Penggunaan Regex dan String Matching Dalam Menyaring Informasi Pendaftaran User

Fahrian Afdholi - 13521031
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521031@std.stei.itb.ac.id

***Abstract*—Regex, atau singkatan dari "Regular Expression," adalah algoritma yang digunakan untuk mencocokkan dan memanipulasi pola teks. Dalam bahasa yang lebih mudah dipahami, bisa kita analogikan sebagai puzzle kata-kata yang membantu kita mencari dan memanipulasi kata-kata yang sesuai dengan pola tertentu. Misalnya, jika kita ingin mencari semua kata yang dimulai dengan huruf "A" dalam sebuah teks, kita dapat menggunakan algoritma regex dengan pola "/^A\w*/". Di sini, simbol-simbol seperti "^" dan "\w" memiliki arti khusus dalam regex dan digunakan untuk menentukan pola yang diinginkan. Algoritma regex dapat digunakan dalam berbagai bahasa pemrograman dan aplikasi, dan membantu kita untuk mencari, mengganti, atau memvalidasi teks dengan efisien berdasarkan pola yang ditentukan.**

Keywords—email, registrasi, regex

I. PENDAHULUAN

Regex, atau singkatan dari "Regular Expression," adalah algoritma yang digunakan untuk mencocokkan dan memanipulasi pola teks. Dalam bahasa yang lebih mudah dipahami, bisa kita analogikan sebagai puzzle kata-kata yang membantu kita mencari dan memanipulasi kata-kata yang sesuai dengan pola tertentu. Misalnya, jika kita ingin mencari semua kata yang dimulai dengan huruf "A" dalam sebuah teks, kita dapat menggunakan algoritma regex dengan pola "/^A\w*/". Di sini, simbol-simbol seperti "^" dan "\w" memiliki arti khusus dalam regex dan digunakan untuk menentukan pola yang diinginkan. Algoritma regex dapat digunakan dalam berbagai bahasa pemrograman dan aplikasi, dan membantu kita untuk mencari, mengganti, atau memvalidasi teks dengan efisien berdasarkan pola yang ditentukan.

Dalam melakukan pendaftaran sebuah user di sebuah aplikasi dalam beberapa kasus kita perlu untuk menyaring terlebih dahulu apa saja informasi yang wajib untuk dipenuhi oleh user untuk mendaftar pada aplikasi yang ingin dia daftar. Tetapi, beberapa orang kebingungan bagaimana caranya untuk menyaring data data user yang harus dipenuhi. Misalkan ada sebuah perusahaan yang memiliki sebuah aplikasi tetapi si perusahaan itu hanya ingin yang mendaftar aplikasi tersebut hanya orang orang yang memiliki email perusahaan saja atau si perusahaan ingin memberikan akses kepada email email tertentu saja dan bisa juga jika si perusahaan ingin usernya hanya memiliki nomor handphone yang hanya berasal dari Indonesia saja.

Permasalahan tersebut dapat diselesaikan dengan regex seperti contohnya jika perusahaan ingin yang mendaftar hanya karyawannya yang memiliki email perusahaan saja bisa dengan menggunakan regex seperti contohnya pada kode di golang “(.+)@perusahaan.co.id” atau jika pada string matcing kita hanya perlu menggunakan “@perusahaan.co.id” lalu algoritma akan menampilkan indeks pertama yang match dengan email tersebut.

II. DASAR TEORI

A. Algoritma String Matching dan Regex

Algoritma String Matching, juga dikenal sebagai pencocokan string, adalah algoritma yang digunakan untuk mencari kemunculan sebuah pola atau string tertentu dalam sebuah teks atau string yang lebih panjang. Tujuan utama dari algoritma ini adalah untuk menemukan posisi atau indeks dimana pola tersebut cocok dengan teks secara tepat atau dengan beberapa modifikasi tertentu.

Algoritma string matching memiliki berbagai macam pendekatan, dan pilihan algoritma yang tepat tergantung pada kasus penggunaan spesifik dan kompleksitas yang diinginkan. Salah satu metode paling sederhana dalam algoritma string matching adalah Brute-Force atau Naive Algorithm. Metode ini bekerja dengan membandingkan pola secara berurutan dengan semua kemungkinan substring dalam teks. Jika pola cocok dengan substring, maka pola tersebut dianggap ditemukan. Namun, metode ini memiliki kompleksitas waktu yang tinggi, terutama jika pola dan teks memiliki ukuran yang besar.

Untuk mengatasi keterbatasan Brute-Force Algorithm, beberapa algoritma string matching yang lebih efisien telah dikembangkan. Salah satunya adalah algoritma Knuth-Morris-Pratt (KMP). Algoritma KMP menggunakan pendekatan yang cerdas dengan memanfaatkan informasi yang sudah diketahui mengenai pola itu sendiri. Ia menghindari pencocokan ulang dengan memanfaatkan fungsi prefiks dan sufiks dari pola. Algoritma KMP mampu meningkatkan efisiensi dalam mencari pola dalam teks dengan mengurangi jumlah perbandingan yang perlu dilakukan.

Selain KMP, terdapat juga algoritma Boyer-Moore yang populer dalam algoritma string matching. Algoritma Boyer-Moore bekerja dengan memanfaatkan informasi dari kemunculan karakter yang tidak cocok antara pola dan teks. Ia memanfaatkan tabel skor heuristik, yaitu tabel yang memberikan informasi tentang pergeseran pola yang mungkin dilakukan berdasarkan karakter yang tidak cocok. Dengan strategi pergeseran yang cerdas, algoritma Boyer-Moore mampu mempercepat pencarian pola dalam teks.

Algoritma string matching memiliki berbagai aplikasi dalam dunia nyata. Contoh penerapannya termasuk pencarian teks dalam editor teks, pencarian kata kunci dalam mesin pencari, validasi input pengguna, analisis log, dan masih banyak lagi. Kecepatan dan efisiensi algoritma string matching menjadi faktor penting dalam aplikasi-aplikasi ini untuk memberikan hasil yang akurat dan responsif.

Regex (Regular Expression) adalah sebuah algoritma yang digunakan dalam pemrograman dan pengolahan teks untuk mencari, memanipulasi, dan memvalidasi pola tertentu dalam sebuah teks. Pada dasarnya, regex adalah sebuah urutan karakter khusus yang membentuk sebuah pola, dan algoritma regex menggunakan pola ini untuk mencocokkan dan memanipulasi teks.

Pola dalam regex terdiri dari karakter-karakter literal (seperti huruf dan angka) yang cocok secara tepat dengan karakter yang sama dalam teks, serta karakter-karakter khusus yang memberikan makna dan fungsi tambahan. Sebagai contoh, karakter khusus "*" dapat digunakan untuk mencocokkan nol atau lebih kemunculan karakter sebelumnya, sedangkan karakter "+" digunakan untuk mencocokkan satu atau lebih kemunculan karakter sebelumnya. Dengan menggunakan karakter-karakter khusus ini, kita dapat membuat pola yang lebih fleksibel dan komprehensif.

Algoritma regex berfungsi dengan cara mencocokkan pola yang diberikan dengan teks masukan. Jika pola cocok dengan sebagian atau keseluruhan teks, maka algoritma regex akan memberikan hasil yang sesuai. Algoritma ini melakukan pencarian pola secara iteratif dalam teks, mencocokkan karakter demi karakter, dan mencoba untuk menemukan pola yang cocok secara keseluruhan atau sebagian. Ketika ada kemungkinan cocok, algoritma akan melanjutkan pencarian hingga menemukan semua kemungkinan cocok atau mencapai akhir teks.

Penggunaan algoritma regex sangat luas. Dalam pemrograman, algoritma ini digunakan untuk pencarian dan manipulasi teks dalam berbagai bahasa pemrograman seperti Python, JavaScript, dan Perl. Regex juga diterapkan dalam aplikasi seperti pengolahan teks lanjutan, validasi data, analisis log, pengambilan informasi dari dokumen, dan masih banyak lagi.

Namun, meskipun algoritma regex memiliki banyak manfaat, perlu diingat bahwa penggunaannya yang tidak benar atau tidak hati-hati dapat menyebabkan kesalahan atau kinerja yang buruk. Pola yang kompleks atau berulang-ulang dalam teks yang panjang dapat menghabiskan banyak waktu pemrosesan dan mengakibatkan kinerja yang lambat.

Untuk mengatasi keterbatasan ini, beberapa implementasi regex telah mengoptimalkan algoritmanya. Misalnya, ada metode yang disebut "compiling" yang mengubah pola regex menjadi struktur data yang lebih efisien untuk pencocokan berulang. Selain itu, beberapa pustaka dan bahasa pemrograman juga menyediakan opsi pengaturan untuk mengoptimalkan kinerja regex.

Dalam kesimpulannya, algoritma regex adalah alat yang sangat berguna dalam pemrograman dan pengolahan teks. Dengan menggunakan pola yang sesuai, kita dapat mencocokkan, memanipulasi, dan memvalidasi teks dengan cara yang lebih efisien dan terstruktur. Pemahaman tentang cara kerja algoritma regex dan penggunaan yang tepat sangat penting untuk mengoptimalkan kinerja dan memanfaatkannya sepenuhnya.

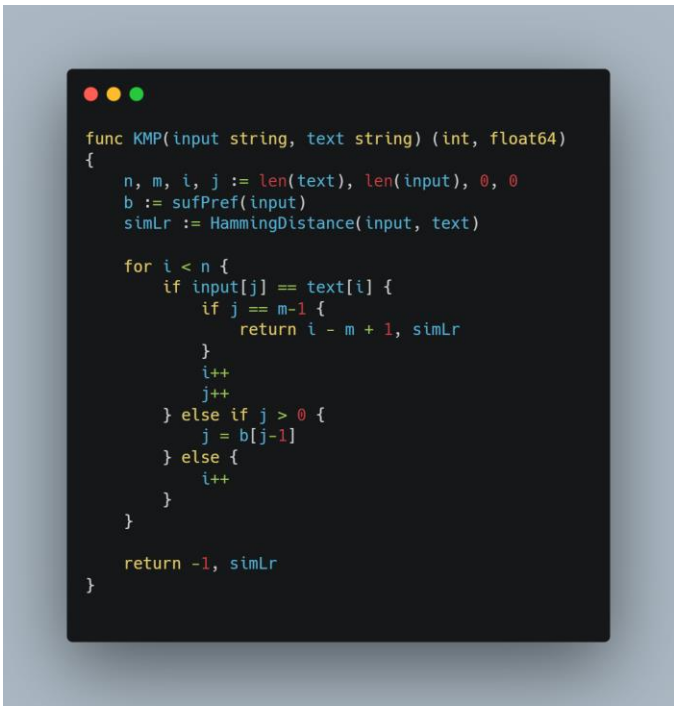
B. FUNGSI ALGORITMA STRING MATCHING DAN REGEX

Pada algoritma string matching digunakan untuk membandingkan apakah sebuah teks/kalimat cocok dengan teks/kaliat yang ingin dibandingkan jika cocok pada indeks ke berapa mereka cocok dan jika tidak cocok maka algoritma akan mengembalikan sebuah value ketidakcocokan.

Pada regex hampir mirip seperti string matching karena regex sediri merupakan bagian dari string matching tetapi regex dalam pengecekannya akan mengahasilkan sebuah kalimat atau teks atau huruf dengan menggunakan kalinat yang dibandingkan pada regex.

Ini merupakan salah satu contoh algoritma string matching KMP, BM, dan salah satu algoritma regex.

a. KMP



```
func KMP(input string, text string) (int, float64)
{
    n, m, i, j := len(text), len(input), 0, 0
    b := sufPref(input)
    simLr := HammingDistance(input, text)

    for i < n {
        if input[j] == text[i] {
            if j == m-1 {
                return i - m + 1, simLr
            }
            i++
            j++
        } else if j > 0 {
            j = b[j-1]
        } else {
            i++
        }
    }

    return -1, simLr
}
```

b. BM

```go
func BM(input string, text string) (int, float64) {

    bl := buildLast(input)
    simLr := HammingDistance(input, text)
    n, m := len(text), len(input)
    i := m - 1
    j := m - 1

    if i > n-1 {
        return -1, simLr
    }

    for {
        if input[j] == text[i] {
            if j == 0 {
                return i, simLr
            } else {
                i--
                j--
            }
        } else {
            lo := bl[text[i]]
            i += m - int(math.Min(float64(j), float64(lo+1)))
            j = m - 1
        }
        if i > n-1 {
            break
        }
    }

    return -1, simLr
}
```

c. Regex

```go
func CheckCalculate(input string) (bool, string) {
    pattern := "Hitunglah (.+)"

    re := regexp.MustCompile(pattern)
    matches := re.FindStringSubmatch(input)

    if len(matches) == 0 {
        return false, ""
    }

    mathPattern := `^[\d*/+\-^()]*$`

    rex := regexp.MustCompile(mathPattern)
    matchesx := rex.MatchString(matches[1])

    if matchesx {
        return matchesx, matches[1]
    }

    return false, ""
}
```

C. PENGGUNAAN ALGORITMA STRING MATCHING DAN REGEX

Algoritma string matching digunakan dalam berbagai situasi di mana kita perlu mencari atau mencocokkan pola tertentu dalam sebuah teks atau string yang lebih panjang. Berikut adalah beberapa contoh penggunaan umum algoritma string matching:

1. Pencarian kata kunci: Dalam mesin pencari atau aplikasi yang memerlukan pencarian berbasis teks, algoritma string matching digunakan untuk mencari kata kunci atau frasa yang dimasukkan oleh pengguna dalam teks yang lebih besar. Algoritma ini memungkinkan kita untuk menemukan kemunculan kata kunci dalam teks dengan cepat dan akurat.
2. Validasi input: Dalam pengembangan perangkat lunak, algoritma string matching sering digunakan untuk memvalidasi input pengguna. Misalnya, saat kita ingin memeriksa apakah input pengguna cocok dengan format yang diinginkan, seperti nomor telepon, alamat email, atau kode pos.
3. Analisis log: Dalam sistem yang menghasilkan log aktivitas, algoritma string matching digunakan untuk mencari pola atau kejadian khusus dalam log tersebut. Ini berguna dalam pemantauan sistem, pemecahan masalah, atau analisis keamanan untuk mengidentifikasi pola yang tidak diinginkan atau aneh dalam log.
4. Pengolahan teks: Algoritma string matching juga digunakan dalam pengolahan teks, seperti manipulasi dan penggantian teks. Misalnya, saat kita ingin mengganti semua kemunculan suatu kata dengan kata lain dalam sebuah teks, atau saat kita ingin memisahkan teks menjadi bagian-bagian yang lebih kecil berdasarkan pola tertentu.
5. Bioinformatika: Dalam bidang bioinformatika, algoritma string matching digunakan untuk mencocokkan dan menganalisis pola DNA, RNA, atau protein dalam rangka memahami struktur genetik dan hubungan antara organisme.

Regex (Regular Expression) digunakan dalam berbagai konteks di mana kita perlu mencocokkan, memanipulasi, atau memvalidasi pola teks tertentu. Berikut adalah beberapa contoh penggunaan umum regex:

1. Pencarian dan penggantian teks: Regex digunakan dalam aplikasi pengolahan teks untuk mencari dan mengganti teks yang sesuai dengan pola yang ditentukan. Misalnya, ketika kita ingin mengganti semua kemunculan kata tertentu dengan kata lain dalam sebuah teks, regex dapat digunakan untuk mencocokkan dan menggantikan pola kata tersebut.
2. Validasi input: Regex sering digunakan untuk memvalidasi input pengguna dalam berbagai bentuk, seperti alamat email, nomor telepon, kode pos, atau format tanggal. Dengan menggunakan regex, kita dapat memeriksa apakah input pengguna cocok

dengan pola yang diharapkan sebelum memprosesnya lebih lanjut.

3. Analisis log: Dalam analisis log atau pemantauan sistem, regex digunakan untuk mencari dan mengekstraksi informasi tertentu dari log yang memiliki pola khusus. Misalnya, kita dapat menggunakan regex untuk mencocokkan pola pesan error dalam log dan mengidentifikasi penyebab masalah.
4. Pemrosesan bahasa alami: Regex juga digunakan dalam pemrosesan bahasa alami untuk mencocokkan pola kata-kata, frasa, atau aturan sintaksis tertentu. Ini berguna dalam memisahkan kata-kata dalam kalimat, mengekstraksi informasi spesifik, atau menerapkan aturan gramatikal.
5. Pengolahan data teks: Regex digunakan dalam pengolahan data teks untuk mengidentifikasi, memisahkan, atau mengekstraksi informasi tertentu dari data. Misalnya, regex dapat digunakan untuk mencocokkan pola nomor telepon dalam data kontak, mencari URL dalam teks, atau mengekstraksi nomor ID dari dokumen.

D. FORMULA ALGORITMA STRING MATCHING DAN REGEX

Pada algoritma string matching terdapat banyak sekali jenisnya tetapi pada percobaan kali ini hanya menggunakan KMP dan regex.

Algoritma Knuth-Morris-Pratt (KMP) adalah salah satu algoritma string matching yang efisien untuk mencocokkan sebuah pola dengan sebuah teks. Algoritma ini menggunakan pendekatan cerdas dengan memanfaatkan informasi tentang pola itu sendiri.

Berikut adalah rumus dasar untuk algoritma KMP:

1. Buat tabel prefiks dan sufiks terpanjang (disebut juga tabel lompatan atau tabel pi) dari pola. Tabel ini menyimpan informasi tentang panjang maksimum prefiks yang juga merupakan sufiks dari bagian pola yang sudah diuji sejauh ini.
   Contoh:
   Misalkan pola yang ingin kita cari adalah "ABCDABD". Tabel prefiks dan sufiks terpanjang untuk pola ini adalah: [0, 0, 0, 0, 1, 2, 0].
2. Lakukan pencocokan pola dalam teks menggunakan tabel prefiks dan sufiks terpanjang.
   - Inisialisasi indeks pola dan indeks teks ke 0.
   - Jika karakter pada indeks pola dan indeks teks cocok, maka increment kedua indeks tersebut.
   - Jika indeks pola mencapai panjang pola, berarti pola sudah ditemukan dalam teks. Lakukan tindakan yang diinginkan (misalnya, mencatat posisi atau melakukan manipulasi pada teks).
   - Jika karakter pada indeks pola dan indeks teks tidak cocok, gunakan tabel prefiks dan sufiks terpanjang untuk menentukan jumlah pergeseran yang tepat dalam pola. Pergeseran ini dilakukan dengan memindahkan indeks pola ke posisi yang sesuai di dalam tabel.

Berikut adalah langkah-langkah untuk mengisi tabel prefiks dan sufiks terpanjang (tabel pi) dari pola:

1. Inisialisasi tabel dengan semua nilai 0.
2. Lakukan perulangan dari indeks 1 hingga panjang pola - 1.
3. Jika karakter pada indeks sekarang tidak cocok dengan karakter pada indeks sebelumnya, maka lakukan langkah berikut:
   - Cari panjang maksimum prefiks yang juga merupakan sufiks dari bagian pola sebelum indeks sekarang.
   - Gunakan informasi ini untuk memperbarui tabel pada indeks sekarang.
   - Kembali ke langkah 2 untuk indeks berikutnya.

Dengan menggunakan tabel prefiks dan sufiks terpanjang ini, algoritma KMP dapat menghindari pencocokan ulang dalam pola dan mencapai efisiensi yang lebih tinggi dalam mencari pola dalam teks.

Algoritma KMP sangat bermanfaat dalam situasi di mana kita perlu mencocokkan pola yang sama secara berulang dalam teks yang panjang. Dengan menggunakan tabel prefiks dan sufiks terpanjang, algoritma ini mempercepat pencocokan pola dan mengurangi jumlah perbandingan yang perlu dilakukan, sehingga meningkatkan kinerja secara keseluruhan.

Regex (Regular Expression) adalah suatu urutan karakter yang membentuk sebuah pola yang digunakan untuk mencocokkan atau memanipulasi teks sesuai dengan pola yang diinginkan. Regex menggunakan simbol khusus dan metakarakter yang memiliki makna khusus untuk melakukan pencocokan pola.

Berikut adalah beberapa rumus yang umum digunakan dalam regex:

1. Karakter Biasa: Karakter-karakter biasa dalam regex akan cocok secara harfiah dengan karakter yang sama dalam teks. Misalnya, karakter "a" akan cocok dengan karakter "a" dalam teks.
2. Metakarakter: Metakarakter adalah karakter khusus dalam regex yang memiliki makna khusus. Beberapa contoh metakarakter yang umum digunakan adalah: "." (titik) cocok dengan karakter apa pun, kecuali karakter baris baru. "^" (tanda caret) cocok dengan awal dari sebuah teks atau baris. "$" (tanda dollar) cocok dengan akhir dari sebuah teks atau baris. "*" (asterisk) cocok dengan nol atau lebih kemunculan karakter sebelumnya. "+" (tanda tambah) cocok dengan satu atau lebih kemunculan karakter sebelumnya. "?" (tanda tanya) cocok dengan nol atau satu kemunculan karakter sebelumnya. "[]" (bracket) digunakan untuk mencocokkan satu karakter dari kumpulan karakter yang diberikan. "()" (parentheses) digunakan untuk mengelompokkan bagian-bagian pola.

3. Karakter Escaping: Karakter-karakter khusus dalam regex dapat di-escape menggunakan tanda "\ (backslash)". Misalnya, "." akan cocok dengan titik secara harfiah, bukan sebagai metakarakter.
4. Karakter Spesial: Ada beberapa karakter dalam regex yang memiliki arti khusus dan harus dihindari jika kita ingin mencocokkan karakter tersebut secara harfiah. Beberapa karakter tersebut adalah ".", "*", "+", "?", "[", "]", "(", ")", "{", "}", "^", "$", "", dan "|".

Selain rumus-rumus di atas, regex juga memiliki penanda kuantifier seperti "{n}", "{n,}", dan "{n,m}" untuk menentukan jumlah kemunculan yang diinginkan dari karakter sebelumnya.

Penggunaan regex sangat luas dan fleksibel dalam memanipulasi teks. Dengan menggunakan pola regex, kita dapat mencari, mengganti, membagi, dan memvalidasi teks dengan cara yang sangat efisien. Regex digunakan dalam berbagai konteks seperti pemrograman, pengolahan teks, pemrosesan bahasa alami, validasi input, ekstraksi data, dan masih banyak lagi.

III. PENGGUNAAN ALGORITMA STRING MATCHING DAN REGEX DALAM MENYARING DATA PENDAFTARAN USER

Pada permasalahan kali ini seseorang ingin dibuatkan aplikasi perusahaan yang bisa menyaring siapa saja yang bisa melakukan pendaftaran pada aplikasi tersebut dengan menggunakan email dan nomor handphone.

Ia ingin hanya user yang memiliki email perusahaan dan memiliki nomor handphone yang valid saja yang bisa mendaftar. Ia pun ingin nomor handphone dan email dari user yang ingin mendaftar harus divalidasi terlebih dahulu apakah nomor handphonenya sudah benar atau tidak apakah email yang didaftarkan sudah benar atau tidak.

Pada percobaan kali ini akan digunakan string matching dan regex dalam memecahkan permasalah tersebut.

A. MENGGUNAKAN STRING MATCHING DAN REGEX DALAM PENYARINGAN PENDAFTARAN EMAIL USER

Pertama-tama yang harus dilakukan adalah dengan mendapatkan input email dari user dengan memvalidasinya apakah email panjangnya sudah sesuai dengan email yang valid atau tidak jika sudah lalu mengecek kalimat setelah “@” apakah valid email tersebut dari perusahaan yang ada.

Berikut adalah kode dalam mengecek input email yang valid menggunakan regex pada kasus kali ini perusahaan memiliki email “xxx@wallpaper.Collect.co.id”

```go
import "regexp"

func ValidateEmail(e string) bool {

    pattern := "
(.+)@wallpaper.Collect.app"
    re := regexp.MustCompile(pattern)
    matches := re.FindStringSubmatch(e)

    if len(matches) == 3 && len(matches)
> 254 {
        return false
    }

    return true
}
```

Pada kode tersebut digunakan untuk melakukan penyaringan apakah user melakukan pendaftaran dengan menggunakan email perusahaan yang sesuai atau tidak jika iya maka akan mengembalikan nilai true dan jika tidak maka mengembalikan nilai false.

Untuk melakukan validasi dengan string matching digunakan cara sebagai berikut dengan menggunakan kode KMP seperti ini:

```go
func KMP(input string, text string) (int, float64)
{
    n, m, i, j := len(text), len(input), 0, 0
    b := sufPref(input)
    simLr := HammingDistance(input, text)

    for i < n {
        if input[j] == text[i] {
            if j == m-1 {
                return i - m + 1, simLr
            }
            i++
            j++
        } else if j > 0 {
            j = b[j-1]
        } else {
            i++
        }
    }

    return -1, simLr
}
```

Dengan menggunakan fungsi validasi email yang melakukan pemanggilan fungsi di atas sebagai berikut

```
func ValidateEmail(input string) bool {

    pattern := "@wallpaper.Collect.app"

    index,_:= KMP(pattern, input)

    if len(index) == -1{
        return false
    }

    return true
}
```

a.

Pada kode tersebut digunakan untuk melakukan validasi paeda nomor handphone dengan menggunakan regex dengan menyaring nomor handphone yang memiliki simbol yang tidak sesuai dengan nomor handphone yang valid jika nomor handphone memiliki simbol yang terdapat pada simbol simbol yang dilarang maka nomor handphone tersebut merupakan nomor yang tidak valid, jika nomor handphone tersebut tidak memiliki simbol yang dilarang maka ia akan dilakukan pengecekan apakah nomor handphone tersebut memiliki panajng nomor handphone yang valid jika iya maka akan melewati pengecekan tersebut jika tidak maka akan mengembalikan nilai false, setelah melakukan pengecekan tersebut dilakukan pengecekan lagi apakah nomor tersebut berawalah + atau 0 jika bukan dari keduanya maka akan mengembalikan nilai false jika tidak maka akan mengembalikan nilai true.

B. MENGGUNAKAN REGEX DALAM MEMVALIDASI NOMOR HANDPHONE

Dalam menggunakan regex untuk melakukan validasi kepada nomor handphone apkah kodenya valid atau tidak. Bisa dilakukan dengan cara membuat kode seperti ini menggunakan regex.

```
import "regexp"

func ValidationNumberPhone(numberPhone
string) bool {
    if len(numberPhone) < 9 {
        return false
    }
    var mustNotIn =
regexp.MustCompile("^[a-zA-
Z!#$%&'*\\/=?^_`{|}~-]$")

    if
mustNotIn.MatchString(numberPhone) {
        return false
    }

    pattern := "08(.+)"
    re := regexp.MustCompile(pattern)
    matches :=
re.FindStringSubmatch(numberPhone)

    if len(matches) != 0 {
        return true
    }

    pattern = `\+(.+)`
    re = regexp.MustCompile(pattern)
    matches =
re.FindStringSubmatch(numberPhone)

    if len(matches) != 0 {
        return true
    }

    return false
}
```

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf (diakses pada 21 mei 2023)

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, 22 Mei 2023

Farhrian Afdholi dan 13521031