Aplikasi Algoritma FJS dalam Applet Pencarian Kata pada Web Browser

Yuandra Ismiraldi
13505069

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha no.10, Bandung, Indonesia
if15069@students.if.itb.ac.id

ABSTRAK

Di dalam suatu *web browser* salah satu fitur yang sering digunakan adalah fitur pencarian kata. Fitur/program ini memungkinkan untuk mencari dan menemukan secara langsung kata yang ingin kita cari pada halaman web tersebut. Fitur ini dapat merupakan fitur bawaan pada *web browser* atau dapat juga merupakan program tambahan pada browser dengan menggunakan *applet Java*. Pencarian kata ini menggunakan algoritma *string matching*. Salah satu algoritma dalam *string matching* adalah algoritma FJS (Franek, Jennings, Smyth) yang merupakan penggabungan dari algoritma KMP (Knuth-Morris-Pratt), algoritma BM (Boyer-Moore), dan algoritma Sunday.

Kata kunci: *FJS, string matching, pencarian kata, web browser, applet*

1. PENDAHULUAN

Di zaman internet yang semakin mendunia ini, kita sering menggunakan internet untuk mencari informasi. Informasi yang kita cari biasanya tersimpan dalam halaman suatu situs web. Permasalahan muncul saat halaman yang kita buka tersebut berukuran sangat besar, sehingga saat kita membuka halaman web itu kita sulit untuk menemukan informasi spesifik yang kita ingin temukan.

Untuk melewati masalah ini, pada *web browser* yang kita pergunakan untuk membuka halaman web itu sering ditemukan sebuah fitur untuk melakukan pencarian kata pada halaman web yang sedang dibuka sekarang. Fitur ini memungkinkan kita untuk cepat mencari dan menemukan informasi yang kita inginkan dalam suatu halaman web.

Fitur pencarian kata tersebut dapat berupa fungsi bawaan dari web browser yang kita gunakan atau dapat berupa program tambahan (plugin/applet) yang ditempelkan di *web browser*. Applet tersebut adalah sebuah program terpisah yang dapat dipanggil dan dijalankan dari *web browser*. *Applet* yang dibahas dalam makalah ini adalah *Applet Java* yang akan diimplemtasikan dengan menggunakan java SDK 1.6

Fitur pencarian kata tersebut di dalamnya menggunakan algoritma *string matching*. Pencarian string ini dalam langkah kerjanya terfokus dalam menemukan kemunculan rangkaian karakter dengan pola tertentu yang terdiri dari n karakter (disebut *pattern*) dalam suatu teks dengan panjang m. Adapun algoritma *string matching* ada banyak sekali jenisnya, dan yang akan digunakan dalam aplikasi pencarian kata ini adalah algoritma FJS (Frannek, Jennings, Smyth), yang merupakan gabungan dari beberapa algoritma pencarian string yang seperti KMP (Knuth-Morris-Pratt), algoritma BM (Boyer-Moore), dan algoritma Sunday.

2. METODE

Metode yang digunakan dalam aplikasi applet pencarian kata pada *web browser* ini terdiri dari 2, yaitu :

1. Algoritma string matching, yaitu algoritma FJS (Frannek, Jennings, Smyth), dan
2. Pengaplikasian algoritma tersebut dalam sebuah applet dengan bahasa Java yang mengambil teks isi dari halaman web tersebut.

2.1 Algoritma FJS (Franek, Jennings, Smyth)

Alogritma pencocokan string sudah mengalami pengembangan sejak pertama kali digunakan dengan menggunakan algoritma Brute Force. Munculnya algoritma seperti Boyer-Morre dan Knuth-Morris-Pratt membuat algoritma pencocokan string ini semakin mengalami optimasi dan makin mecapai efisiensi dan efektivitas yang diharapkan. Akan tetapi kedua algoritma ini dirasakan belum cukup sebab untuk skala besar pencocokan string ini masih membutuhkan pemakaian sumber daya komputer yang cukup besar dan waktu yang cukup lama apabila *worst-case scenario* (saat string yang

dicari tidak ditemukan sama sekali pada teks dan program harus mengakses sampai akhir teks) pada pencocokan string itu terjadi.

Frantisek Franek, Christopher G. Jennings dan W.F. Smyth membuat sebuah algoritma pencocokan string yang menggabungkan metode algoritma Boyer Moore, Knuth-Morris-Pratt, dan Sunday. Algoritma yang diciptakan oleh mereka bertiga ini selanjutnya dikenal dengan nama algoritma FJS.

Algoritma FJS dibuat dengan fokus terhadap dua hal, yaitu :

1. Mengurangi jumlah perbandingan karakter yang diperlukan dalam pencocokan string saat terjadi *worst-case scenario*
2. Mengurangi waktu komputasi yang diperlukan saat terjadi *worst-case scenario*

Secara umum, cara kerjaalgoritma FJS adalah :

1. Pertama tama string yang dicari (kita sebut dengan array of char p[]) kita taruh di teks (kita sebut dengan array of char x[]) sehingga karakter pertama pada string yang dicari terletak pada posisi yang pertama pada karakter pertama teks, p[1] = x[1].

2. Kemudian gunakan algoritma Boyer Moore dan Knuth Morris Pratt untuk mencocokan string tersebut dengan teks. Algoritma FJS pada awalnya akan membuat sebuah tabel fungsi pergeseran string yang dicari yang diambil dari algoritma Knuth Morris Pratt. Pola string p[i] kemudian akan dicek dari kiri kanan dengan x[i], x[i+1], terus menerus sampai pola string p ketemu semua atau terjadi ketidakcocokan terjadi. Jika ketidakcocokan tidak terjadi di awal pencocokan string dengan algoritma Knuth Morris Pratt (j>1), maka terjadi ketidakcocokan parsial sehingga akan digunakan algoritma Knuth Morris Pratt untuk mengecek teks selanjutnya.

3. Jika ketidakcocokan langsung terjadi di awal pencocokan string dengan Knuth Morris Pratt (j=0), maka akan dilakukan algoritma Sunday untuk melakukan pergeseran posisi untuk pengecekan teks. Apabila posisi tersebut telah didapat, maka kemudian akan dilakukan algoritma Knuth Morris Pratt untuk mengecek teks selanjutnya.

4. Pergeseran posisi pengecekan teks dan pengecekan teks tersebut akan terus berlangsung sampai posisi pengecekan teks sudah bergeser melewati akhir teks, dimana pada saat tersebut teks sudah tidak mungkin dicek kembali.

Algoritma FJS ini akan menggunakan dua array dalam penggunaannya, yaitu array yang memuat fungsi pinggiran string yang dicari yang merupakan bagian dari algoritma Knuth Morris Pratt serta array pergeseran posisi yang merupakan bagian dari algoritma Sunday.

Pseudo code algoritma FJS adalah sebagai berikut :

```
Procedure cariString(input arrayofChar
p,arrayofChar x; output int i)
//mencari semua kemunculan p[1..m] di
x[1..n]

Kamus :
integer i,i1,j,m,m1,n
array KMP : B
array Sunday : S

Algoritma

begin

if m<1 then
     Return j  ← 1;i ← 1; i1 ← m; m1
← m-1;
while i1 <= n do
     if j <- 1 then
     //bila tidak ada kecocokan parsial
dengan p, lakukan pergeseran Sunday,
sehingga      mengembalikan      posisi
berikutnya i1 sehingga x[i1] = p[m]
     pergeseranSunday(i1);

     //inisialisiasi   lagi   KMP   untuk
p[1..m-1]
     j ← 1; i ← i1- m1
     cekKMP(m1;j,i);

else

     //lanjutkan   pencocokan   KMP   untuk
p[1..m]
     cekKMP(m;j,i);

     //inisialisasi   kembali   varibel
untuk pencocokan berikutnya
     j ← B[j] ; i1 ← i+m-j

end

Procedure cekKMP(input/Output integer :
m; input integer j,i)
//algoritma pengecekan KMP

kamus
-

algoritma
```

```
begin

while j<= m and x[i] = p[j] do
     I ← i+1; j ← j+1
if j>m then
     m ← i-m

end

Procedure pergeseranSunday(input/output
integer i1)
//merubah posisi i1 sesuai dengan
algoritma Sunday

kamus :
-

algoritma

begin

while x[i1] <> p[m] do
     i1 ← i1 + S[x[i1+1]]
     if i1 > n then
       break;

end
```

Kompleksitas waktu algoritma FJS adalah :

1. Saat kasus terbaik algoritma FJS akan melakukan [n/(m+1)] perbandingan karater sehingga kompleksitas waktu algoritma FJS untuk kasus terbaik adalah O(n/m).

   Kasus terbaik terjadi saat :
   1. p[m] tidak sesuai dengan karakter di x[m]
   2. karakter di x tidak ada di string p

2. Saat kasus terburuk (string tidak ditemukan sampai akhir teks), algoritma FJS akan melakukan sampai 3n-2m perbandingan karakter, sehingga kompleksitas waktu algoritma FJS untuk kasus terburuk adalah O(n+m).

Sebagai contoh misalkan kita ingin mencari kata “kita” pada string “ayo kita cari kelereng”

Tabel 1. Tabel Contoh Fungsi Pinggiran KMP dan Pergeseran Sunday pada Algoritma JFS

j	1	2	3	4
p[j]	k	i	t	a
B[j]	0	1	1	1
S[p[j]]	5	4	3	2

Dengan P[] sebagai teks “ayo kita cari kelereng” dan X[] sebagai string “kita”,

Tempatkan P[1] = X[1].
cek P[4] = X[4]
Ternyata a tidak cocok, lakukan pergeseran sehingga didapat posisi P[5]

```
ayo kita cari kelereng
   |
kita
```

Cek P[8] = X[4]
ternyata cocok, lanjut cocokkan dengan algoritma KMP
kata “kita” ketemu.
Selesai dicek, lakukan pergeseran selanjutnya sehingga P[9].

```
ayo kita cari kelereng
       |
    kita

ayo kita cari kelereng
    |  |
    kita

ayo kita cari kelereng
    || |
    kita

ayo kita cari kelereng
    ||||
    kita
```

cek P[12] = X[4], ternyata tidak cocok, lakukan pergeseran sehingga P[12]

```
ayo kita cari kelereng
          |
        kita
```

cek P[15] = X[4], ternyata tidak cocok, lakukan pergeseran sehingga P[17]

```
ayo kita cari kelereng
             |
           kita
```

cek P[20] = X[4], ternyata tidak cocok, lakukan pergeseran sehingga P melebihi panjang teks, program berhenti

```
ayo kita cari kelereng
                   |
                 kita
```

Hasilnya dilakukan 8 kali perbandingan dengan string "kita" ditemukan sebanyak 1 kali

2.2 Aplikasi dengan Java Applet

Bahasa Java yang telah dikembangkan oleh Sun Microsystem telah menjadi sebuah bahasa Open Source yang telah digunakan oleh banyak orang. Keunggulan Java terletak pada prinsipnya yaitu *"write once, run anywhere"*. Prinsip ini berarti bahwa seseorang dapat menulis sebuah program pada seuatu komputer dengan menggunakan bahasa Java dan setelah itu dia dapat dengan mudah menggunakan program yang telah dibuatnya tersebut di komputer-komputer lainnya sesukanya dengan catatan bahwa di komputer-komputer lainnya tersebut sudah terinstalasi dengan *Java Runtime Enviroment (JRE),* sebuah aplikasi yang memungkinkan suatu komputer menjalankan sebuah aplikasi yang dibuat dengan bahasa Java.

Kelebihannya, program Java tersebut dapat dibungkus dalam sebuah program kecil Java yang dapat berdiri sendiri yang disebut dengan Applet. Program Applet ini kemudian dapat disertakan dan dijalankan dengan web browser seperti Mozilla Firefox ataupun Internet Explorer. Applet Java juga berukuran sangat kecil dan mudah dibawa kemana-mana atau dapat pula diintegrasi langsung sebagai plug-in dari web browser tersebut.

Program Applet Java ini akan melakukan 3 hal, yaitu :
1. Mengambil isi teks halaman web
2. Menggunakan algoritma JFS untuk mencarai posisi kemunculan string yang dicari pada teks
3. Memfokuskan layar *web browser* ke posisi teks tersebut

Pseudo code Applet java untuk pencarian kata

```
Program CariTeks

kamus:
int I // posisi teks
array of char X //string yang dicari
array of char P // teks yang akan
dilakukan pencarian
char c

algoritma:

begin

//ambil teks dari web browser
//teks adalah diantara tag <body>
</body>
//teks dimasukkan ke array of char P
 AmbilTeks(P)

//ambil input user, masukkan ke array
of char X
AmbilInput(X)

//panggil algoritma JFS
cariString(P,X,i)

//fokuskan layar ke posisi i
FokusLayar(i)

end.
```

IV. KESIMPULAN

Algoritma Frannek-Jennings-Smyth mampu melakukan pencarian string secara optimal dan efisien karena dapat menggabungkan antara algoritma Boyer Moore, Knuth-Morris-Pratt, dan Sunday.
Algoritma ini mampu untuk dibuat dalam bentuk aplikasi yang kecil dan efisien dan dapat berfungsi sebagai sebuah *applet* pencarian kata untuk dijadikan sebagai *plug-in* program tertentu.

REFERENSI

[1] http://java.sun.com/javase/6/, diakses pada tanggal 19 Mei 2007 pukul 12.00

[2] http://www-igm.univ-mlv.fr/%7Elecroq/string/node1.html, diakses pada tanggal 21 Mei 2007 pukul 19.55

[3] http://www.sfu.ca/%7Ecjenning/index.html, diakses pada tanggal 21 Mei 2007 pukul 20.00

[4] Frantisek Franek, Christopher G. Jennings, and W. F. Smyth. *A Simple Fast Hybrid Pattern-Matching Algorithm.* 2006.

[5] Rinaldi Munir. *Diktat Kuliah Strategi Algoritmik*. Program Studi Teknik Informatika ITB : 2007.