Aplikasi Algoritma Pencarian String Boyer-Moore pada Pencocokan DNA

Arie Minandar[1], Andri Tanoto[2], Davis Tanadi[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if13074@students.if.itb.ac.id[1], if13078@students.if.itb.ac.id[2], if13110@students.if.itb.ac.id[3]

Abstrak

Rangkaian DNA merupakan hal terpenting untuk semua jenis kehidupan di alam semesta ini. Karena itu, segala upaya untuk memahami DNA menjadi snagat penting, tidak terkecuali upaya pencocokan rangkaian DNA, terutama untuk kemajuan bioinformatika. Makalah ini mencoba untuk menerangkan mengenai algoritma pencarian string Boyer-Moore dan penerapannya pada pencocokan DNA. Algoritma Boyer-Moore merupakan algoritma pencarian string yang dianggap paling mangkus saat ini. Algoritma Boyer-Moore melakukan pencocokan string dari kanan ke kiri.

***Kata kunci**:boyer moore algorithm, rangkaian DNA, pencocokan DNA*

1. Pendahuluan

Rangkaian DNA, yang mengandung informasi kehidupan untuk setiap organisme, dapat dianggap sebagai rangkaian string yang merupakan kombinasi dari empat karakter berikut, {A,T,G,C}. Manusia memiliki rangkaian DNA yang sangat panjang, hal ini merupakan salah satu ciri manusia sebagai mamalia. Pencarian pola pada rangkaian DNA merupakan langkah pertama dan penting pada studi yang berhubungan dengan DNA sebagai penyimpan informasi kehidupan.

Selain DNA manusia, para ilmuwan juga tertarik untuk mempelajari rangkaian DNA pada spesies lain. Ketertarikan ilmuwan pada DNA telah menimbulkan banyak sekali riset untuk menentukan rangkaian DNA pada berbagai spesies di alam ini. Salah satu cara untuk mencocokkan DNA adalah dengan menggunakan algoritma Boyer-Moore yang sering dipakai juga untuk teknik pencocokan string pada bidang komputer.

Tujuan pembuatan makalah ini adalah :

1. Mempelajari metode pencocokan string dengan menggunakan algoritma Boyer-Moore.
2. Menerapkan algoritma Boyer-Moore untuk melakukan pencocokan DNA.

2. Algoritma Boyer-Moore

Algoritma Boyer-Moore dianggap sebagai algoritma pencocokan string yang paling mangkus dalam berbagai aplikasi. Algoritma ini sering diimplementasikan dalam berbagai teks editor (misalnya : Microsoft Word) untuk fungsi "*Find and Replace*".

Algoritma Boyer-Moore melakukan pencocokan karakter dimulai dari kanan ke kiri. Karakter paling kanan pada pola merupakan karakter pertama yang akan dicocokkan dengan teks. Algoritma ini mempunyai dua fase, yaitu fase *preprocessing* dan fase pencarian. Pada fase *preprocessing* terdapat dua buah fungsi untuk menggeser pola ke arah kanan. Kedua fungsi ini disebut *good-suffix-shift* dan *bad-character-shift*. Fungsi *good-suffix-shift* disimpan ke dalam sebuah tabel *bmGs* berukuran *m*+1. Sedangkan fungsi *bad-character-shift* disimpan ke dalam sebuah tabel *bmBc* yang berukuran *n*.

Pembentukan tabel *bmBc* dan *bmGs* mempunyai kompleksitas waktu ***O****(m+n)* dan kompleksitas ruang ***O****(m+n)*. Sedangkan kompleksitas waktu untuk fase pencarian adalah ***O****(mn)* . Kasus terbaik untuk algoritma ini mempunyai kompleksitas waktu ***O****(n/m)* sedangkan pada kasus terburuk akan terjadi sebanyak *3n* kali perbandingan untuk pencarian dengan pola yang tidak berulang (periodik).

2.1 Pseudo-code Algoritma Boyer-Moore

```
procedure preBmBc(input x: array of
char, input m:integer, input/output
bmBc: array of integer)

Deklarasi
   i:integer

Algoritma
   for (i=0; i<ASIZE; ++i)
      bmBc[i] ← m
```

```
   for (i=0; i < m-1; ++i)
      bmBc[x[i]] ← m-i-1


procedure suffixes(input x: array of
char, input m: integer, input/output
suff: array of integer)

Deklarasi
   f, g, i : integer

Algoritma
   suff[m-1] ← m;
   g ← m-1

   for (i = m-2; i >= 0; --i)
      if (i>g and suff[i+m-1-f]<i-g)
         suff[i] ← suff[i+m-1-f]
      else {
        if (i<g)
           g←i
        f = i
        while(g> 0 and x[g]=x[g+m-1-f])
            --g
        suff[i] ← f-g


procedure preBmGs(input x: array of
char, input m:integer, input/output
bmGs: array of integer)

Deklarasi
   i, j : integer
   suff : array [0..XSIZE] of integer
Algoritma
   suffixes(x, m, suff)

   for (i = 0; i < m; ++i)
      bmGs[i] ← m
   j ← 0
   for (i = m-1; i >= -1; --i)
      if (i = -1 or suff[i] = i+1)
         for (j = 0; j < m-1-i; ++j)
            if (bmGs[j] = m)
               bmGs[j] ← m-1-i
   for (i = 0; i <= m-2; ++i)
      bmGs[m-1-suff[i]] ← m-1-i



procedure BM(input x: array of char,
input m:integer, input y: array of
char, input n: integer)

Deklarasi
   i, j : integer
   bmGs : array [0..XSIZE] of integer
   bmBc : array [0..ASIZE] of integer

Algoritma
   /* Preprocessing */
   preBmGs(x, m, bmGs)
   preBmBc(x, m, bmBc)

   /* Searching */
   j = 0
   while (j <= n-m)
      for (i = m-1; i >= 0 and x[i]=
y[i+j]; --i)
      if (i < 0)
         OUTPUT(j)
         j ← j + bmGs[0]
      else
         j ← j + MAX(bmGs[i],
bmBc[y[i+j]]-m+1+i)
```

2.2 Penerapan Boyer-Moore pada Pencocokan DNA

Seperti yang sudah disebutkan sebelumnya bahwa DNA dapat dianggap sebagai rangkaian string, maka pencocokan DNA tidak lain merupakan pencocokan string. Berikut ini contoh pencocokan DNA dengan Boyer-Moore:

Suatu rangkaian DNA (R)

GCATCGCAGAGAGTATACAGTACG

akan dicocokan dengan potongan atau pola DNA (P)

GCAGAGAG

Tahapan-tahapan yang terjadi :

Tahap 1:

GCATCGC**A**GAGAGTATACAGTACG
GCAGAGA**G**
 GCAGAGAG

(*bmGs*[7]=*bmBc*[A]-8+8)

Simbol terakhir pada pola P, yaitu G, dan simbol yang sejajar dengannya, yaitu simbol A dibandingkan, karena kedua simbol tersebut berbeda (*mismatch*), maka pola P digeser sedemikian sehingga simbol A paling kanan pada pola P sejajar dengan simbol A.

Tahap 2:

GCATCG**CAG**AGAGTATACAGTACG
 GCAGA**GAG**
       GCAGAGAG

(*bmGs*[5]=*bmBc*[C]-8+6)

Simbol terakhir pada pola P dan simbol yang sejajar dengannya dibandingkan (simbol G dan G). Karena kedua simbol sama, maka dilakukan pembandingan terhadap simbol sebelumnya (simbol A dan A). Hal serupa dilakukan lagi, karena kedua simbol masih sama. Namun karena simbol C dan G tidak cocok, maka pola P digeser sedemikian sehingga simbol C paling kanan pada pola P sejajar dengan simbol C.

Tahap 3:

GCATC**GCAGAGAG**TATACAGTACG
GCAGAGAG
GCAGAGAG

(*bmGs*[0])

Simbol terakhir pada pola P dan simbol yang sejajar dengannya dibandingkan. Namun, karena kedua simbol sama, maka dilakukan pembandingan terhadap simbol sebelumnya, begitu seterusnya, sampai pada akhirnya semua simbol pada pola P sudah dibandingkan. Pada tahap ini pola pada DNA R sudah ditemukan. Karena R belum habis, maka pencocokan dilanjutkan. Pola P digeser sedemikian sehingga simbol paling kanannya serupa dengan simbol pada rangkaian R.

Tahap 4:

GCATCGCAGAGAGTATA**CAG**TACG
GCAGA**GAG**
GCAGAGAG

(*bmGs*[5]=*bmBc*[C]-8+6)

Simbol terakhir pada pola P dan simbol yang sejajar dengannya dibandingkan. Namun karena kedua simbol sama, dilakukan pembandingan terhadap simbol sebelumnya. Simbol C dan G dibandingkan, karena tidak sama, maka pola P digeser sehingga simbol C paling kanan pada pola sejajar dengan simbol C.

Tahap 5:

GCATCGCAGAGAGTATACAGTA**CG**
GCAGAG**AG**
GCAGAGAG

(*bmGs*[6])
Simbol G yang merupakan simbol terakhir pada pola P dibandingkan dengan simbol yang sejajar dengannya. Karena kedua simbol sama, maka pembandingan dilakukan terhadap simbol sebelumnya, yaitu simbol C dan A. Namun, kedua simbol tersebut berbeda, sehingga simbol C paling kanan pada pola P sejajar dengan simbol C. Akan tetapi, karena simbol paling kanan pada pola P tidak memiliki pasangan yang sejajar untuk dibandingkan, maka pencocokan berhenti.

Keterangan: bmGs : boyer-moore Good suffix
bmBc : boyer-moore Bad character

3. Kesimpulan dan Saran

Kesimpulan dari topik makalah ini adalah algoritma Boyer-Moore menggunakan dua buah metode heurestic yang berbeda untuk menentukan pergeseran maksimum yang mungkin dilakukan, yaitu *bad character* dan *good suffix* sehingga pencocokan string akan lebih efisien. Metode Boyer-Moore dapat digunakan sebagai salah satu alternatif untuk pencocokan rangkaian DNA.

Saran pengembangan untuk makalah ini adalah pada kenyataannya susunan suatu DNA sangat panjang oleh karena itu sebaiknya penelitian memakai mesin lebih dari satu untuk mencari suatu susunan DNA yang diinginkan.

4. Referensi

[1] Carras C., Lecroq T., Boyer-Moore Algorithm, http://www-igm.univ-mlv.fr/~lecroq/string/node14.html, diakses tanggal 18 Mei 2005 pukul 11.00

[2] Carras C., Lecroq T., Boyer-Moore Algorithm, http://www-igm.univ-mlv.fr/~lecroq/string/examples/exp14.html, diakses tanggal 18 Mei 2005 pukul 11.00

[3] Cheng Lok-Lam, Cheung D. W., Yiu Siu-Ming, Approximate String Matching in DNA Sequences, http://www.cs.hku.hk/~dcheung/publication/dasfaa2003_2.pdf, diakses tanggal 17 Mei 2005 pukul 21.00

[4] Lang H.W., String Matching Boyer-Moore Algorithm, http://www.iti.fh-flensburg.de/lang/algorithmen/pattern/bmen.htm, diakses tanggal 18 Mei 2005 pukul