


Gambar 2. Regex dan Boyer-Moore mendeteksi kesalahan masukkan tidak sesuai dengan rules formulir

.

II. DASAR TEORI

A. Algoritma String Matching dengan Brute Force



Gambar 3. String Matching dengan Algoritma Brute Force

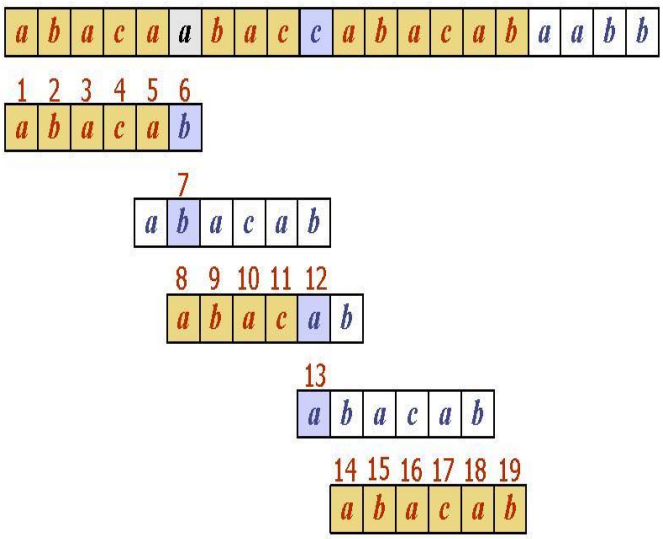
Pada perkembangannya algoritma brute force adalah algoritma pertama yang mendasari perkembangan teknologi string matching. Algoritma ini akan mencocokan satu per satu karakter pada pattern akan dibandingkan dengan teks. Karakter dibandingkan dari kiri ke kanan. Jika hasil perbandingan karakter sama, maka indeks pattern akan bertambah satu dan indeks juga bertambah satu. Aktivitas ini akan terus diiterasi selama ditemukan kecocokan hingga indeks pattern mencapai indeks terakhir. Ketika indeks pattern mencapai indeks terakhir dan menghasilkan kecocokan dengan karakter pada teks maka dapat disimpulkan bahwa terdapat pattern tersebut pada teks yang dicari. Jika pada perbandingan karakter pada teks dan pattern ditemukan ketidakcocokan, maka indeks teks akan bertambah satu dan indeks pattern akan kembali lagi ke awal. Aktivitas ini akan terus diiterasi selama ditemukan ketidakcocokan dengan antara karakter pada pattern dan karakter pada teks. Jika perbandingan telah mencapai ujung indeks teks dan ujung indeks pattern serta ditemukan ketidakcocokan karakter maka dapat dinyatakan pattern tersebut tidak ditemukan pada teks.

Tentunya sebuah algoritma pasti memiliki kasus terbaik dan kasus terburuk. Kasus terbaik terjadi ketika ketidakcocokan karakter selalu terjadi pada indeks pertama pada pattern, misalnya ketika teksnya adalah “qwertyuiopaaa” dan patternnya adalah “aaa”. Pada kasus ini perbandingan maksimalnya adalah n kali sehingga kompleksitasnya adalah O(n). Kasus terburuk pada string matching terjadi ketika ketidakcocokan karakter selalu terjadi pada ujung pattern, misalnya ketika teks adalah "aaaaaaaab" dan patternya "aab". Jika m adalah jumlah karakter pada pattern dan n adalah jumlah karakter pada teks maka untuk kasus terburuk jumlah perbandingan yang dilakukan adalah m(n-m+1) sehingga kompleksitasnya adalah O(mn)

B. Algoritma String Matching Knuth-Morris-Pratt

Algoritma KMP adalah algoritma yang ditemukan oleh Donald E. Knuth. Algoritma ini tercipta karena algoritma brute force yang kurang efektif dimana membandingkan seluruh karakter pada pattern. Oleh karena itu, algoritma ini hadir dengan sedikit modifikasi dari algoritma brute force dengan cara menggeser patternnya sedemikian rupa sehingga jumlah perbandingan yang dilakukan menjadi lebih sedikit.

Pada KMP dikenal fungsi pinggiran atau border function, dinotasikan dengan b(k), dimana k adalah nomor indeks pada pattern. Fungsi b(k) akan mengembalikan panjang kesamaan antara prefiks dan sufiks pada sebuah pattern.Untuk k=4, misal ada sebuah pattern "abcabc" maka prefiksnya adalah "a", "ab", "abc", "abca", sedangkan suffiksnya adalah "b", "ab", "cab", "bcab". Pada suffiks dan prefiks ditemukan kesamaan pada "ab". Maka fungsi b(4) akan mengembalikan nilai panjang "ab", yaitu dua



x	0	1	2	3	4	5
P[x]	a	b	a	c	a	b
f(x)	0	0	1	0	1	2

. Gambar 4. String Matching dengan Algoritma KMP

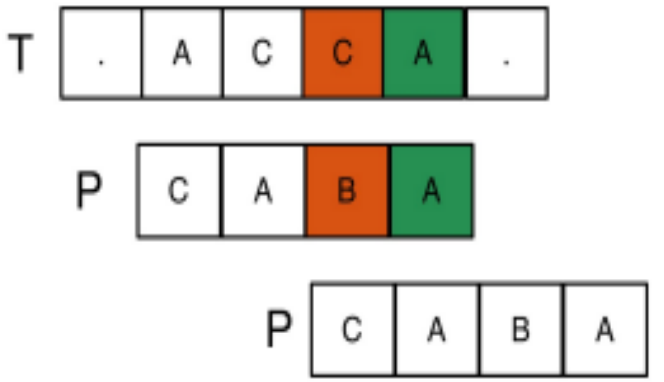
Kompleksitas waktu yang dibutuhkan pada KMP untuk menghitung fungsi pinggiran adalah O(m), sedangkan komplesitas yang dibutuhkan untuk pencarian string adalah O(n) sehingga kompleksitas total dari algoritma KMP adalah O(m+n). Jika kita bandingkan dengan algoritma bruteforce maka algoritma KMP ini sangat cepat.

Algoritma KMP sendiri menawarkan keuntungan dan kekurangan tersendiri, Dengan algoritmanya maka dengan KMP bisa dipastikan tidak ada pencocokan yang berulang pada pattern yang sudah dibandingkan dan sama sebelumnya, sedangkan kekurangannya adalah KMP kemangkusannya akan berkurang apabila pattern yang dipakai mempunyai fungsi pinggiran yang sedikit atau bahkan nol.

C. **Algoritma String Matching Boyer-Moore**

Algoritma Boyer-Moore agak berbeda dari kedua algoritma sebelumnya. Pada algoritma ini, string tidak dicocokkan dari depan ke belakang tetapi mundur dari belakang ke depan.

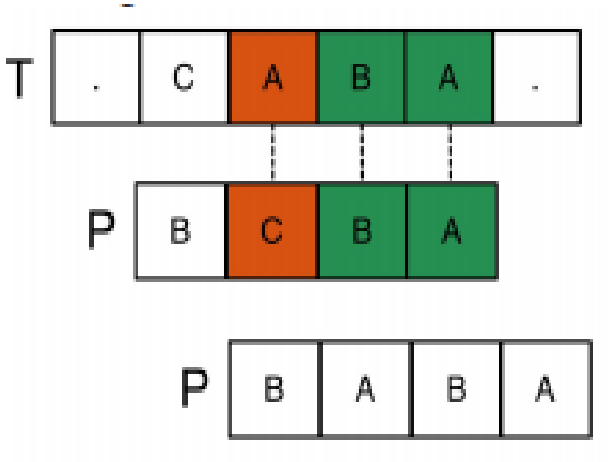
Boyer-Moore membagi-bagi kasus yang ada menjadi 3 bagian. Untuk kasus yang pertama, ketika terjadi ketidakcocokan antara karakter di teks dan di pattern maka pattern akan digeser ke kanan hingga ditemukan karakter yang sama dengan karakter pada teks.



Gambar 5. Kasus 1 pada Boyer-Moore

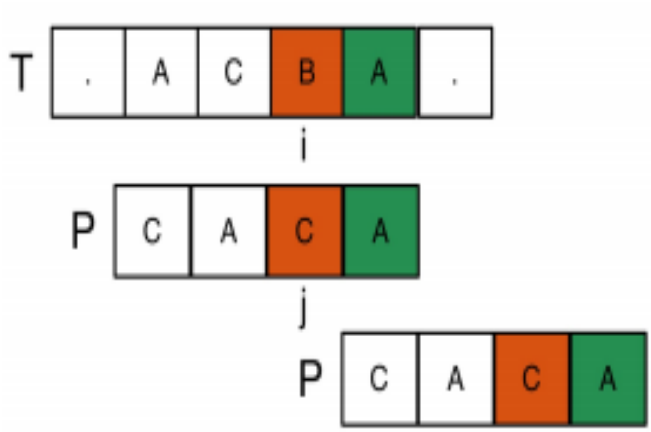
Seperti dilihat pada gambar 5, pertama-tama karakter dicocokkan mulai dari indeks terakhir. Ketika karakternya cocok, maka pencocokan berlanjut ke indeks sebelumnya. Ketika "C" dibandingkan dengan "B" dan tidak cocok maka pattern akan digeser ke kanan hingga "C" di pattern sejajar dengan "C" di teks.

Kasus 2 Boyer-Moore, misal i adalah indeks pada teks dan j adalah indeks pada pattern, terjadi ketika T[i] dan P[j] tidak cocok dan tidak ada karakter sebelum indeks ke-j pada pattern yang sama dengan indeks ke i maka pattern akan digeser sebesar 1 blok ke kanan.



Gambar 6. Kasus 2 pada Boyer-Moore

Kasus 3 Boyer-Moore terjadi ketika ditemukan ketidakcocokan antarar karakte T[i] dan P[j] serta tidak ditemukan karakter pada pattern yang sama dengan karakter pada T[i]. Jika hal ini terjadi maka pattern akan digeser ke kanan sehingga P[1] sejajar dengan T[i+1].



Gambar 7. Kasus 3 pada Boyer-Moore

Pada kasus terburuk, yaitu ketika disepanjang teks seluruh karakternya mirip dengan pattern, kompleksitas algoritma Boyer-Moore adalah O(nm). Contohnya ketika teksnya "bbbbbbbbbb" dan patternnya "abb". Algoritma Boyer-Moore akan semakin baik performansinya ketika alfabetnya semakin bervariasi. Hal ini menyebabkan algoritma ini cepat ketika digunakan untuk mencari teks pada bahasa sehari-hari namun akan berkurang performansinya ketika digunakan untuk mencari pada string biner.

D. **Regular Expression (Regex)**

Regular expression (Regex) merupakan sekumpulan notasi dan karakter yang digunakan untuk mendeskripsikan suatu pola pada pencarian berbasis huruf. Dengan Regex dimungkinkan untuk mengenali suatu string yang mempunya karakteristik dan pola tertentu, seperti email, tanggal, nomor kartu kredit, dll. Misal dengan notasi Regex p.+ing maka akan

menyaring semua kata-kata kecuali kata yang diawali huruf "p" dan mempunyai akhiran ing, seperti playing, praying, pulling, dll.

.	Any character except newline.
\.	A period (and so on for *, \(, \\, etc.)
^	The start of the string.
$	The end of the string.
\d,\w,\s	A digit, word character [A-Za-z0-9_], or whitespace.
\D,\W,\S	Anything except a digit, word character, or whitespace.
[abc]	Character a, b, or c.
[a-z]	a through z.
[^abc]	Any character except a, b, or c.
aa|bb	Either aa or bb.
?	Zero or one of the preceding element.
*	Zero or more of the preceding element.
+	One or more of the preceding element.
{*n*}	Exactly *n* of the preceding element.
{*n*,}	*n* or more of the preceding element.
{*m*,*n*}	Between *m* and *n* of the preceding element.
??,*?,+?, {*n*}?, etc.	Same as above, but as few as possible.
(*expr*)	Capture *expr* for use with \1, etc.
(?:*expr*)	Non-capturing group.
(?=*expr*)	Followed by *expr*.
(?!*expr*)	Not followed by *expr*.

Near-complete reference

Gambar 8. Notasi Regex

Notasi Regex dapat dikombinasikan sedemikian kompleksnya sehingga dapat menemukan kata yang mempunya pola-pola cukup rumit. Adapun contohnya:

```
^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-  ]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$
```

akan menampilkan error atau warning ketika string yang dimasukkan tidak memiliki format seperti alamat email, misal stima@yahoo.com

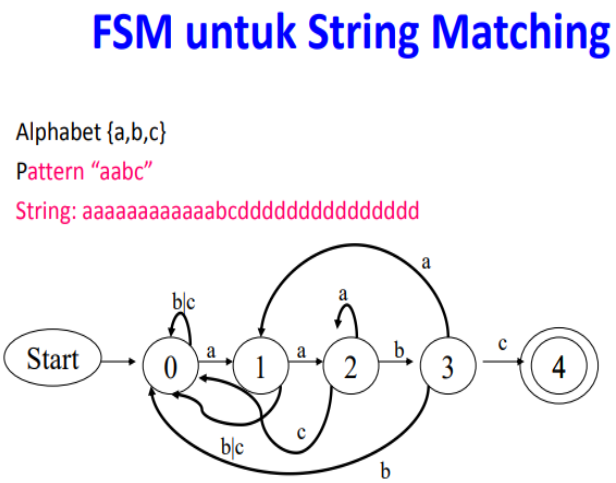
```
lulusan .*[Ii]nformatika [Ii][Tt][Bb]
```

```
Sebagai seorang lulusan Teknik Informatika ITB, saya mal
bedanya dengan tahun 1990an?

Pernah nemu ada yg ngaku lulusan Informatika ITB. Gw tan
ngelampirin sertifikat training HMIF ITB.

Siapa kak ? Siapa ? "@niaputrinia: Umurnya 26 tahun, lul
fotografer dan kontributornya National"
```

Gambar 9. Contoh penggunaan Regex

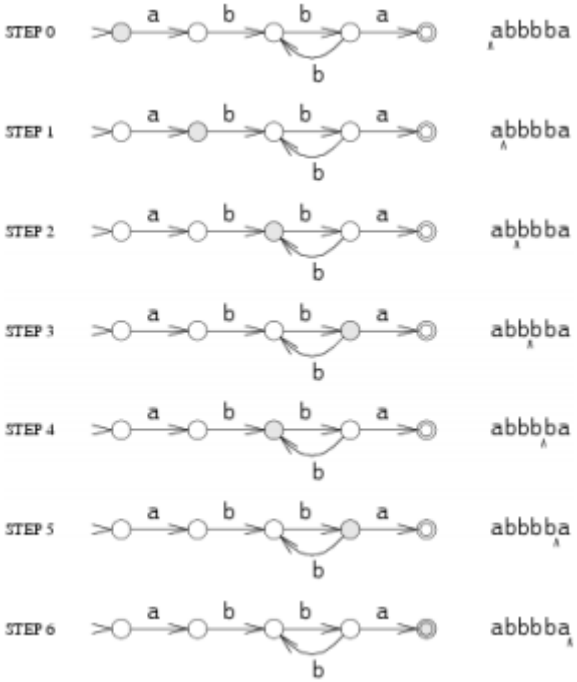
E. String Matching dengan Finite State Machines (FSM)

Finite state machines juga merupakan konsep dalam pengembangan string matching. Konsep dari Finite State Machines adalah dengan gambar-gambar atau diagram yang berisi state-state dan suatu transisinya.



Gambar 10. String Matching pada Finite State Machines

Sebuah finite automata selalu berada pada salah satu state yang direpresentasikan oleh lingkaran pada diagram dengan label yang menunjukkan state yang berbeda dengan state lainnya. State awal digambarkan dengan panah masuk yang tidak berasal dari state lainnya, sementara state akhir digambarkan dengan lingkaran yang menutupi lingkaran state (dua lingkaran). Berikut ilustrasi perpindahan state pada sebuah finite automata:



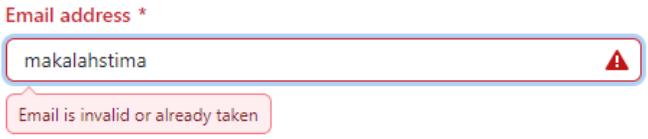
Gambar 11. Pencocokan string pada FSM

III. PENERAPAN

A. Sistem Validasi dengan Regex dan Boyer-Moore

Sistem validasi formulir pada sebuah website baiknya terjadi secara langsung menggunakan AJAX. AJAX adalah sebuah singkatan dari Asynchronous Javascript and XML dan mengacu pada sekumpulan teknis pengembangan web (web development) yang memungkinkan aplikasi web untuk bekerja secara asynchronous (tidak langsung) – memproses setiap request (permintaan) yang datang ke server di sisi background. Yang dimaksud langsung disini adalah ketika user selesai mengisi suatu field maka akan muncul pesan apakah data yang dimasukkan sudah valid atau belum. Tentunya hal ini akan sangat membantu user terlebih apabila ada typo sehingga user tidak perlu mereload page untuk mengetahu validitas formulirnya.

Lazimnya ketika memvalidasi formulir, ada 2 hal utama yang divalidasi. Pertama adalah mengecek apakah data yang dimasukkan sudah sesuai pola/bentuk yang diharapkan. Yang kedua adalah mengecek apakah data yang dimasukkan masih layak pakai.



Gambar 12. Kasus Pertama Tidak Berhasil

Gambar 13. Kasus Pertama Berhasil

Kasus pertama, data dikatakan valid ketika data yang dimasukkan adalah data yang diminta. Misal, pada textbox email address yang diisi harus benar-benar berbentuk sebuah email yang berbentuk character. Disinilah peran Regex dan Regex seharusnya dapat mengenali apakah data yang dimasukkan sudah valid.

Untuk meninjau kasus pertama akan diambil contoh pada validasi email dan password pada github.com. Ketika " makalahstima" dimasukkan, pada kolom email akan muncul error. Terlihat alamat email yang dimasukkan tidak memiliki karakter "@" . Adapun Regex yang digunakan kurang lebih seperti berikut:

```
^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0- 9-.]+$
```

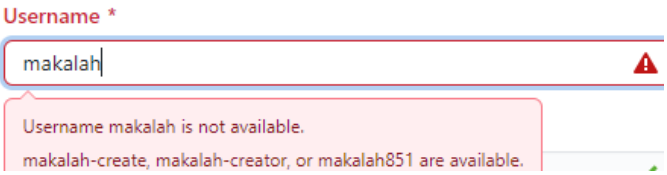
Penjelasan singkat mengenai Regex diatas sbb:

- ^ menandakan awal string - [a-zA-Z0-9_.+-] semua huruf termasuk kapital dan angka serta karakter "_", ".", "+", "-".

- +@ menyatakan minimal terdapat satu karakter atau lebih sebelum sebuah karakter "@".

- [a-zA-Z0-9-] semua huruf termasuk kapita dan angka serta karakter "-". - +\. menyatakan minimal terdapat satu karakter atau lebih sebelum karakter ".".

- $ menandakan akhir dari sebuah string



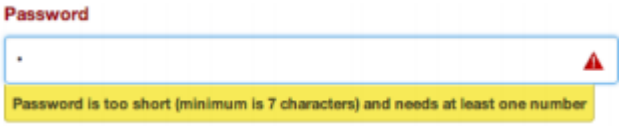
Gambar 14. Kasus Kedua Tidak Berhasil

Gambar 15. Kasus Kedua Berhasil

Kasus kedua, data dikatakan valid ketika data yang dimasukkan masih tersedia. Hal ini sering terjadi ketika ingin mendaftar pada suatu website. Biasanya kita diminta untuk mengisi suatu username dan username tersebut haruslah unik. Ketika username yang dimasukkan tidak unik maka masukkan tersebut dianggap tidak valid. Untuk mengatasi kasus kedua ini dapat dimanfaatkan salah satu dari algoritma string matching yang ada.

Untuk meninjau kasus kedua akan diambil contoh pada github.com Ketika ingin mendaftar di github, pendaftar wajib mengisi kolom username. Ketika username tidak unik maka akan muncul pesan error. Disini algoritma string matching digunakan untuk mengecek username yang dimasukkan dan dicocokkan dengan daftar username yang ada di database. String matching yang digunakan disini sedikit berbeda meski konsep dasarnya sama saja. Algoritma yang disarankan untuk digunakan adalah Boyer-Moore mengingat kecepatannya dibandingkan dengan bruteforce dan pertimbangan bahwa variasi alfabet pada username sangat tinggi. String matching akan menghasilkan true jika dan hanya jika panjang username yang dimasukkan sama dengan panjang username di database dan semua karakternya exact match.

Misal username yang dimasukkan "makalah" dan didatabase terdapat username terdaftar "makalah", maka string matching akan mengembalikan true saat diperiksa. Sehingga username masukan tidak valid. Jadi disini yang menjadi pattern adalah username yang dimasukkan dan yang menjadi teks adalah daftar username yang ada di database. Namun, ketika username dimasukkan “makalahstima” Boyer-Moore akan mengembalikkan nilai false, dank arena false artinya username tersebut belum terdaftar pada database sehingga username valid.



Gambar 16. String Matching pada Pengisian Password

Untuk memvalidasi password dengan syarat minimal 7 karakter dan mempunyai paling sedikit satu angka dapat digunakan Regex: ^(?=.*\d).{7,}$ Penjelasan Regex diatas sbb:

- (?=.*\d). menerima 0 atau lebih karakter yang diikuti oleh sebuah angka.
- {7,}$ menerima jika string tersebut mempunyai panjang minimal 7 karakter.

Dibalik kemudahan yang ditawarkan regular expression dalam mengenali suatu pola, regex juga mempunyai kekurangan. Meski regex dapat dibuat sedemikian kompleksnya untuk mengenali suatu pola, ada beberapa kasus dimana regex masih belum mampu untuk mengenali seratus persen kecocokan suatu pola. Hal ini dapat dilihat pada kasus pengenalan alamat email. Pada regex:

```
^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0- 9-.]+$
```

akan memblok semua string yang bukan merupakan alamat email. Akan tetapi masih ada beberapa kasus yang bocor dengan regex diatas.

Misalnya saja, “makalah..stima@gmail.com” Terlihat pada email tsb terdapat dua buah titik yang berturut-turut, yang mana seharusnya tidak diperbolehkan pada sebuah alamat email. Dengan menggunakan notasi regex diatas, alamat email tsb akan dianggap valid. Agar benar-benar bisa memvalidasi semua alamat email maka notasi regex diatas harus didefinisi ulang, yang mana akan membutuhkan berpuluh-puluh baris kode. Hal ini tentu menjadi tidak efisien. Oleh karena itu tidak setiap kasus dapat ditangani dengan regex, diperlukan juga algoritma KMP atau Boyer-Moore untuk melakukan pencocokan string.

B. Implementasi Kode dalam Bahasa Python

```
import re
def badCharHeuristic(string, size):
    '''
    The preprocessing function for
    Boyer Moore's bad character heuristic
    '''

    # Initialize all occurrence as -1
    badChar = [-1]*256

    # Fill the actual value of last occurrence
    for i in range(size):
        badChar[ord(string[i])] = i;

    # retun initialized list
    return badChar
```

```
def search(txt, pat):
    '''
    A pattern searching function that uses Bad
Character
    Heuristic of Boyer Moore Algorithm
    '''
    m = len(pat)
    n = len(txt)

    # create the bad character list by calling
    # the preprocessing function badCharHeuristic()
    # for given pattern
    badChar = badCharHeuristic(pat, m)

    # s is shift of the pattern with respect to text
    s = 0
    while(s <= n-m):
        j = m-1

        # Keep reducing index j of pattern while
        # characters of pattern and text are matching
        # at this shift s
        while j>=0 and pat[j] == txt[s+j]:
            j -= 1

        # If the pattern is present at current shift,
        # then index j will become -1 after the above
loop
        if j<0:
            return True

            '''
                Shift the pattern so that the next
character in text
                    aligns with the last occurrence of it in
pattern.
                The condition s+m < n is necessary for
the case when
                    pattern occurs at the end of text
                '''
            s += (m-badChar[ord(txt[s+m])] if s+m<n
else 1)
        else:
            '''
                Shift the pattern so that the bad character
in text
                aligns with the last occurrence of it in
pattern. The
                max function is used to make sure that we
get a positive
                shift. We may get a negative shift if the
last occurrence
                of bad character in pattern is on the right
side of the
                current character.
            '''
            s += max(1, j-badChar[ord(txt[s+j])])

    return False
```

```
def password(text):
    match = re. match("^(?=.*\d).{7,}$", text)
    boolean = bool(match)
    return boolean


# Driver program to test above function
def main():
    booluser = False
    boolpass = False
    database = "makalah"
    while (True):
        username = input("Masukkan username : ")
        katasandi = input("Masukkan password : ")

        booluser = search(database, username)
        boolpass = password(katasandi)

        if (not booluser):
            if (boolpass):
                print("Akun berhasil terverifikasi")
                return
            else:
                print("Kata sandi salah")
        else :
            print("Username telah dipakai")


if __name__ == '__main__':
    main()
```

Output Program

```
Masukkan username : makalah
Masukkan password : stima123
Username telah dipakai

Masukkan username : makalahstima
Masukkan password : stima
Kata sandi salah

Masukkan username : makalahstima
Masukkan password : stima123
Akun berhasil terverifikasi
```

Pada program diatas terdapat 3 fungsi dan 1 main program. Fungsi badCharHeuristic(string,size), search(txt, pat), dan password(text). Fungsi badCharHeuristic(string,size) digunakan sebagai fungsi antara yang mendukung fungsi search(txt,pat). Fungsi search(txt, pat) adalah fungsi yang akan mengecek username dan bertindak sebagai algoritma Boyer-Moore, fungsi ini akan mengembalikan nilai True jika pattern ditemukkan pada txt, dan mengembalikan False jika pattern tidak ditemukkan pada txt. Pattern disini adalah masukkan username dari pengguna dan txt disini adalah database. Fungsi password(text) adalah fungsi yang digunakan untuk Smemvalidasi masukkan password pengguna, fungsi ini menggunakan algoritma regex di dalamnya. Fungsi akan mereturn True jika masukkan password pengguna mengandung minimal 7 karakter huruf dan angka, tetapi fungsi akan mereturn False jika masukkan password pengguna tidak mengandung minimal 7 karakter huruf dan angka.

Pada program driver, fungsi search dan password akan di panggil didalam kalang perulangan while(True) dan jika username menghasilkan True atau artinya username telah terdapat pada database maka akan dikeluarkan output “Username telah dipakai”, dan jika password yang digunakan tidak mengandung minimal 7 karakter huruf dan angka maka akan dikeluarkan output “Kata sandi salah”.

IV. KESIMPULAN DAN SARAN

String matching dan regular expression dapat diaplikasikan untuk mendeteksi kesalahan pada saat pengisian formulir. Dengan memanfaatkan kedua teknik diatas, maka dapat membantu pengguna dalam mengetahui error yang terjadi seketika mungkin. Meski begitu, Regex belum mampu sepenuhnya untuk benar-benar mengecek validitas suatu masukan hingga 100% pada kasus tertentu, hal ini dikarenakan batasanbatasan yang dimiliki Regex. Kalaupun hal tersebut dapat dilakukan maka akan dibutuhkan notasi Regex yang sangat panjang dan kompleks untuk dimengerti. Oleh sebab itu, dibutuhkan juga algoritma Boyer-Moore, KMP, ataupun brute force sebagai alternative dalam melakukan string matching.

PRANALA VIDEO YOUTUBE

Berikut merupakan pranala video yang berisi tentang penjelasan materi makalah ini.

https://www.youtube.com/watch?v=K37PpY2enMA

UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan terima kasih yang sebesar-besarnya kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmatnya, penulis bisa menyelesaikan makalah ini sesuai waktu yang telah diberikan. Tidak lupa juga penulis mengucapkan terima kasih kepada kedua orang tua yang selalu mendukung dan mendidik penulis sehingga bisa melanjutkan pendidikan di Institut Teknologi Bandung. Tentu juga penulis mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidewi, ST., M.Sc yang telah memberikan begitu banyak ilmu kepada penulis selama 1 semester ini. Terakhir penulis juga mengucapkan terima kasih kepada teman-teman yang mendukung saat pembuatan makalah ini. Besar harapan saya agar makalah ini bisa berguna bagi banyak orang.

REFERENCES

[1]Munir, Rinaldi. Diktat Kuliah IF2251 Strategi Algoritmik, Institut Teknologi Bandung. 2007.

[2]https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/ Diakses 10 Mei 2021 12:20

[3]https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf Diakses 10 Mei 2021 11.30
[4]https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014-genap/Makalah2014/MakalahIF2211-2014-006.pdf Diakses 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.

Kebumen, 10 Mei 2021
Muhammad Fahmi Alamsyah – 13519077