Aplikasi Algoritma *Colussi* Pada Pencocokan DNA dan Perbandingannya Dengan Algoritma *Knuth-Morris Pratt*

Raden Prana Adikhrisna (13506105)

Jurusan Teknik Informatika, Institut Teknologi Bandung, Indonesia
Bandung, Jawa Barat, Indonesia
e-mail: if16105@students.if.itb.ac.id

ABSTRAK

DNA adalah sejenis asam nukleat yang tergolong biomolekul utama penyusun berat kering setiap organisme. DNA tersusun atas rantai-rantai basa, yaitu *adenin* (A), *guanin* (G), *cytosin* (C), dan *timin* (T). Susunan AGTC tersebut dapat dijadikan landasan dalam pencocokan DNA. Algoritma berjenis algoritma *string matching* dapat dijadikan pedoman teknik pencocokan pola DNA karena kesamaan prinsip algoritma, yaitu membandingkan karakter per karakter. Ada berbagai jenis algoritma *string matching*, dan salah satu algoritma yang dapat digunakan adalah algoritma *Colussi*, salah satu turunan dari algoritma *Knuth-Morris-Pratt*.

Kata kunci: DNA, *string matching*, basa nitrogen, *Colussi*, *Knuth-Morris-Pratt*, *substring*, pencocokan, perbandingan, pola, subhimpunan, *adenin*, *guanin*, *cytosin*, *timin*.

1. PENDAHULUAN

1.1 DNA

DNA, atau asam deoksiribosa nukleat, adalah sejenis asam nukleat yang tergolong biomolekul utama penyusun berat kering setiap organisme. Di dalam sel, umumnya DNA terletak di dalam inti sel. Secara garis besar, peran DNA di dalam sebuah sel adalah sebagai materi genetik; artinya DNA menyimpan cetak biru bagi segala aktivitas sel. Hal ini berlaku umum untuk setiap organisme. Di antara perkecualian yang menonjol adalah beberapa jenis virus seperti HIV.

DNA merupakan polimer yang terdiri dari tiga komponen utama, yaitu gugus fosfat, gula deoksiribosa, dan basa nitrogen. Sebuah unit monomer DNA yang terdiri dari ketiga komponen tersebut dinamakan nukleotida, sehingga DNA tergolong polinukleotida. Rangka utama untai DNA terdiri dari gugus fosfat dan gula yang berselang-seling. Gula pada DNA adalah gula pentosa(berkarbon lima), yaitu 2-deoksiribosa. Dua gugus gula terhubung dengan fosfat melalui ikatan fosfodiester antara atom karbon ketiga pada cincin satu gula dan atom karbon kelima pada gula lainnya.

DNA terdiri atas dua untai yang berpilin membentuk struktur heliks ganda. Pada struktur heliks ganda, orientasi rantai nukleotida pada satu untai berlawanan dengan orientasi nukelotida untai lainnya. Hal ini disebut sebagai *antiparalel*. Masing-masing untai terdiri dari rangka utama, sebagai struktur utama, dan basa nitrogen, yang berinteraksi dengan untai DNA satunya pada heliks. Kedua untai pada heliks ganda DNA disatukan ikatan hidrogen antara basa-basa yang terdapat pada kedua untai tersebut. Empat basa yang ditemukan pada DNA adalah adenin(dilambangkan A), guanin(dilambangkan G), sitosin(dilambangkan C), dan timin(T). Adenin berikatan hidrogen dengan timin, sedangkan guanin berikatan dengan sitosin.

DNA memainkan peran penting dalam ilmu komputer, baik sebagai masalah riset dan sebagai sebuah cara komputasi. Riset dalam algoritma pencarian *string*, yang menemukan kejadian dari urutan huruf di dalam urutan huruf yang lebih besar, dimotivasi sebagian besar oleh riset DNA, dimana algoritma tersebut digunakan untuk mencari urutan tertentu dalam nukleotida dalam sebuah urutan yang besar. Dalam aplikasi lainnya seperti editor teks, algoritma sederhana sudah mencukupi untuk meyelesaikan masalah itu, namun urutan DNA menyebabkan algoritma-algoritma ini untuk menunjukkan sifat kasus-mendekati-terburuk disebabkan jumlah kecil dari karakter yang berbeda.

1.2 Algoritma Pencarian String

Algoritma pencarian string adalah algoritma untuk menemukan satu atau lebih keberadaan sebuah string(atau dalam hal ini disebut pola) dalam suatu teks. Pola dinotasikan sebagai

$$x = x[0..m - 1] \quad (1)$$

dengan m sebagai panjang pola. Teks dinotasikan sebagai

$$y = y[0..n - 1] \quad (2)$$

dengan n sebagai panjang teks. Keduanya dibentuk oleh kumpulan karkater alfabetis yang dinotasikan sebagai $\sum$ dengan σ sebagai ukurannya.

Algoritma pencarian string yang digunakan bekerja dengan cara sebagai berikut: Pemindaian teks dilakukan dengan bantuan sebuah jendela yang ukurannya = *m*. Pertama-tama letak ujung kiri jendela disamakan dengan ujung kiri teks, lalu karakter-karakter pada jendela dibandingkan dengan karakter-karakter pada pola (nama lainnya = percobaan) dan setelah seluruh pola dibandingkan atau setelah menemukan ketidakcocokan, maka jendelanya akan melompat ke arah kanan. Cara yang sama diulangi terus hingga ujung kanan jendela melebihi ujung kanan teks. Mekanisme ini sering disebut *sliding window mechanism.* Tiap percobaan pencocokan diasosiasikan dengan posisi ke-j pada teks ketika jendelanya diposisikan pada *y*[*j..j* + *m* - 1].

Ada bermacam-macam tipe pencarian string, antara lain:

- Algoritma Pencarian String secara *Brute Force*
- Algoritma *Knuth-Morris-Prat.*
- Algoritma *Boyer-Moore*
- Algoritma *Karp-Rabin*
- Dan sebagainya

Algoritma-algoritma tersebut juga memiliki berbagai variasi, contohnya algoritma Colussi yang merupakan pengembangan dari algoritma *Knuth-Morris-Pratt*, dan algoritma *Turbo-BM* yang merupakan pengembangan dari algoritma *Boyer-Moore*.

1.2 Algoritma *Colussi*

Algoritma *Colussi* merupakan suatu pengembangan dari algoritma *Knuth-Morris-Pratt*. Algoritma *Knuth-Morris-Pratt* sendiri adalah algoritma pencocokan string dengan cara memelihara informasi karakter-karakter sebelumnya untuk melakukan jumlah pergesaran yang lebih jauh. Pada algoritma *Colussi* sendiri, himpunan dari posisi pola dibagi menjadi dua subhimpunan terpisah. Lalu percobaan pencocokan berlangsung selama dua fase:

- Fase pertama: Perbandingan dilakukan dari kiri ke kanan pada teks yang terletak pada posisi yang sama dengan posisi pola dimana nilai dari fungsi *kmpNext* sedikit lebih besar dari -1. Posisi-posisi ini dinamakan *nohole(s)*.
- Fase kedua: Membandingkan posisi-posisi yang tersisa (nama lainnya adalah *hole(s)*) dari arah kiri ke kanan.

Strategi ini memberikan dua kelebihan, yaitu:

- Ketika terjadi ketidakcocokan pada fase pertama, maka setelah terjadi pergeseran tidak perlu untuk membandingkan teks dengan *noholes* yang dibandingkan pada percobaan sebelumnya.
- Ketika ketidakcocokan terjadi pada fase kedua yang berarti ada *suffix* dari pola yang sama dengan bagian di dalam teks, maka setelah terjadi pergeseran bagian *prefix* dari pola akan tetap sama dengan bagian dari teks tersebut, sehingga tidak diperlukan pembandingan kembali.

2. METODE

Pada algoritma *Colussi,* sistematika langkah yang digunakan sebagai berikut:

Untuk $0 \le i \le m - 1$, maka

$$k\min[i] = \begin{cases} d > 0 \Leftrightarrow x[0..i-1-d] = x[d..i-1] \;\&\; x[i-d] \neq x[i] \\ 0 \qquad\qquad\qquad\qquad lainnya \end{cases} \qquad (3)$$

Jika $kmin[i] \neq 0$ maka sebuah periodisitas berhenti pada posisi *i* di dalam *x*.

Untuk $0 < i < m$, jika $kmin[i - 1] \neq 0$ maka *i* adalah sebuah *nohole* dan *i* adalah sebuah *hole* jika terjadi sebaliknya. Misalkan *nd* + 1 adalah jumlah *nohole* di dalam *x*. Sebuah tabel *h* pertama mengandung *nohole* sebanyak *nd* + 1 secara terurut menaik dan selanjutnya *hole* sebanyak *m* – *nd* – 1 secara terurut menurun, maka:

- Untuk $0 \le i \le nd$, $h[i]$ adalah sebuah *nohole* dan $h[i] < h[i+1]$ untuk $0 \le i < nd$.
- Untuk $nd < i < m$, $h[i]$ adalah sebuah *hole* dan $h[i] > h[i+1]$ untuk $nd < i < m - 1$.

Jika *i* adalah sebuah *hole* maka *rmin*[*i*] adalah periode terkecil ketika *x* lebih besar dari *i*.

Nilai dari *first*[*u*] adalah suatu *integer* terkecil *v* dimana $u \le h[v]$. Kemudian asumsikan bahwa *x* sama dengan *y*[*j..j* + *m* – 1]. Jika $x[h[k]] = y[j + h[k]]$ untuk $0 \le k < r < nd$ dan $x[h[r]] \neq y[j + h[r]]$ dengan $j' = j + kmin[h[r]]$, maka tidak terjadi suatu okurensi *x* permulaan pada *y*[*j..j'*] dan *x* dapat berpindah posisi sejauh *kmin*[*h*[*r*]] ke arah kanan. Untuk lebih lanjutnya pada $x[h[k]] = y[j' + h[k]]$ untuk $0 \le k < first[h[r] - kmin[h[r]]]$ menandakan bahwa perbandingan dapat dilanjutkan dengan $x[h[first[h[r] - kmin[h[r]]]]]$ dan $y[j' + h[first[h[r] - kmin[h[r]]]]]$.

Jika $x[h[k]] = y[j + h[k]]$ untuk $0 \le k < r$ dan $x[h[r]] \neq y[j + h[r]]$ dengan $nd \le r < m$, lalu dimisalkan bahwa $j' = j + rmin[h[r]]$, maka tidak terjadi okurensi pada *x* permulaan pada *y*[*j..j'*] sehinga *x* dapat berpindah posisi sejauh *kmin*[*h*[*r*]] ke arah kanan. Untuk lebih lanjutnya pada $x[0..m - 1 - rmin[h[r]]] = y[j'..j + m - 1]$ menandakan bahwa perbandingan dapat dilanjutkan dengan $x[h[first[m - 1 - rmin[h[r]]]]]$ dan $y[j' + h[first[m - 1 - rmin[h[r]]]]]$.

Untuk menghitung nilai dari *kmin*, sebuah tabel *hmax* digunakan dan didefinisikan sebagai berikut:

hmax[*k*] adalah sesuatu dimana $x[k..hmax[k] - 1] = x[0..hmax[k] - k - 1]$ dan $x[hmax[k]] \neq x[hmax[k] - k]$

Nilai dari *ndh*0[*i*] adalah jumlah dari *nohole* yang sedikit lebih kecil dari *i*.

Sekarang kita dapat mendefinisikan dua fungsi yaitu *shift* dan *next* sebagai berikut:

- *shift*[*i*] = *kmin*[*h*[*i*]] dan *next*[*i*] = *ndh*0[*h*[*i*] – *kmin*[*h*[*i*]]] untuk *i* < *nd*
- *shift*[*i*] = *rmin*[*h*[*i*]] dan *next*[*i*] = *ndh*0[*m* – *rmin*[*h*[*i*]]] untuk *nd* ≤ *i m*
- *shift*[*m*] = *rmin*[0] dan *next*[*m*] = *ndh*0[*m* – *rmin*[*h*[*m* – 1]]]

Selanjutnya, dalam suatu percobaan dimana jendela diposisikan pada bagian teks *y*[*j*..*j* + *m* – 1], ketika terjadi suatu ketidakcocokan antara *x*[*h*[*r*]] dan *y*[*j* + *h*[*r*]] maka jendela harus bergerak menggunakan fungsi *shift*[*r*] dan perbandingan dapat dilanjutkan pada posisi pola *h*[*next*[*r*]].

Secara keseluruhan, fase *preprocessing* dapat dilakukan dengan kompleksitas waktu dan ruang *O*(*m*). Fase pencarian sendiri dapat dilakukan dengan kompleksitas waktu *O*(*n*) dan paling banyak dilakukan 3n/2 perbandingan karakter teks selama fase tersebut.

3. IMPLEMENTASI DAN PERBANDINGAN

Sekarang akan dilakukan pengujicobaan penggunaan algoritma *Colussi* pada pencarian suatu pola DNA pada suatu DNA sampel. Setelahnya juga dilakukan analisis dan perbandingan dengan algoritma *Knuth-Morris-Pratt*. Contoh program berdasarkan pengujicobaan ini juga dapat diunduh dari folder:

students.if.itb.ac.id/~if16105/files/DNAMatcher

3.1 Uji Coba Kasus

Contoh sampel DNA : G C A T C G C A G A G A G T A T A C A G T A C G.

Contoh pola secarik DNA yang akan dibandingkan dengan sampel: G C A G A G A G

Tabel yang digunakan untuk perhitungan:

Tabel 1 Tabel Untuk Fase Pencarian

i	0	1	2	3	4	5	6	7	8
x[*i*]	G	C	A	G	A	G	A	G	
kmpNext[*i*]	-1	0	0	-1	1	-1	1	-1	1
kmin[*i*]	0	1	2	0	3	0	5	0	
h[*i*]	1	2	4	6	7	5	3	0	
next[*i*]	0	0	0	0	0	0	0	0	0
shift[*i*]	1	2	3	5	8	7	7	7	7
hmax[*i*]	0	1	2	4	4	6	6	8	8
rmin[*i*]	7	0	0	7	0	7	0	8	
*ndh*0[*i*]	0	0	1	2	2	3	3	4	

nd = 3

Percobaan pertama:

y = G C A T C G C A G A G A G T A T A C A G T A..
1 2 3
x = G C A G A G A G
Pindah 3 karakter (*shift*[2])

Percobaan kedua:

y = ..T C G C A G A G A G T A T A C A G T A C G
1 2
x = G C A G A G A G
Pindah 2 karakter (*shift*[1])

Percobaan ketiga:

y = ..T C G C A G A G A G T A T A C A G T A C G
8 1 2 7 3 6 4 5
x = G C A G A G A G
Pindah 7 karakter (*shift*[8])

Percobaan keempat:

y = ..T C G C A G A G A G T A T A C A G T A C G
1
x = G C A G A G A G
Pindah 1 karakter (*shift*[0])

Percobaan kelima:

y = ..T C G C A G A G A G T A T A C A G T A C G
1
x = G C A G A G A G
Pindah 1 karakter (*shift*[0])

Percobaan keenam:

y = ..T C G C A G A G A G T A T A C A G T A C G
1
x = G C A G A G A G
Pindah 1 karakter (*shift*[0])

Percobaan ketujuh:

y = ..T C G C A G A G A G T A T A C A G T A C G
1
x = G C A G A G A G
Pindah 1 karakter (*shift*[0])

Percobaan kedelapan:

y = ..T C G C A G A G A G T A T A C A G T A C G
1 2 3
x = G C A G A G A G
Pindah 3 karakter (*shift*[2])

3.2 Analisis dan Perbandingan Dengan Algoritma *Knuth-Morris-Pratt*

Dari percobaan-percobaan yang diapat, jumlah perbandingan keseluruhan yang terjadi sebanyak 20 kali. Pertama kali ditemukan kecocokan pada pola didapat setelah dua kali pergeseran dan 12 kali pencocokan

karakter. Dengan menggunakan pola DNA yang sama, dengan algoritma *Knuth-Morris-Pratt* hasil yang didapat adalah perbandingan keseluruhan dilakukan sebanyak 18 kali dan kecocokan pada pola didapat setelah dua kali pergeseran dan 13 kali pencocokan karakter.

Setelah diamati secara seksama, terlihat bahwa algoritma *Colussi* menemukan kecocokan lebih cepat daripada algoritma *Knuth-Morris-Pratt*. Dari segi kompleksitas waktu pencarian pun terlihat bahwa algoritma *Colussi* lebih cepat daripada algoritma *Knut-Morris-Pratt* ($O(n)$ pada algoritma *Colussi* dibandingkan $O(m + n)$ pada algoritma *Knuth-Morris-Pratt*).

4. KESIMPULAN

DNA *matching* merupakan salah satu persoalan yang dapat diselesaikan dengan algoritma *string matching*. Persoalannya adalah sedikitnya jumlah variabel karakter yang ada membuat tingginya frekuensi kasus-kasus yang mendekati kasus-kasus terburuk.

Algoritma *Colussi* sebagai salah satu turunan dari algoritma *Knuth-Morris-Pratt* dapat digunakan dalam proses DNA *matching* tersebut. Keunggulan algoritma ini adalah kompleksitas waktu yang cukup rendah dan kemampuan menemukan kecocokan yang cukup cepat.

Dibandingkan dengan algoritma *Knuth-Morris-Pratt*, algoritma *Colussi* memiliki keunggulan dalam hal kompleksitas waktu dan kecepatan pencarian.

REFERENSI

[1] DNA, "http://en.wikipedia.org/DNA", 2008, tanggal akses: 19 Mei 2008 pukul 10.55

[2] Exact string matching algorithm, "http://www-igm.univ-mlv.fr/~lecroq/string", 1997, tanggal akses: 12 Mei 2008 pukul 16.35

[3] Algoritma Knuth-Morris-Pratt, "http://en.wikipedia.org/Knuth-Morris-Pratt_algorithm", 2008, tanggal akses: 19 Mei 2008 pukul 11.02

[4] Rinaldi Munir, "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Teknik Informatika ITB, 2006.

[5] *String Matching*, "http://en.wikipedia.org/String_searching_algorithm", 2008, tanggal akses: 19 Mei 2008 pukul 10.55