Optimasi String Matching Dengan Menerapkan Regular Expression Pada Java

Rani Purbaningtyas
STMIK Dipanegara Makassar
Jl. Perintis Kemerdekaan KM. 9 Makassar
e-mail : fakgous@yahoo.com

Abstrak

Algoritma string matching semakin berkembang dan masing-masing algoritma tersebut memiliki keunggulan. Salah satu metode string matching yang ada yaitu regular expression. Penelitian ini mencoba untuk melihat sejauh mana optimasi pencocokan string mengunakan metode regular expression dengan menerapkan class-class yang dalam Java library.Melalui penelitian ini juga dapat diketahui tingkat keakuratan pencocokan pattern dalam file teks.

Kata kunci*: pencocokan string, regular expression*

Abstract

String matching algorithm is growing and each algorithm has own advantages. One existing method of string matching is regular expression. This paper tries to see the extent to which optimization method using regular expression classes that implements the Java library.Toward this paper may also be known level of accuracy of the matching pattern in a text file.

Keywords*: string matching, regular expression*

1. Pendahuluan

Pada beberapa kasus pengembangan sistem, metode-metode pencocokan teks banyak dibutuhkan. Metode-metode ini lebih dikenal dengan istilah string matching[1]. Diantaranya digunakan untuk mendeteksi adanya gangguan (intrusion) pada jaringan [5], mengkoreksi kesalahan pada SMS Gateway [4] pencarian dan menampilkan struktur kimia [2] dan masih banyak lagi.

Metode string matching sendiri dapat didefinisikan sebagai metode pencocokan kata atau pola kalimat (untuk selanjutnya disebut dengan pattern) dalam sebuah teks. Menurut Charras dan Lecroq [1], secara umum metode pencocokan pattern memiliki pola pencocokan dari kiri ke kanan, dari kanan ke kiri, dan gabungan keduanya. Algoritma string matching yang menerapkan pola pencarian dari kiri ke kanan diantaranya ada algoritma Karp-Rabin, Shift Or, Morris-Pratt, Knutt-Morris-Pratt, Simon, Galil-Giancarlo, Apostolico-Crochemore, Not So Naïve, dan Forward Dawg Matching. Algoritma string matching yang menerapkan pola pencarian dari kanan ke kiri diantaranya ada Collusi, Boyer-Moore, Turbo-BM, Apostolico-Giancarlo, Reverse Collusi, Horspool, Quick Search, Tuned Boyer-Moore, Zhu-Takaoka, Berry-Ravindran, Reverse Factor, Turbo Reverse Factor. Sedangkan algoritma string matching yang merupakan gabungan dari keduanya dan memiliki pola tertentu diantaranya adalah Smith, Raita, Backward Oracle Matching, Galil-Seiferas, Skip Search, KmpSkip Search, dan Alpha Skip Search. Setiap metode string matching tersebut memiliki keunggulan masing-masing.

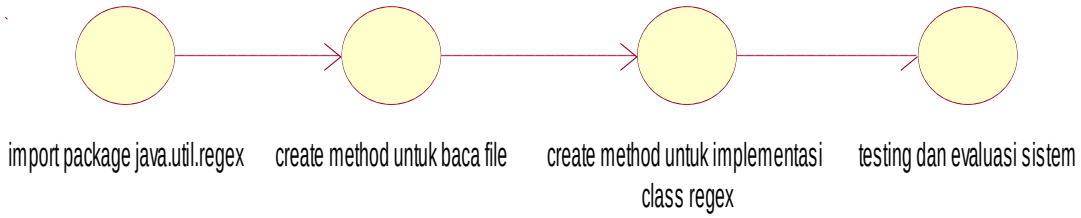
Selain algoritma string matching diatas, ada juga metode regular expression. Regular expression merupakan istilah yang digunakan untuk kodifikasi metode pencarian yang ditemukan oleh ahli ilmu Matematika Amerika Stephen Kleene. Secara singkat, dapat dijabarkan bahwa regular expression merupakan cara cepat untuk pencarian pattern dalam sebuah teks. Regular expressions dapat digunakan untuk mencari pattern tertentu dalam teks dan jika sudah ditemukan, pattern dapat dimodifikasi dengan berbagai cara.

Bahasa pemrograman Java merupakan bahasa pemrograman yang perkembangannya sangat pesat. Hal ini dikarenakan Java memiliki library yang cukup lengkap. Khusus untuk algoritma regular expression, Java memiliki class-class library yang ada dalam package java.util.regex.* yaitu class Pattern

dan class Matcher. Penelitian ini mencoba untuk mengeksplorasi lebih jauh optimasi string matching menggunakan metode reguler expression khususnya pada bahasa pemrograman Java. Sekaligus melalui penelitian ini juga ingin diketahui sejauh mana tingkat keakuratan metode regular expression yang ada dalam Java library.

2. Metode Penelitian

Metode penelitian yang digunakan menggunakan model *prototyping*. Tahapan pengembangan sistemnya sendiri dapat digambarkan dalam *use case* diagram berikut ini :



Gambar 1. Tahapan pengembangan sistem

Tahap awal yang dilakukan adalah melakukan proses import untuk Java *library* yang mendukung algoritma *regular expression*. Selanjutnya dilakukan proses pembuatan *method* yang akan digunakan untuk membaca file teks. Selain itu juga dibutuhkan *method* yang berisi implementasi *method-method* yang ada dalam *package java.util.regex*. Tahap terakhir yaitu testing dan evaluasi sistem. Metode testing yang digunakan yaitu *acceptance testing* yang menitikberatkan pada kategori *performance* sistem. Dari hasil pengujian ini nantinya sistem akan dievaluasi. Jika sistem banyak melakukan kesalahan dalam pencarian *pattern*, maka sistem perlu dikembangkan ulang.

3. Hasil dan Analisis

Berdasarkan tahapan pengembangan sistem diatas, maka yang perlu dilakukan pertama kali adalah melakukan proses import untuk Java *library* yang mendukung algoritma *regular expression*. Hal ini dilakukan dengan cara mengimport *package* java.util.regex.*.

Tahap kedua yaitu membuat *method* yang akan digunakan untuk membaca file teks. *Method* yang dibuat adalah sebagi berikut :

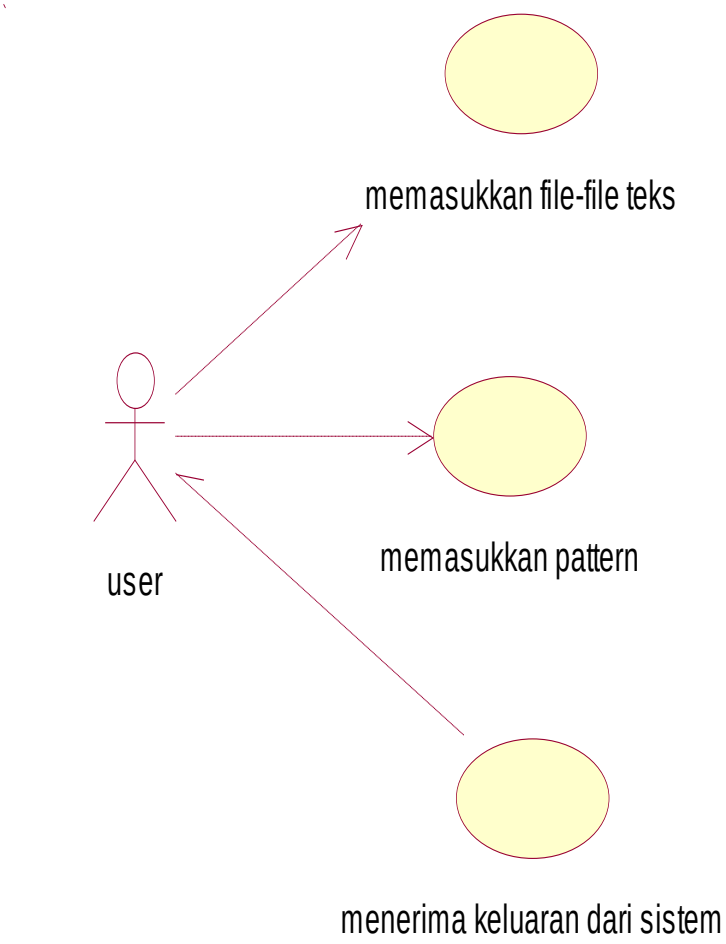
```
public static CharSequence fromFile(String filename) throws IOException {
    FileInputStream fis = new FileInputStream(filename);
    FileChannel fc = fis.getChannel();
    ByteBuffer bbuf = fc.map(FileChannel.MapMode.READ_ONLY, 0,
    (int)fc.size());
  CharBuffer cbuf =
  Charset.forName("8859_1").newDecoder().decode(bbuf);
  return cbuf;
}
```

Tahap selanjutnya yaitu membuat *method* yang berisi implementasi *method-method* yang ada dalam *package* java.util.regex.*. *Method* yang dibuat adalah sebagai berikut :

```
public static void AppRegex(String kontrol, String filePola) {
    static boolean status;
  try {
    Pattern pattern = Pattern.compile(kontrol);
    Matcher matcher = pattern.matcher(fromFile(filePola));
    if (matcher.find()) {
      System.out.println("Pattern ditemukan");
      status=true;
    } else{
      System.out.println("Pattern tidak ditemukan");
      status=false;
```

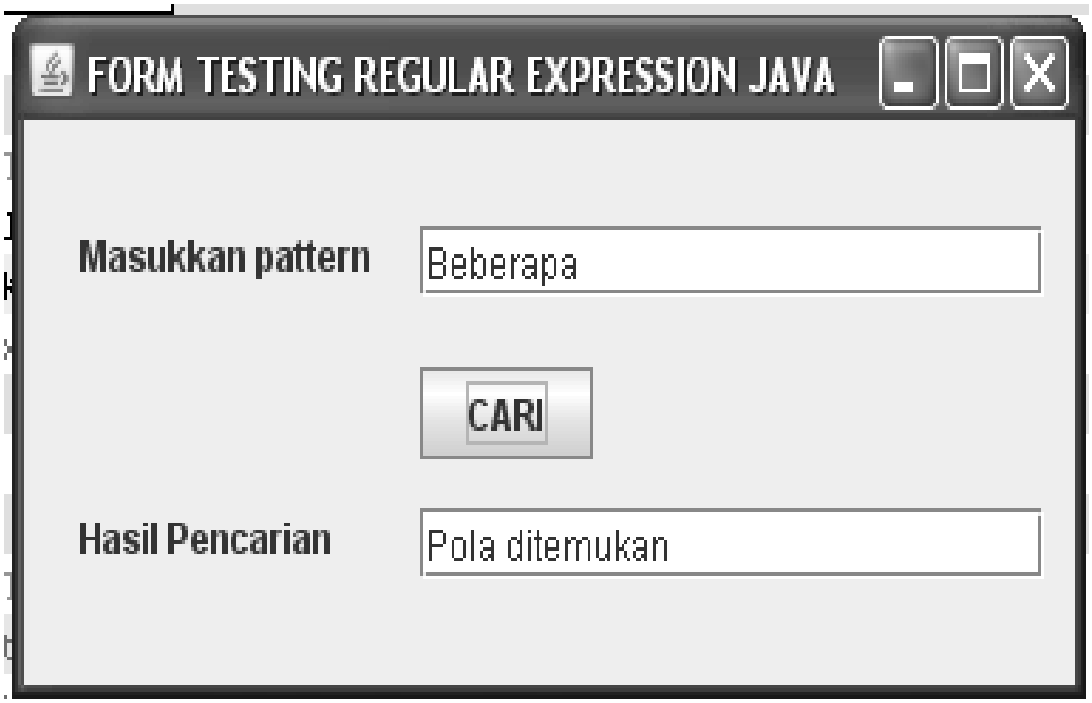
```
        }
      } catch (Exception e) {
        System.out.println("Error pada pencarian pattern "+e.toString());
      }
    }
```

Tahap terakhir yaitu testing dan evaluasi sistem. Tahapan uji coba sistem oleh *user* adalah sebagai berikut :

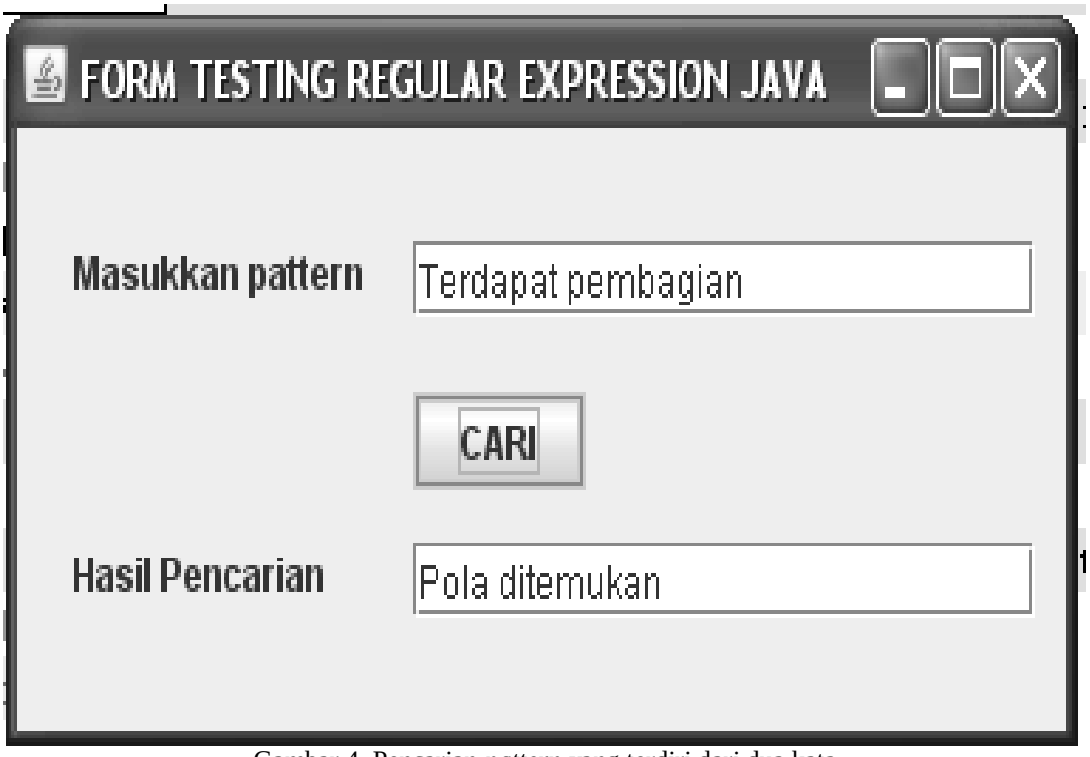


Gambar 2. Tahap uji coba sistem

User diminta untuk memasukkan file-file teks kedalam sistem. Kemudian melalui layar GUI, user diminta untuk memasukkan kata atau kalimat yang akan dijadikan *pattern.* Sistem akan melakukan proses *string matching*. Sistem akan mencari *pattern* di setiap file teks yang ada dalam sistem. Sistem akan menampilkan pesan apakan *pattern* yang dicari ditemukan atau tidak.



Gambar 3. Pencarian *pattern* yang terdiri dari satu kata



Gambar 4. Pencarian *pattern* yang terdiri dari dua kata

Gambar 5. Pencarian *pattern* yang terdiri dari kalimat lengkap

Pada tahap uji coba sistem ini akan dilakukan pengujian terhadap *performance* sistem. Akan dilihat sejauh mana ketepatan sistem dalam melakukan proses pencarian suatu *pattern* dalam file teks. Hasil uji coba *user* terhadap sistem adalah sebagai berikut :

Tabel 1. Hasil uji coba berdasarkan kategori *performance* sistem

PATTERN	JUMLAH FILE TEKS	JUMLAH FILE TEKS YANG TERDAPAT DITEMUKAN	JUMLAH FILE TEKS YANG TIDAK TERDAPAT PATTERN	KETEPATAN PENCARIAN
Beberapa	20	5	15	100%
Ini	20	20	0	100%
Terdapat anak	20	4	16	100%
Ada kesempatan	20	2	18	100%
Terdapat pembagian	20	1	19	100%
Beberapa hari ini	20	1	19	100%
Semua pergi ke sekolah	20	3	17	100%
Pada hari Minggu sekolah libur	20	0	20	100%

Dari hasil pengujian sistem diatas dapat dilihat bagaimana *performance* sistem. Sistem dapat tepat mencari *pattern* yang terdiri dari satu kata (100%). Sistem dapat tepat mencari *pattern* yang terdiri lebih dari satu kata (100%). Sistem juga dapat tepat mencari *pattern* yang berupa kalimat lengkap didalam file teks (100%).

4. Kesimpulan

Metode string matching sangat beragam. Dan setiap algoritma string matching memiliki keunggulan tersendiri. Salah satu algoritma string matching yaitu regular expression. Java sebagai salah satu bahasa pemrograman yang memiliki library komplek pun juga menerapkan metode ini. Dari hasil pengujian, metode regular expression yang dimiliki Java ini memiliki keakuratan 100% dalam pencarian pattern dalam file teks, baik pattern yang berupa satu kata, beberapa kata maupun kalimat lengkap.

Daftar Pustaka

[1] Charras C, Lecroq T. Handbook of Exact String-Matching Algorithms. Oxford University Press. 2001.

[2] Klaib A.F, Zainol Z, Ahamed N.H, Ahmad R, Hussin W. Application of Exact String Matching Algorithms towards SMILES Representation of Chemical Structure. World Academy of Science, Engineering and Technology. 2007; 34: p. 36-40.

[3] Kumara G.H. Visualisasi Beberapa Algoritma Pencocokan String Dengan Java. Sekolah Teknik Elektro Informatika ITB Bandung. 2007

[4] R.A Dewanto, Aradea. Aplikasi SMS Gateway Dengan Koreksi Kesalahan Menggunakan Fuzzy String Matching. Proceeding pada Seminar Nasional Aplikasi Teknologi Informasi (SNATI) 2007.Yogyakarta. 2007; C35-C38.

[5] Tuck N, Sherwood T, Calder B, Varghese G. Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection. IEEE Infocom. 2004.