Penerapan Pencocokan String pada Pencarian Message di Aplikasi Line

Christian Kevin Saputra / 13516073
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
ck3putra@gmail.com

***Abstract*— Di zaman sekarang ini, tidak dapat dipungkiri bahwa penggunaan media sosial sudah menjadi kebiasaan di dalam masyarakat. Terdapat berbagai macam media sosial seperti Line, Instagram, WhatsApp dan berbagai macam lainnya. Salah satu fungsi yang berada di dalam social media tersebut adalah fungsi untuk mencari sebuah message untuk mempermudah penggunanya. Pencarian message di dalam media sosial ini merupakan salah satu penerapan dari algoritma pencocokan string (*string matching*). Makalah ini akan membahas mengenai algoritma pencocokan string yang diterapkan dalam fungsi pencarian pada aplikasi Line.**

Keywords—String Matching, Line, Social Media.

I. PENGENALAN

Perkembangan teknologi telah mempermudah hubungan antar manusia di mana pun dan kapanpun. Salah satu bentuk nyata dari perkembangan teknologi adalah berkembangnya aplikasi – aplikasi social media. Dengan adanya social media , berkomunikasi dengan kerabat yang berbeda negara bukanlah hal yang mustahil. Salah satu media sosial yang marak digunakan oleh masyarakat Indonesia adalah Line.

Line adalah sebuah aplikasi media sosial yang menawarkan berbagai fitur seperti *chatting* , *news*, *timeline*, *sticker*, dan *games.* Aplikasi yang tidak membutuhkan biaya untuk digunakan ini menjadi salah satu aplikasi yang dipilih oleh masyarakat Indonesia (khususnya kaum muda) dalam berinteraksi dengan satu sama lain . Dalam menggunakan Line untuk berinteraksi dengan sesama, tentunya dibutuhkan sebuah fitur untuk mencari message untuk mempermudah penggunanya. Fitur ini ditetapkan di dalam Line dan merupakan salah satu bentuk penerapan algoritma *string matching.*

Apa itu algoritma *string matching? String matching* atau pencocokan string merupakan sebuah algoritma yang digunakan untuk menentukan sebuah pattern dari sebuah kata ataupun kalimat. Misalkan terdapat sebuah kalimat “aku ingin lulus” dan pattern “ingin”, maka algoritma *string matching* akan mencari pattern “ingin” di dalam kalimat tersebut. Terdapat berbagai variasi algoritma dalam *string matching* seperti algoritma brute force, algoritma KMP, algoritma Boyer – Moore, dan *string matching* dengan menggunakan *regular – expression.*

II. DASAR TEORI

A. *Algoritma Pencocokan String (String Matching)*

Algoritma pencocokan string adalah algoritma yang menerima sebuah pattern dan sebuah text. Tujuan dari algoritma ini adalah untuk mengetahui apakah pattern yang didefinisikan dapat ditemukan pada text tersebut. Biasanya algoritma pencocokan string digunakan dalam beberapa hal seperti pencarian text di dalam sebuah aplikasi (seperti pada Line yang akan dibahas di dalam makalah ini), *web search engine*, analisis citra, dan bioinformatics seperti dalam pencocokan DNA.

Berikut adalah illustrasi pencocokan string secara brute – force :

Text : Aku ingin lulus

Pattern : ingin

```
Aku ingin lulus
i
 i
  i
   i
    ingin
```

Pada pencocokan secara brute – force, pattern akan dicocokkan per karakter dengan text dimulai dari awal text. Apabila berbeda, maka pattern akan bergeser ke karakter berikutnya. Hal tersebut akan diulangi hingga akhirnya terdapat kesamaan pada seluruh pattern ataupun text berakhir yang berarti tidak terdapat pattern yang sama dalam text tersebut. Pseudocode untuk algoritma pencocokan string secara *brute force* dapat dilihat di bawah ini:

```
T = text
P = pattern
m = panjang dari teks
n = panjang dari pattern

for (int i =0 ; i<m; i++){
```

```
        for(int j=0; j<m && i+j<n;j++){
                if(T[i+j] != P[j]){
//karakter dalam pattern tidak sesuai
dengan karakter pada text
                        break
                }
                else if(j==m){
        //pattern ditemukan dalam text
                //lakukan sesuatu
                }
        }
}
```

Walaupun algoritma di atas dapat digunakan untuk mencari pattern di dalam sebuah text, namun keefektifan dan keefisienan program akan sangat berkurang apabila text dan pattern yang dicari sangat panjang. Hal tersebut dapat terjadi karena pergeseran karakter pada saat pencocokan hanya dilakukan sekali setiap tidak ditemukan kecocokan. Oleh karena itu, tidak dianjurkan untuk menggunakan algoritma brute-force apabila pattern dan text yang dicari panjang dan menggunakan algoritma lain seperti KMP , Boyer – Moore, ataupun regex.

B. Algoritma Knuth – Morris – Pratt

Algoritma KMP atau Knuth – Morris – Pratt adalah sebuah algoritma *string matching* yang dikembangkan oleh Donald Knuth, Vaughan Pratt, dan James H. Morris. Algoritma KMP ini melakukan pencocokan string yang dimulai dari kiri ke kanan sama seperti algoritma *brute – force* namun memiliki suatu informasi untuk menentukan jumlah pergeseran karakter sehingga tidak menggeser karakter satu per satu seperti pada algoritma *brute – force* sehingga menghemat waktu pencarian.

Seperti dikatakan sebelumnya, algoritma KMP menyimpan sebuah informasi yang membuat pergeseran tidak satu per satu. Informasi tersebu didapatkan melalui fungsi pinggiran (*border function*). Fungsi pinggiran merupakan sebuah fungsi yang dilakukan terhadap pattern yang akan dicari untuk menentukan seberapa jauh karakter akan berpindah agar tidak terjadi pergeseran yang tidak berguna.

Fungsi pinggiran sendiri merupakan fungsi yang dikerjakan sebelum pencocokan string dimulai karena fungsi pinggiran hanya diterapkan terhadap pattern. Cara menghitung fungsi pinggiran adalah dengan mencari panjang kata yang merupakan prefix dari pattern[0..j-1] dan suffix dari pattern[1..j-1] (Dengan j adalah panjang pattern). Agar lebih jelas, dapat diilustrasikan dengan tabel berikut :

j	0	1	2	3	4	5
P[j]	*a*	*b*	*a*	*a*	*b*	*a*
k	-	0	1	2	3	4
b(k)	-	0	0	1	1	2

Gambar 1 Tabel Algoritma KMP dengan fungsi pinggiran
sumber : Slide Pencocokan String (Revisi 2018), Kuliah Strategi Algoritma, 2018

Pada gambar 1, pattern yang dicari adalah abaaba. Fungsi pinggiran terlihat pada baris b(k) dengan k adalah j-1. Mengapa b(4) = 2? b(4) berarti akan mencari panjang dari kata yang merupakan prefix p[0..4] (kata “abaab”) sekaligus suffix p[1..4] (kata “baab”). Dengan demikian kata yang merupan prefix sekaligus suffix adalah “ab” dengan panjang 2. Oleh karena itu, nilai dari b(4) adalah 2. Berikut adalah pseudocode dari algoritma untuk fungsi pinggiran:

```
computeFail(String pattern){
        fail[panjang pattern] //bentuk
array sepanjang pattern
        fail[0] = 0;
        int m = panjang pattern
        int j = 0;
        int i = 1;
        while (i < m) {
                if (pattern[j]
==pattern[i]) { //j+1 chars match
                        fail[i] = j + 1;
                        i++;
                        j++;
                }
                else if (j > 0){ // j
follows matching prefix
                        j = fail[j-1];
                }
                else { // no match
                        fail[i] = 0;
                        i++;
                }
        }
        return fail;
}
```

Sumber : Slide Pencocokan String (Revisi 2018), Kuliah Strategi Algoritma, 2018

Pseudocode Algoritma KMP sendiri :

```
kmpMatch(String text, String pattern){
        int n = panjang text
        int m = panjang pattern
```

```
        int fail[] = computeFail(pattern);
//fungsi pinggiran
        int i=0;
        int j=0;
        while (i < n) {
                if (pattern[j] == text[i])
{
                        if (j == m - 1)
                                return i -
m + 1; // match
                                i++;
                                j++;
                }
                else if (j > 0)
                        j = fail[j-1];
                else
                        i++;
                }
        return -1; // no match
}
```

Sumber : Slide Pencocokan String (Revisi 2018), Kuliah Strategi Algoritma, 2018

Dibandingkan dengan algoritma brute – force yang memiliki kompleksitas O(mn), algoritma KMP lebih efektif dengan kompleksitas O(m+n) dengan O(m) merupakan waktu untuk menghitung fungsi pinggiran.

C. *Algoritma Boyer - Moore*

Algoritma Boyer – Moore merupakan algoritma yang dikembangkan oleh Robert S. Boyer dan J. Strother Moore. Berbeda dengan algoritma KMP dan *brute – force* yang melakukan pencocokan dari awal pattern ke akhir, algoritma Boyer – Moore melakukan pencocokan dari akhir pattern. Algoritma Boyer- Moore menerapkan 2 teknik dalam penerapannya dan sebuah informasi yang membantu pergeseran pattern sehingga pergeseran pattern tidak satu per satu.

Teknik yang digunakan dalam algoritma Boyer – Moore ada 2 yaitu :

1. *The Looking – Glass Technique*

   Teknik ini menyakatan bahwa pattern dicari bukan dari awal pattern, melainkan dari akhir pattern. Misalkan text = ababc dan pattern = abc, maka pencocokan dilakukan sebagai berikut:

   ababc

   abc

    abc

     abc

   Dari contoh tersebut, pattern abc dicocokan dari awal text, namun tidak dimulai dari awal pattern, melainkan dari akhir pattern. (Pencocokan sesuai dengan warna pada illustrasi di atas)

2. *The Character Jump Techique*

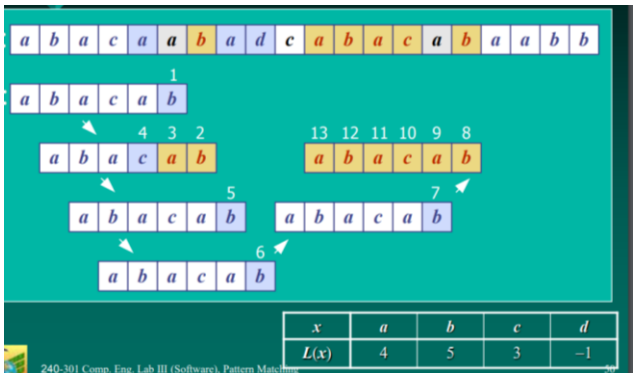
   Teknik ini digunakan ketika terjadi ketidak cocokan pada pattern dan text. Ada 3 kemungkinan yang terjadi :

   a. Jika pattern mengandung karakter yang berbeda di kiri pattern, geser pattern hingga karakter tersebut sejajar dan ulangi pencocokan string

   b. Jika pattern mengandung karakter yang berbeda di kanan, maka geser pattern sebanyak 1 kali.

   c. Jika karakter tidak ditemukan di dalam pattern, geser pattern sebanyak panjang pattern.

Selain kedua teknik tersebut, algoritma Boyer – Moore juga menyimpan informasi mengenai kemunculan terakhir dari karakter pada pattern untuk mempermudah pergeseran. Berikut adalah contoh pencocokan string dengan algoritma Boyer – Moore dan tabel kemunculan terakhirnya :



Gambar 2 Algoritma Boyer - Moore dan tabel kemunculan terakhir
sumber : Slide Pencocokan String (Revisi 2018), Kuliah Strategi Algoritma, 2018

Berikut pseudocode untuk algoritma Boyer – Moore :

```
bmMatch(String text, String pattern){
int last[] = array yang menampung last
occurence karakter-karakter pada pattern.
int n = panjang text
int m = panjang pattern
int i = m-1;
int j = m-1;
do {
        if (pattern[j] == pattern[i])
        if (j == 0){
                return i; // match
        }
        else { // looking-glass technique
                i--;
                j--;
        }
```

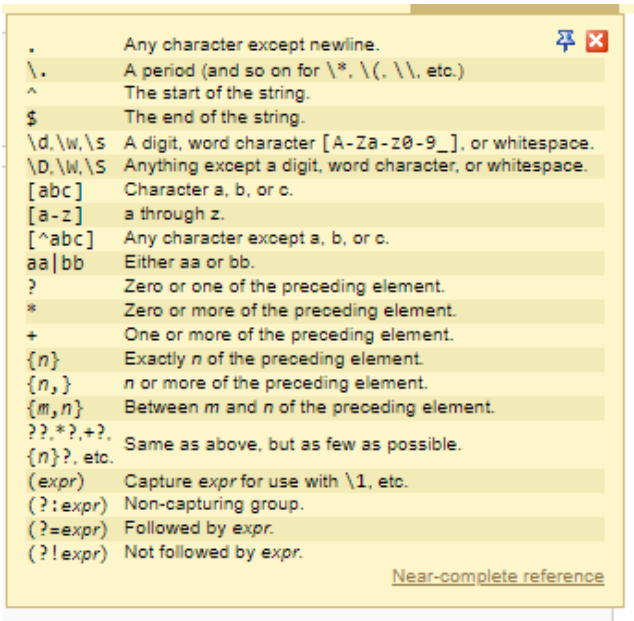
```
        else { // character jump technique
                int lo = last[text[i]];
//last occ
                i = i + m - Math.min(j,
1+lo);
                j = m - 1;
        }
} while (i <= n-1)
```

Sumber : Slide Pencocokan String (Revisi 2018), Kuliah Strategi Algoritma, 2018

Algoritma Boyer – Moore memliki kompleksitas waktu O(mn+A) dengan m adalah panjang pattern, n adalah panjang text, dan A adalah jumlah alphabet pada text.

D. Regular Expression (Regex)

Regular Expression atau Regex adalah sebuah kumpulan notasi yang akan menyatakan sebuah *search pattern*. Kumpulan notasi tersebut dapat dilihat pada gambar 3 berikut:



.	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*.

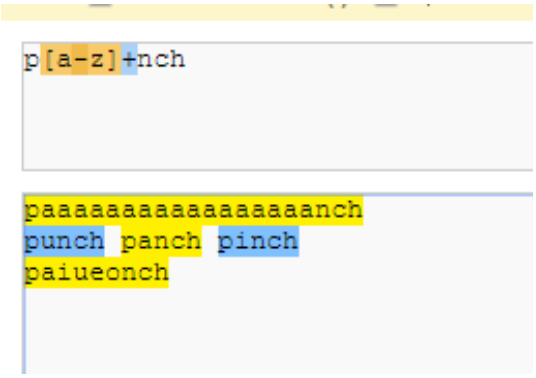
Near-complete reference

Gambar 3 Sumber : http://regexpal.com.s3-website-us-east-1.amazonaws.com/?_ga=2.118453771.365632462.1526241381-317489324.1524379473

Salah satu contoh penggunaan Regex adalah sebagai berikut:

Pattern = p[a-z]+nch

Dari gambar 3 diatas, diketahui bahwa tanda + akan mencari satu atau lebih dari elemen sebelumnya dimana pada pattern diatas adalah [a-z] yang berarti kemunculan satu atau lebih huruf diantara a – z. Maka apabila pattern tersebut digunakan, kata yang akan ditemukan oleh pattern tersebut adalah :



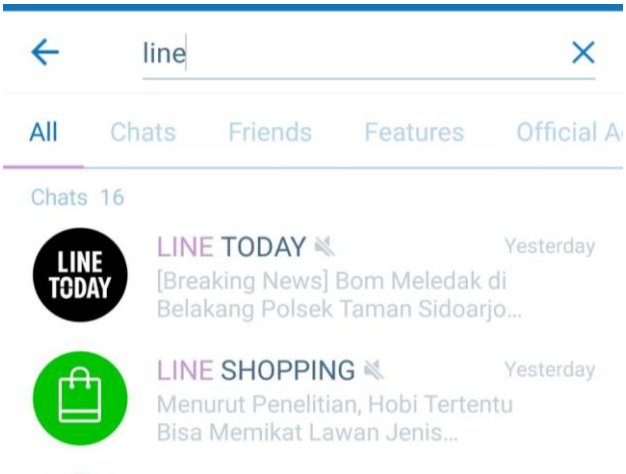
Gambar 4 Percobaan Regex
Sumber : http://regexpal.com.s3-website-us-east-1.amazonaws.com/?_ga=2.118453771.365632462.1526241381-317489324.1524379473

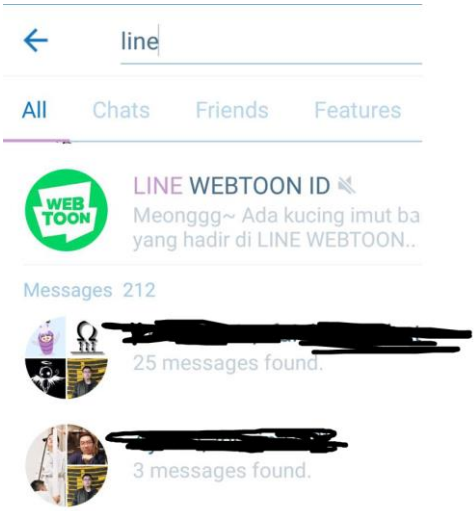
III. Analisa Algoritma Pencocokan String pada fitur Pencarian Message Line

Seperti dikatakan sebelumnya , makalah ini akan membahas fitur pencarian chat dan message yang dimiliki oleh aplikasi Line. Pada aplikasi Line, pencarian chat dan message dapat dilakukan dengan menekan tombol kaca pembesar di bagian atas chat.

Gambar 5 Sumber : Dokumen Pribadi

Pencarian fitur message dapat dimulai dengan menekan tombol kaca pembesar yang diberikan lingkaran pada gambar 5 di atas. Setelah dipilih, pengguna dapat memasukkan pattern (message yang akan dicari) ke dalam text box tersebut. Selain melakukan pencocokan string pada message di dalam chat, fitr pencarian di Line juga melakukan pencockan string dengan nama user lain ataupun nama grup untuk mempermudah pencarian chat. Berikut adalah contoh pencarian di dalam Line:

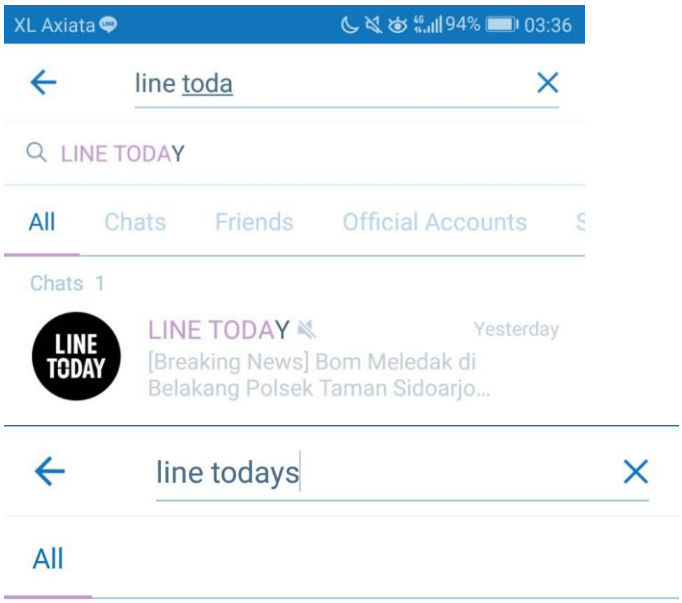




Gambar 5 dan 6 Hasil Pencarian message dan chat
Sumber : Dokumen Pribadi

Pada gambar 5 dan 6 tersebut, dilakukan pencarian terhadap pattern line. Hasil dari pencarian tersebut pertama – tama menghasilkan user atau grup yang mengandung pattern yang kita cari, diikuti dengan message setelah semua user ataupun grup telah selesai dicari. Pencarian juga diurutkan berdasarkan waktu untuk mempermudah pengguna dalam mencari message.

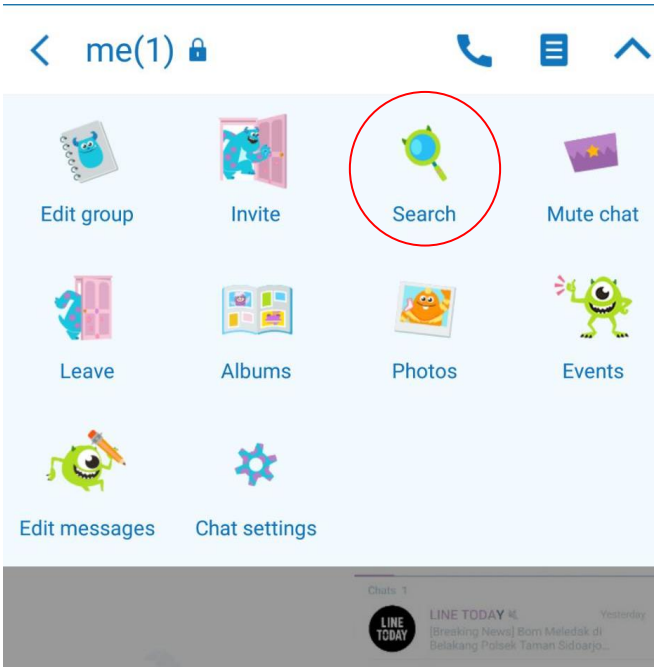
Pencarian yang dilakukan pada pencarian message di awal ini dapat saja menggunakan algoritma KMP ataupun Boyer – Moore dikarenakan pencariannya yang *exact matching* dengan case insensitive dibuktikan dengan line dengan huruf kecil ditemukan pada Line denga huruf L besar. Pembuktian lainnya bahwa pencarian dilakukan dengan algoritma KMP ataupun boyer – moore yang exact matching dapat dilihat pada gambar berikut.



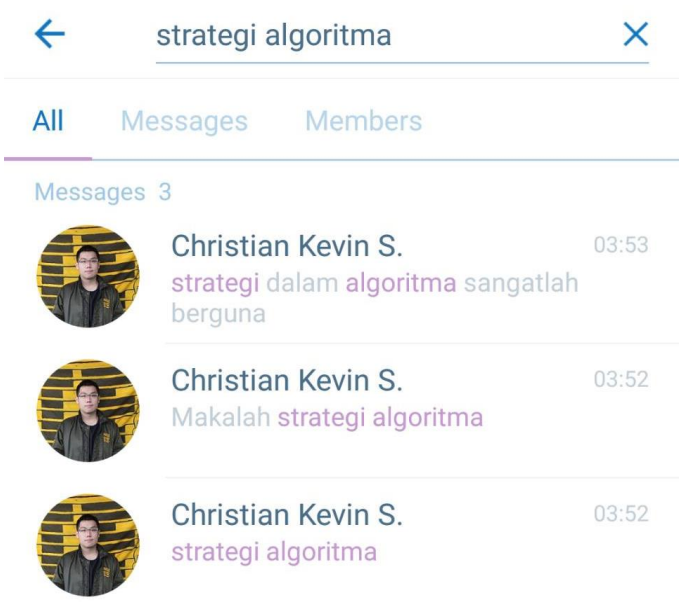
Gambar 7 dan 8 Hasil pencarian message dan chat
Sumber : Dokumen Pribadi

Hasil pada gambar 7 dan 8 semakin menyakinkan bahwa pencarian yang dilakukan adalah pencarian dengan *exact string matching* dikarenakan line toda mendapatkan pasangan pada kata LINE Today sedangkan line todays tidak mendaptkan pasangan. Hal ini menunjukkan bahwa fitur pencarian message dan chat pada Line memungkinkan penggunaan algoritma KMP atau Boyer – Moore.

Setelah beberapa eksperimen yang dilakukan, dapat diketahui bahwa pencarian chat dan message yang dilakukan pada line menggunakan *exact matching* dan dimungkinkan dengan teknik Boyer – Moore ataupun KMP. Namun, fitur pencarian message yang dilakukan dari dalam *chatroom* tidak menggunakan *exact string matching*. Fitur pencarian message dari dalam *chatroom* dapat dilakukan dengan pertama – tama masuk ke dalam *chatroom* dan menekan tombol "search" dengan lambang kaca pembesar. Berikut illustrasi dari pencarian fitur di dalam *chatroom.*



Gambar 9 Fitur pencarian message di dalam *chatroom*
Sumber : Dokumen Pribadi



Gambar 10 Hasil pencarian message di dalam *chatroom*
Sumber : Dokumen Pribadi

Dari gambar 9, fitur pencarian message di dalam *chatroom* dapat dijalankan dengan menekan tombol search di bagian atas chat. Hasil pengamatan dari fitur pencarian message di dalam *chatroom* dapat dilihat pada gambar 10.

Pada gambar 10, dilakukan pencarian terhadap pattern ”strategi algoritma” namun tidak seperti pada fitur pencarian chat dan message sebelumnya, pencarian tidak dilakukan secara exact matching. Terlihat pada gambar, pattern strategi algoritma mendapatkan pasangan pada kalimat “strategi dalam algoritma sangatlah berguna” padahal pattern yang dicari adalah “strategi algoritma”. Hasil ini menunjukkan bahwa fitur pencarian message di dalam *chatroom* tidak menggunakan exact matching.

Apabila bukan exact matching yang digunakan maka pencarian terhadap pattern tersebut dimungkinkan dengan penggunaan regular expression ataupun pencarian secara exact dengan melakukan exact matching terhadap kata bukan kalimat sehingga algoritma KMP ataupun algoritma Boyer – Moore dapat digunakan.

IV. Kesimpulan

Dari hasil eksperimen dalam penggunaan fitur pencarian chat dan message dan fitur pencarian message di dalam *chatroom* dapat dipastikan bahwa kedua fitur tersebut melakukan pencocokan string untuk menemukan chat atau message sesuai dengan pattern yang diinginkan oleh pengguna. Pencarian chat dan message di dalam Line menggunakan *exact matching* yang memungkinkan penggunaan algoritma KMP ataupun Boyer – Moore.

Berbeda dengan fitur pencarian chat dan message, fitur pencarian message di dalam *chatroom* tidak menggunakan *exact matching*. Pencarian dimungkinkan dengan menggunakan regular expression ataupun dengan melakukan *exact matching* sesuai dengan kata yang dipisahkan oleh spasi bukan dalam satu kalimat penuh.

Ucapan Terima Kasih

Penulis ingin mengucapkan terima kasih pertama – tama kepada Tuhan Yang Maha Esa atas berkat dan penyertaan-Nya, penulis dapat menyelesaikan makalah ini. Penulis juga berterima kasih kepada orang tua yang telah memberikan bantuan moral serta doa – doa yang diberikan sehingga penulis dapat terus berusaha menyelesaikan makalah ini. Tak lupa penulis mengucapkan terima kasih kepada Dr. Masayu Leylia Khodra, S.T., M.T., Dr. Ir. Rinaldi Munir, M.T., dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc. selaku dosen mata kuliah Strategi Algoritma yang telah membimbing dan mengajarkan materi yang diperlukan untuk menyelesaikan makalah ini dan juga kepada teman – teman yang telah membantu memberikan motivasi dan ide.

References

[1] https://www.regular-expressions.info/, Diakses pada 13 Mei 2018 Pukul 23.43

[2] Slide Pencocokan String (revisi 2018), Kuliah Strategi Algoritma, 2018.

[3] Slide Pencocokan String dengan Regular Expression (Regex), Kuliah Strategi Algoritma, 2018

[4] https://www.regexpal.com/ , Diakses pada 14 Mei 2018 Pukul 03.20

[5] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2018.

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, 14 Mei 2018

Christian Kevin Saputra - 13516073