Aplikasi Algoritma String Matching dan Regex untuk Validasi Formulir

Edmund Ophie - 13512095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*edmund.ophie@yahoo.com*

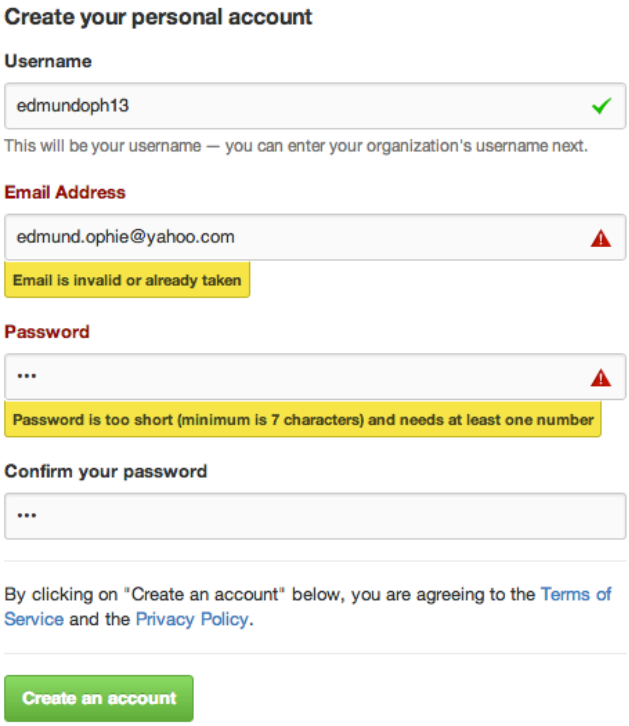
Sekarang ini hampir sebagian besar layanan berbasis web yang dibangun membutuhkan informasi tentang pengguna layanannya. Tidak jarang informasi ini menjadi hal yang wajib dipenuhi agar layanan tsb dapat digunakan oleh pengguna. Informasi yang diminta pun biasanya bermacam-macam. Mulai dari username, password hingga informasi yang sifatnya pribadi seperti nomor kartu kredit. Suatu layanan biasanya menyediakan halaman tersendiri untuk mengumpulkan informasi penggunanya. Halaman ini umumnya dibuat dalam bentuk formulir. Pada dasarnya pengguna dapat memasukkan sembarang informasi pada formulir yang telah disediakan. Hal ini dapat menyebabkan pemborosan pada database. Oleh karena itu dibutuhkan suatu teknik untuk memvalidasi masukan dari pengguna sehingga menjamin informasi yang dimasukkan sudah sesuai yang diminta. Untuk itu dalam makalah ini dipaparkan aplikasi algoritma string matching yang dikombinasikan dengan pattern matching regex untuk memvalidasi suatu string.

***Kata kunci* — regex, regular expression, string matching, validasi**

I. PENDAHULUAN

Di era seperti sekarang ini sudah banyak bermunculan layanan-layanan berbasis web. Tren pun berkembang menuju era cloud. Kemampuan aplikasi berbasis web yang ditawarkan mulai menyamai aplikasi berbasis desktop bahkan ada beberapa aplikasi berbasis web yang mengungguli mutklak suatu aplikasi berbasis dekstop. Kebutuhkan masyarakat yang lebih mobile menuntut tersedianya layanan dimanapun mereka berbeda. Jumlah pengguna pun terus bertambah dari hari ke hari dengan tingkat akselerasi eksponensial growth. Untuk membedakan pengguna sebanyak ini tentunya diperlukan suatu informasi tentang pengguna sehingga sistem dapat mengidentifikasi pengguna yang sedang menggunakan suatu layanan.

Pada saat mendaftar sebagai pengguna di suatu situs, misalnya ebay, pengguna akan diminta untuk mengisi informasi mengenai email, kartu kredit, dan username yang ingin dipilih. Ketika kita selesai mengisikan username, sistem biasanya akan mencocokkan secara otomatis dengan database dan mencari apakah username yang dipilih sudah digunakan atau belum. Jika belum maka akan diberikan tanda hijau. Khusus untuk pengisian alamat email dan kartu kredit ada langkah tambahan yang dilakukan oleh sistem. Selain mencari kecocokan dengan database, alamat email dan kartu kredit akan divalidasi patternnya. Hal ini dimaksudkan untuk memastikan user telah memasukkan sebuah email dan sebuah kartu kredit. sehingga dapat menghindari dimasukkannya data secara sembarangan.

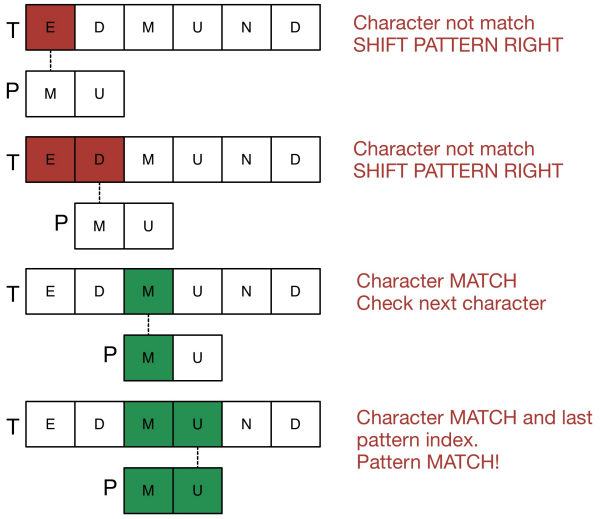


Gambar 1. Aplikasi string matching dan regex pada formulir registrasi online

Tentunya hal ini sangat banyak manfaatnya. Dampak yang paling dirasakan ada pada pengguna. Dengan adanya sistem verifikasi informasi yang dimasukkan akan memberitahu pengguna jika terdapat typo. Selain itu, sistem verifikasi akan menghemat dan menjaga database tetap efisien dengan menjamin bahwa data yang dimasukkan adalah informasi yang valid .

II. DASAR TEORI

II.1 Algoritma String Matching Brute Force



Gambar 2. Algoritma Brute Force

Pada algoritma brute force, 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 tsb 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 tsb tidak ditemukan pada teks.

Pada algoritma ini, semakin banyak variasi karakter pada suatu teks maka akan semakin baik dan cepat performansi algoritmanya. Kasus terburuk pada string matching terjadi ketika ketidakcocokan karakter selalu terjadi pada ujung pattern, misalnya ketika teks adalah "bbbbbbbbbbba" dan patternya "bba". 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). Kasus terbaik terjadi ketika ketidakcocokan selalu terjadi pada indeks pertama pada pattern, misalnya ketika teks adalah "abcdefghijxxx" dan patternnya "xxx". Pada kasus terbaik jumlah perbandingan maksimal adalah n kali sehingga kompleksitasnya O(n).

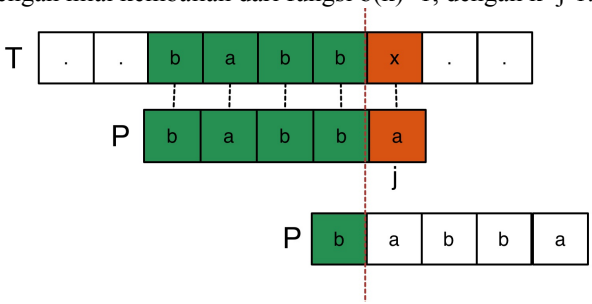
Akan tetapi pada kenyataannya jarang ditemui kasus terbaik atau terburuk. Kasus rata-rata yang ditemukan mempunyai kompleksitas O(m+n).

II.2 Algoritma String Matching Knuth-Morris-Pratt

Algoritma yang ditemukan oleh Donald E. Knuth ini sebenarnya sama aja dengan algoritma bruteforce, membandingkan karakter dari kiri ke kanan, hanya saja algoritma ini 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.

Misal i adalah nomor indeks teks dan j adalah nomor indeks pattern. Ketika terjadi ketidakcocokan saat perbandingan T[i] dan P[j] maka j akan digantikan dengan nilai kembalian dari fungsi b(k)+1, dengan k=j-1.



Gambar 3. Pergeseran pattern pada 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.

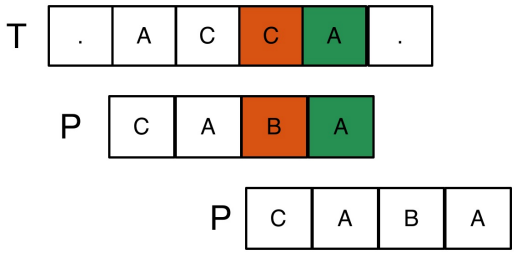
Algoritma KMP sendiri menawarkan keuntungan dan kekurangan tersendiri, Dengan algoritmanya maka dengan KMP bisa dipastikan tidak ada pencocokan yang berulang pada pattern yang sudah dibandingkan dan sama sebelumnya, sedangkan kekurangannya adalah KMP kemangkusannya akan berkurang apabila pattern yang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

II.3 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

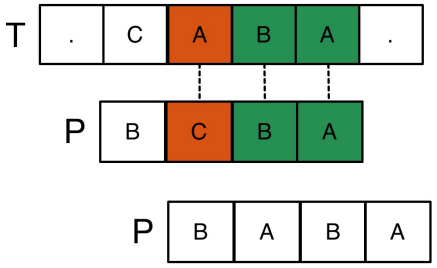
maka pattern akan digeser ke kanan hingga ditemukan karakter yang sama dengan karakter pada teks.



Gambar 4. Kasus 1 pada Boyer-Moore

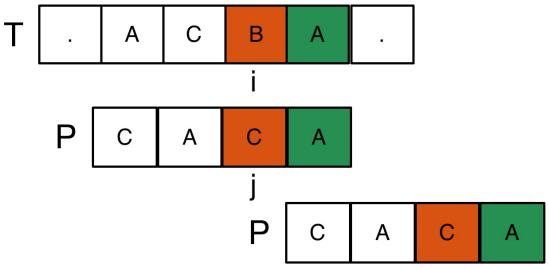
Seperti dilihat pada gambar 4, pertama-tama karakter dicocokkan mulai dari indeks terakhir. Ketika karakternya cocok, maka pencocokan berlanjut ke indeks sebelumnya. Ketika "C" dibandingkan dengan "B" dan tidak cocok maka pattern akan digeser ke kanan hingga "C" di pattern sejajar dengan "C" di teks.

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 5. Kasus 2 pada Boyer-Moore

Kasus 3 Boyer-Moore terjadi ketika ditemukan ketidakcocokan antarar karakter 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 6. 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".

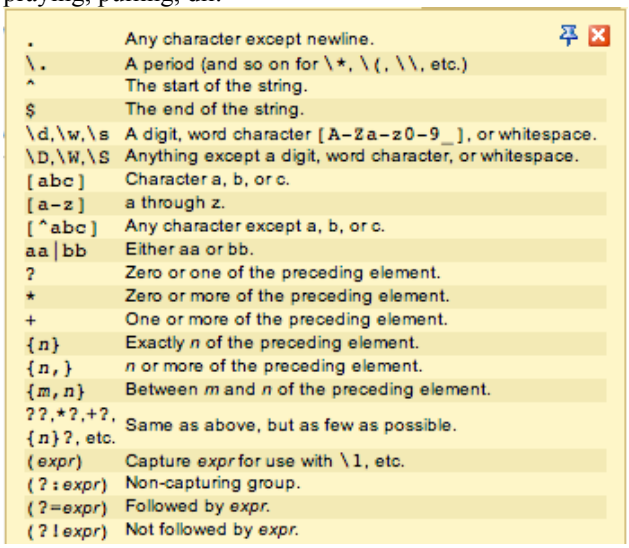
Algoritma Boyer-Moore akan semakin baik performansinya ketika alfabetnya semakin bervariasi. Hal ini menyebabkan algoritma ini cepat ketika digunakan untuk mencari teks pada bahasa sehari-hari namun akan berkurang performansinya ketika digunakan untuk mencari pada string biner.

II.4 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*.

Gambar 7. Notasi pada regular expression
Sumber: regexpal.com

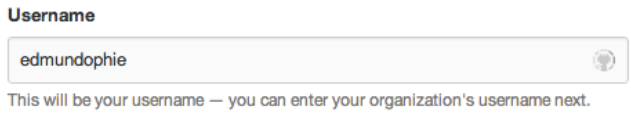
Notasi regex dapat dikombinasikan sedemikian kompleksnya sehingga dapat menemukan kata yang mempunya pola-pola cukup rumit. Adapun contohnya:

```
^[_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 john@yahoo.com

III. PENERAPAN

Sistem validasi formulir pada sebuah website baiknya terjadi secara langsung menggunakan ajax. 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.



Gambar 8. Proses validasi sedang berlangsung

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.

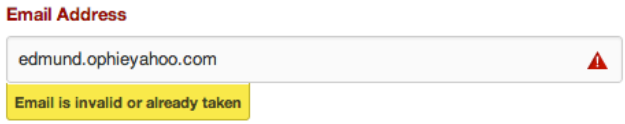
Kasus pertama, data dikatakan valid ketika data yang dimasukkan adalah data yang diminta. Misal, pada textbox email address yang diisi harus benar-benar berbentuk sebuah email, bukannya nama atau tanggal lahir. Disinilah peran regex dan regex seharusnya dapat mengenali apakah data yang dimasukkan sudah valid.

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 tsb haruslah unik. Ketika username yang dimasukkan tidak unik maka masukkan tsb dianggap tidak valid. Untuk mengatasi kasus kedua ini dapat dimanfaatkan salah satu dari algoritma string matching yang ada.

Untuk meninjau kasus pertama akan diambil contoh pada validasi email dan password pada github.com. Ketika " edmund.ophieyahoo.com" 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.

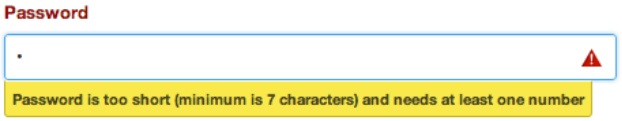


Gambar 9. Pesan invalid email

Untuk memvalidasi password dengan syarat minimal 7 karakter dan mempunyai paling sedikit satu angka dapat digunakan regex:

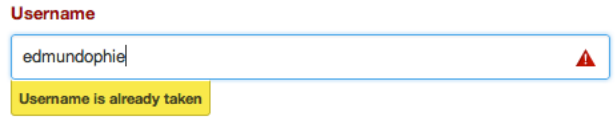
```
^(?=.*\d).{7,}$
```

Penjelasan regex diatas sbb:
- `(?=.*\d).` menerima 0 atau lebih karakter yang diikuti oleh sebuah angka.
- `{7,}$` menerima jika string tsb mempunyai panjang minimal 7 karakter.



Gambar 10. Pesan error karena password kurang

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 Kntuh-Morris-Pratt 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.



Gambar 11. Pesan bahwa username tidak tersedia lagi

Misal username yang dimasukkan "edmundophie" dan didatabase terdapat username terdaftar "edmundophieku", maka string matching tidak akan mengembalikan true saat diperiksa. Jadi disini yang menjadi pattern adalah username yang dimasukkan dan yang menjadi teks adalah daftar username yang ada di database.

```php
<?php
$string = '1n1passw0rd';

if(preg_match("/^(?=.*\d).{7,}$/", $string))
    {
    echo 'Password valid';
    }
else
    {
    echo 'Error! Password tidak valid.';
    }
?>
```

Gambar 12. Regex pada PHP

Dibalik kelebihan 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, edmund..ophie@yahoo.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.

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 batasan-batasan yang dimiliki regex. Kalaupun hal tsb dapat dilakukan maka akan dibutuhkan notasi regex yang sangat panjang dan kompleks untuk dimengerti.

Oleh sebab itu, regex hanyalah sebagai pendeteksi masukan sederhana dan program seharusnya tidak bergantung sepenuhnya pada regex sebagai fungsi untuk validasi masukan.

REFERENSI.

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

[2] http://www.phpro.org/tutorials/Introduction-to-PHP-Regex.html Diakses 1 Mei 2014 12:20

[3] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp#Special_characters_in_regular_expressions Diakses 1 Mei 2014 11.30

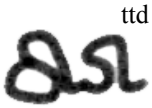
[4] http://fivedots.coe.psu.ac.th/Software.coe/LAB/PatMatch/PatternMatching.ppt Diakses 29 April 2014 21:34

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, 1 Mei 2014

ttd



Edmund Ophie - 13512095