Penerapan Algoritma Pencocokan String Knuth-Morris-Pratt Sebagai Algoritma Pencocokan DNA

Kukuh Nasrul Wicaksono

Departemen Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha No 10 Bandung Indonesia
e-mail: if15097@students.if.itb.ac.id

ABSTRAK

Di era modern ini, identifikasi seseorang bisa dilakukan dengan berbagai cara yang dahulu dianggap tidak mungkin. Selain melalui pencocokan sidik jari, pencocokan DNA (Deoxyribo Nucleic Acid) adalah salah satu cara yang paling ampuh untuk mengidentifikasi seseorang. Pencocokan DNA ini sangat mirip dengan konsep pencocokan string yang sering di pakai di bidang informatika. Bisa dibilang sama dengan pencocokan string karena pada pencocokan DNA juga mencoba untuk mencari kesamaan antara teks sample dengan pattern yang telah ada. Pada pencocokan DNA kita mencari kecocokan antara DNA pattern yang merupakan DNA dari orang yang ingin diidentifikasi dengan DNA sample yaitu DNA kerabat dekat orang yang akan diidentifikasi. DNA bisa digambarkan sebagai kumpulan gugus karbon dimana setiap gugus karbon dapat dianalogikan sebagai karakter, sehingga DNA itu sendiri bisa diibaratkan sebagai rangkaian karakter atau string. Karena kesamaan itu, maka algoritma yang biasa digunakan pada pencocokan string bisa juga dipakai sebagai algoritma untuk mencocokan DNA. Salah satu algoritma pencocokan string yang mangkus adalah algoritma Knuth-Morris-Pratt. Dalam makalah ini akan dijelaskan mengenai cara kerja algoritma Knuth-Morris-Pratt dalam pencocokan DNA serta kompleksitas waktu yang dimiliki algoritma tersebut

Kata kunci: Algoritma Knuth-Morris-Pratt, DNA, Fungsi pinggiran.

1. PENDAHULUAN

DNA merupakan suatu unit informasi kehidupan terkecil yang dimiliki oleh semua makhluk hidup dan diturunkan secara turun temurun. Semakin dekat kekerabatan seseorang maka semakin mirip DNA yang dimilikinya. DNA yang dimiliki tiap makhluk hidup ini adalah unik, dengan kata lain DNA seseorang tidak mungkin sama dengan orang lain. Karena keunikan itu maka pencocokan DNA menjadi salah satu senjata ampuh untuk mengidentifikasi seseorang meskipun sampel yang akan diidentifikasi hanya berupa sehelai rambut atau secuil kulit karena DNA seseorang tersebar di semua bagian tubuh dan memiliki pola yang sama. Karena hal tersebut, maka pencocokan DNA ini sering dipakai untuk identifikasi jasad seseorang yang telah hancur dan tidak bisa dikenali hanya dengan melihatnya.

Rangkaian DNA, yang mengandung informasi kehidupan unik untuk setiap makhluk hidup terdiri dari kumpulan gugus karbon dimana setiap gugus karbon dapat dianalogikan sebagai karakter, sehingga DNA itu sendiri dapat dianalogikan sebagai rangkaian karakter atau String. DNA terdiri dari empat jenis gugus karbon yaitu Adenin (A), Sitosin (S), Timin (T), dan Guanin (G). Dengan demikian maka DNA sama dengan string yang merupakan kombinasi dari 4 jenis karakter yaitu A, S, T, dan G.

2. METODE

Metode yang digunakan dalam pembuatan makalah ini adalah metode studi literatur.

3. ALGORITMA KNUTH-MORRIS-PRATT

Algoritma Knuth-Morris-Pratt merupakan salah satu algoritma yang sering digunakan untuk menyelesaikan masalah pencocokan string. Algoritma ini adalah penyempurnaan dari algoritma pencocokan string dengan menggunakan algoritma *brute force*. Pada algoritma *brute force*, setiap kali ditemukan ketidak cocokan *pattern* dengan teks, maka *pattern* akan digeser satu ke kanan. Sedangkan pada algoritma Knuth-Morris-Pratt, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma menggunakan informasi tersebut untuk membuat pergeseran ang lebih jauh, tidak hanya satu karakter seperti pada algoritma *brute force*. Dengan algoritma Knuth-Morris-Pratt ini, waktu pencarian dapat dikurangi secara signifikan. Algoritma Knuth-Morris-Pratt ini dikembangakan oleh D. E. Knuth, bersama-sama dengan J. H. Morris dan V. R. Pratt.

Dalam algoritma Knuth-Morris-Pratt ini kita akan menemui beberapa definisi yang nantinya akan digunakan dalam algoritma ini. Beberapa definisi tersebut akan dijelaskan dengan contoh berikut. Misal x = abacab maka awalan dari x adalah [], a, ab, aba, abac, abaca. Kemudian akhiran dari x adalah [], b, ab, cab, acab, bacab. Kemudian pinggiran dari x adalah [], ab. Pinggiran [] mempunyai panjang 0 dan ab mempunyai panjang 2.

3.1 Fungsi Pinggiran

Algoritma Knuth-Morris-Pratt melakukan proses awal atau *preproccesing* terhadap pattern P dengan menghitung fungsi pinggiran (dalam literatur lain menyebut fungsi *overlap*, fungsi *failure*, dsb) yang mengindikasikan pergeseran s terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum pencarian string. Dengan adanya fungsi pinggiran ini, dapat dicegah pergeseran yang ridak berguna seperti halnya pada algoritma *brute force*. Fungsi pinggiran hanya berganung pada kerakter-karekter di dalam pattern, dan bukan pada kerakter-karekter di dalam teks yang dicari. Oleh karena itu, kita dapat melakukan perhitungan fungsi awalah sebelum pencarian string dilakukan.

Fungsi pinggiran b(j) didefinisikan sebagai ukuran awalan terpanjang dari P yang merupakan akhiran dari P[1..j]. Sebagai contoh, tinjau pattern P = abacabd. Nilai F untuk setiap karakter di dalam P adalah sebagai berikut.

Tabel 1 Fungsi pinggiran untuk pattern abacabd

j	**1**	**2**	**3**	**4**	**5**	**6**
P(j)	a	b	c	a	b	d
B(j)	0	0	1	1	2	0

Di bawah ini adalah algoritma untuk menghitung fungsi pinggiran.

```
procedure HitungPinggiran
(input m : integer, P : array[1..m] of
char,output b : array[1..m] of integer)
{ Menghitung nilai b[1..m] untuk
pattern P[1..m] }

Deklarasi
  k,q : integer

Algoritma:
  b[1]←0
  q←2
  k←0
  for q←2 to m do
    while ((k > 0) and (P[q] ≠ P[k+1]))
do
      k←b[k]
    endwhile
    if P[q]=P[k+1] then
      k←k+1
    endif
    b[q]=k
  endfor
```

3.2 Fungsi Pembandingan String

Kemudian cara untuk melakukan pencocokan string dengan menggunakan algoritma Knuth-Morris-Pratt adalah sebagai berikut.

Misal kita akan mencocokan teks T = abcabcabd dan kita mempunya pattern P = abcabd.

```
            123456789
Teks    = abcabcabd
Pattern = abcabd
```

Mula-mula kita hitung fungsi pinggiran dari pattern P tersebut. Fungsi pinggiran P = abacabd tertera seperti tabel 1 di atas. Kemudian lakukan langkah-langkah berikut. Samakan ujung kiri pattern dengan ujung kiri teks. Karakter-karakter pada karakter 1-5 sama, tetapi pada posisi ke 6 tidak sama. Hal itu karena karakter ke 6 pada teks yaitu c tidak sama dengan karakter ke 6 pada pattern yaitu d. Untuk mencocokan kembali, kita harus menggeser pattern. Jika dalam *brute force* kita akan menggeser pattern 1 karakter ke kanan. Namun jika menggunakan algoritma Knuth-Morris-Pratt jumlah pergeseran pattern ditentukan oleh pinggiran dari awalan P yang bersesuaian. Pada contoh di atas, awalan yang bersesuaian adalah abcab, dengan panjang l = 5. Pinggiran terpanjang untuk string P[1..5] adalah ab yang panjangnya adalah b(5) = 2. Jarak pergeseran adalah l – b = 5 – 2 = 3. Jadi, pattern P digeser sejauh 3 karakter dan perbandingan dilakukan mulai pada posisi j = 3 dihitung dari awal pattern.

```
                123456789
Teks          : abcabcabd
Pattern       :    abcabd
                    ↑
                  j = 3
```

Setelah itu kita kembali membandingkan karakter per karekter seperti di proses sebelumnya sampai kita menemukan teks yang sama dengan pattern hingga karakter terakhir.

Algoritma Knuth-Morris-Pratt selengkapnya adalah sebagai berikut:

```
procedure KMPsearch(input m,n:integer,
input P : array[1..m] of char,input T :
array[1..n] of char, output idx :
integer)
{ Mencari kecocokan pattern P di dalam
teks T dengan algoritma Knuth-Morris-
Pratt. Jika ditemukan P di dalam T,
lokasi awal kecocokan disimpan di dalam
peubah idx.
  Masukan: pattern P yang panjangnya m
dan teks T yang panjangnya n.
          Teks T direpresentasika
sebagai string (array of character)
  Keluaran: posisi awal kecocokan
(idx). Jika P tidak ditemukan, idx = -
1.
}
Deklarasi
  i, j : integer
  ketemu : boolean
  b : array[1..m] of integer

procedure HitungPinggiran(input m :
integer, P : array[1..m] of char,
output b : array[1..m] of integer)
  { Menghitung nilai b[1..m] untuk
pattern P[1..m] }

Algoritma:
 HitungPinggiran(m, P, b)
 j←0
 i←1
 ketemu←false
 while (i ≤ n and not ketemu) do

   while((j > 0) and (P[j+1]≠T[i])) do
     j←b[j]
   endwhile

   if P[j+1]=T[i] then
     j←j+1
   endif
   if j = m then
     ketemu←true
   else
     i←i+1
   endif
 endwhile
 if ketemu then
   idx←i-m+1  { catatan: jika indeks
array dimulai dari 0, maka idx←i-m }
 else
   idx←-1
 endif
```

3.3 Kompleksitas

Untuk menghitung fungsi pinggiran dibutuhkan waktu $O(m)$, sedangkan pencarian *string* membutuhkan waktu $O(n)$, sehingga kompleksitas waktu algoritma Knuth-Morris-Pratt adalah $O(m+n)$.

4. PENERAPAN ALGORITMA KNUTH-MORRIS-PRATT PADA PENCOCOKAN DNA

Pada pendahuluan di atas telah disebutkan bahwa DNA dapat dianalogikan sebagai string yang terdiri dari 4 jenis karakter yaitu A, S, T, G. Karena itu, pencocokan DNA bisa diibaratkan sebagai pencocokan string seperti biasa.

Alasan kita menggunakan algoritma Knuth-Morris-Pratt pada pencocokan DNA adalah karena DNA hanya terdiri dari 4 jenis karakter dan seringkali dalam satu pattern karakter tersebut diulang-ulang. Karena seringnya karakter diulang-ulang, maka penggunaan Algoritma Knuth-Morris-Pratt pada pencocokan DNA akan menjadi sangat mangkus daripada algoritma *brute force* biasa. Berikut ini adalah contoh pencocokan DNA sample (S) dengan potongan DNA pattern (P).

Susunan DNA Sample (S)
AGTAGTAGTCAGTAGTCAGTCTGAC

Susunan DNA Pattern (P)
AGTAGTCAGTC

Langkah pertama kita akan mencari fungsi pinggiran dari DNA pattern P. Berikut ini adalah tabel nilai fungsi pinggiran dari DNA P

Tabel 1 Fungsi pinggiran untuk DNA Pattern

J	1	2	3	4	5	6	7	8	9	10	11
P(j)	A	G	T	A	G	T	C	A	G	T	C
B(j)	0	0	0	1	2	3	0	1	2	3	0

Langkah kedua samakan ujung kiri DNA Pattern dengan ujung kiri DNA Sample.

```
S = AGTAGTAGTCAGTAGTCAGTCTGAC
P = AGTAGTCAGTC
```

Karakter-karakter pada posisi 1-6 sama, tetapi pada posisi ke-7 karakter dari S dan P berbeda. Makaakan kita lakukan pergeseran dengan memperhatikan fungsi awalan dari P yang bersesuaian. Awalan yang bersesuaian adalah AGTAGT yang memiliki panjang l = 6. Pinggiran terpanjang dari P[1..6] adalah AGT yang panjangnya adalah 3. Maka jarak pergeseran adalah l – b = 6 – 3 = 3. Jadi, Pattern P digeser sejauh 3 karakter

```
S = AGTAGTAGTCAGTAGTCAGTCTGAC
P =    AGTAGTCAGTC
```

Kemudian kita bandingkan kembali mulai dari karakter ke-4 P. Kita akan temui bahwa karakter ke-11 dari P tidak sama dengan S. Jadi kita akan geser P sama dengan aturan sebelumnya. Awalan yang bersesuaian adalah AGTAGTCAGT yang memiliki panjang l = 10. Pinggiran terpanjang dari P[1..6] adalah AGT yang panjangnya adalah 3. Maka jarak pergeseran adalah l – b = 10 – 3 = 7. Jadi kita akan menggeser P sebanyak 7 karakter.

```
S = AGTAGTAGTGAGTAGTCAGTCTGAC
P =           AGTAGTCAGTC
```

Langkah selanjutnya akan kita bandingkan P dan S mulai dari karakter ke-7. Ternyata karakter dari P mulai dari karakter ke 7 hingga ke 11 memiliki kesamaan dengan karakter dari S. Maka dari itu kita akan mendapatkan bahwa P cocok dengan S.

4. KESIMPULAN

Algoritma pencocokan String Knuth-Morris-Pratt sangat cocok untuk digunakan dalam Algoritma untuk pencocokan DNA. Hal ini dikarenakan DNA hanya terdiri dari 4 jenis gugus karbon. Jika dianalogikan DNA adalah sebuah string, maka DNA tersebut merupakan kombinasi dari 4 jenis karakter. Dengan demikian sangat dimungkinkan bahwa akan terjadi perulangan karakter dalam DNA. Karena akan seringnya terjadi perulangan karakter itu, maka Algoritma pencocokan String Knuth-Morris-Pratt sangat cocok untuk digunakan sebagai Algoritma pencocokan DNA.

Di era yang semakin maju ini, penggunaan Algoritma Knuth-Morris-Pratt mungkin bisa menjadi alternatif dalam mencocokkan DNA untuk identifikasi.

REFERENSI

[1] Cheng Lok-Lam, Cheung D. W., Yiu Siu-Ming, Approximate String Matching in DNA Sequences, http://www.cs.hku.hk/~dcheung/publication/dasfaa2003_1.pdf, diakses tanggal 22 Mei 2007 pukul 9.00 WIB

[2] Rinaldi Munir, *Diktat Kuliah IF2251 Strategi Algoritmik*, Program Studi Teknik Informatika ITB, 2005