Algoritma *Greedy* untuk Menyelaraskan DNA *Sequence*

Dita Anindhika 13509023[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13509023@std.stei.itb.ac.id

***Abstract*—DNA sequencing merupakan suatu tahapan yang penting pada proyek-proyek melibatkan gen manusia karena kesalahan pengurutan DNA tentunya akan berdampak sangat besar pada jalannya keseluruahn proyek tersebut. Permasalahan yang biasanya terjadi pada DNA sequencing adalah kesalahan pengurutan DNA maupun error lainnya yang sejenis. Untuk menanggulangi permasalahan tersebut dapat digunakan dynamic programming ataupun algoritma *greedy*. Penggunaan algoritma tersebut memanfaatkan suatu variable X-drop. Dalam makalah ini, hanya dibahas penyelesaian solusi dengan menggunakan algoritma *greedy* saja. Nantinya algoritma *greedy* akan melakukan pengambilan keputusan untuk membuat suatu urutan DNA yang dibuat terurut mengecil (*descending*) dari hasil penomoran dari tiap-tiap elemen DNA.**

***Index Terms*—Algoritma greedy, DNA, DNA sequence, DNA sequencing, Dynamic programming, X-drop**

I. PENDAHULUAN

Eksperimen mengenai DNA sequencing biasanya melalui dua buah tahapan. Tahapan pertama adalah shotgun sequencing dan tahap selanjutnya adalah walking yang mana merupakan proses pengurutan yang dilakukan kepada bagian yang tidak tercover oleh tahap yang sudah dilakukan.

Permasalahan yang biasanya terjadi pada DNA-sequencing sendiri adalah kesalahan dalam pengurutan DNA atau permasalahan lainnya yang masih sejenis dengan isu tersebut. Biasanya untuk menanggulangi permasalahan tersebut dapat digunakan penyelesaian dengan memanfaatkan dynamic programming atau agoritma *greedy*. Nantinya akan digunakan sebuah variabel X-drop yang dalam pengimplementasiannya dapat menggunakan dynamic programming atau algoritma *Greedy* yang sudah disebutkan sebelumnya. Namun, pada makalah ini hanya akan membahas penggunaan variabel X-drop yang diimplemantasikan menggunakan algoritma *greedy*.

II. DASAR TEORI

A.Algoritma Greedy

Algoritma *Greedy* merupakan aloritma yang paling sering digunakan dalam menyelesaikan masalah optimasi. Walaupun dengan menggunakan Algoritma *Greedy* belum tentu didapatkan hasil yang paling optimum (optimum global), algoritma ini tetap digunakan sebagai basis untuk pendekatan heuristik. Algortima ini membentuk solusi tersebut dengan langkah demi langkah (*step by step*). Tentunya akan banyak kemungkinan langkah yang dapat ditelusuri untuk mencapai solusi, maka dengan algoritma *Greedy* ini akan dilakukannya pengambilan keputusan yang memenuhi suatu kondisi optimum lokal yang diharapkan nantinya dapat menghasilkan suatu solusi yang optimum global.

Berikut ini adalah salah satu contoh penggunaan algoritma *Greedy* dalam masalah penukaran uang. Misalnya terdapat uang $25 yang ingin ditukarkan dengan sekumpulan uang pecahan $10, $5, $2, $1. Tentukanlah kombinasi uang pecahan dengan jumlah terminimum yang dapat ditukarkan dengan $25! Dengan algoritma *Greedy*, maka yang pertama akan dipilih adalah uang pecahan yang memenuhi kondisi optimum lokal yaitu uang pecahan $10. Langkah tersebut kemudian akan diulang terus menerus sampai ditemukannya sebuah solusi. Untuk kasus ini, maka algoritma *Greedy* akan menghasilkan solusi $25=$10+$10+$5. Pengambilan keputusan dengan optimum lokal tersebut pun berhasil menghasilkan suatu solusi yang optimum global.

Persoalan optimasi pada algoritma *Greedy* ini disusun oleh berbagai elemen yang dijelaskan sebagai berikut :

1. Himpunan kandidat
   Himpunan ini berisi elemen-elemen yang mungkin mejadi kandidat solusi
2. Himpunan solusi
   Himpunan ini berisi elemen-elemen yang terpilih sebagai solusi persoalan
3. Fungsi seleksi
   Fungsi yang memilih kandidat untuk menjadi

sebuah solusi yang dianggap optimal

4. Fungsi kelayakan
   Fungsi yang menilai layak atau tidaknya suatu kandidat solusi sehingga tidak melanggar constraint-constraint yang ada dalam penyelesaian masalah.
5. Fungsi objektif
   Fungsi yang meminimalkan atau memaksimumkan nilai solusi

```
function greedy(input C: himpunan_kandidat)→ himpunan_kandidat
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy
  Masukan: himpunan kandidat C
  Keluaran: himpunan solusi yang bertipe himpunan_kandidat
}
Deklarasi
    x : kandidat
    S : himpunan_kandidat

Algoritma:
    S ← {}   { inisialisasi S dengan kosong }
    while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)        { pilih sebuah kandidat dari C}
      C ← C - {x}           { elemen himpunan kandidat berkurang satu }
      if LAYAK(S ∪ {x}) then
          S ← S ∪ {x}
      endif
    endwhile
    {SOLUSI(S) or C = {} }

    if SOLUSI(S) then
      return S
    else
      write('tidak ada solusi')
    endif
```

Gambar 1 Skema Umum Algoritma *Greedy*

Dengan kata lain, algoritma *Greedy* akan mencari sebuah himpunan solusi, S, yang merupakan himpunan bagian dari himpunan kandidat, C. Dalam hal ini, himpunan S merupakan himpunan kandidat, C, yang dikenai fungsi seleksi dan kelayakan. Dan solusi akhir yang diambil adalah himpunan solusi, S, yang dikenai fungsi objektif.

Pilihan yang dibuat menggunakan algoritma *Greedy* ditentukan oleh pilihan-pilihan yang telah dibuat sampai saat ini namun tidak oleh pilihan-pilihan yang akan datang. Secara iteratif, pilihan *Greedy* dilakukan sehingga membuat permasalahan yang ada menjadi permasalahan yang lebih kecil. Perbedaan mendasar antara algoritma *Greedy* dengan program dinamis adalah pada algoritma *Greedy* tidak pernah merubah pilihan yang telah dibuat sebelumnya. Perbedaan lain adalah pada algoritma *Greedy* hanya dilakukan perhitungan untuk sebuah kemungkinan solusi saja sedangkan pada program dinamis akan dilakukan perhitungan untuk banyak kemungkinan solusi.

B. DNA (Deoxyribonucleid Acid)

DNA (*deoxyribonucleid acid*) adalah rantai double helix berpilin yang terdiri atas polinukleotida yang berfungsi sebagai pewaris sifat dan juga berperan dalam proses sintesis protein. Bentuk DNA sendiri adalah rantai duoble helix yang berpilin ke kanan. Berikut ini adalah komponen penyusun DNA, yaitu:

1. Gula deoksiribosa
2. Gugus fosfat
3. Basa nitrogen

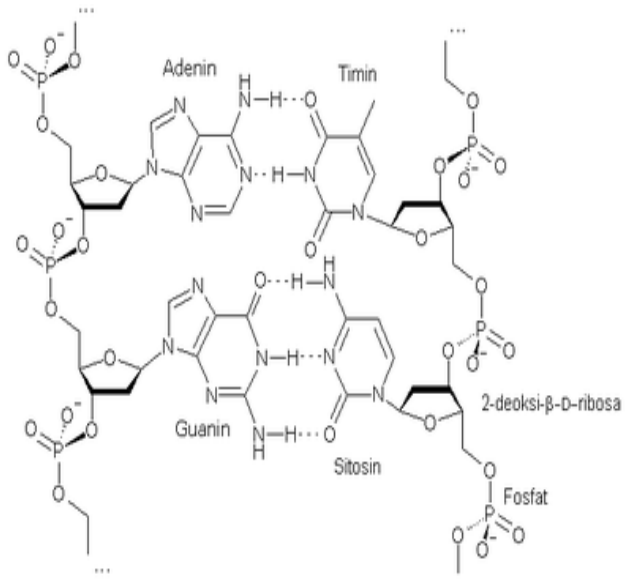
Gambar 2 Double Helix DNA

Jika pada sebuah lempeng DNA mengandung gugus fosfat, gula deoksiribosa dan basa nitrogen, maka lempeng tersebut di sebut nikleutida. Namun, jika lempeng DNA tersebut hanya mengandung basa nitrogen dan gula deoksiribosa saja maka lempeng tersebut adalah nukleusida.

Terdapat dua jenis basa nitrogen yang berikatan pada DNA, yaitu :

1. Purin
   Purin merupakan basa nitrogen adenine dan guanin
2. Pirimidin
3. Pirimidin merupakan basa nitrogen sitosin dan timin.

Basa purin dan pirimidin tersebut selalu berpasangan dan membentuk suatu ikatan hydrogen. Adenin akan selalu berpasangan dengan timin dan membentuk dua buah ikatan hidrogen, sedangkan guanine akan selalu berpasangan dengan sitosin dan membentuk tiga buah ikatan hidrogen.



Gambar 3 Basa Nitrogen Berpasangan pada DNA

DNA manusia sendiri terdiri dari sekitar tiga milyar basa nitrogen dan lebih dari 99% dari basa tersebut adalah sama untuk setiap orang. Urutan dari basa nitrogen tersebut nantinya menentukan informasi yang tersedia untuk membangun dan memelihara suatu organisme.

III. PENYELARASAN DNA SEQUENCE MENGGUNAKAN ALGORITMA *GREEDY*

Eksperimen mengenai DNA sequence biasanya dilakukan dalam dua tahapan, tahapan pertama adalah *shotgun sequencing* dan tahapan kedua adalah *walking*. Shotgun sequencing biasanya dianggap sebagai sebuah proses yang *stochastic* karena banyak subinterval pendek pada lokasi acak di untaian DNA yang diurutkan. Sedangkan *walking* adalah proses penyelesaian yang deterministik karena semua bagian yang tidak tercakup oleh *shotgun sequencing* akan diurutkan.

1. $T' \leftarrow T \leftarrow S(0,0) \leftarrow 0$
2. $k \leftarrow L \leftarrow U \leftarrow 0$
3. repeat
4. $\quad k \leftarrow k+1$
5. $\quad$ for $i \leftarrow \lceil L \rceil$ to $\lfloor U \rfloor + 1$ in steps of $\frac{1}{2}$ do
6. $\quad\quad j \leftarrow k - i$
7. $\quad\quad$ if i is an integer then
8. $$S(i,j) \leftarrow \max \begin{cases} S(i-\frac{1}{2}, j-\frac{1}{2}) + mat/2 & \text{if } L \le i-\frac{1}{2} \le U \text{ and } a_i = b_j \\ S(i-\frac{1}{2}, j-\frac{1}{2}) + mis/2 & \text{if } L \le i-\frac{1}{2} \le U \text{ and } a_i \ne b_j \\ S(i, j-1) + ind & \text{if } i \le U \\ S(i-1, j) + ind & \text{if } L \le i-1 \end{cases}$$
9. $\quad\quad$ else
10. $$S(i,j) \leftarrow S(i-\tfrac{1}{2}, j-\tfrac{1}{2}) + \begin{cases} mat/2 & \text{if } a_{i+\frac{1}{2}} = b_{j+\frac{1}{2}} \\ mis/2 & \text{if } a_{i+\frac{1}{2}} \ne b_{j+\frac{1}{2}} \end{cases}$$
11. $\quad\quad T' \leftarrow \max\{T', S(i,j)\}$
12. $\quad\quad$ if $S(i,j) < T - X$ then $S(i,j) \leftarrow -\infty$
13. $\quad L \leftarrow \min\{i : S(i, k-i) > -\infty\}$
14. $\quad U \leftarrow \max\{i : S(i, k-i) > -\infty\}$
15. $\quad L \leftarrow \max\{L, k+1-N\}$
16. $\quad U \leftarrow \min\{U, M-1\}$
17. $\quad T \leftarrow T'$
18. until $L > U + 1$
19. report T'

Gambar 4 Penyelesaian Persoalan dengan Dynamic Programming

Dalam melakukan penyelarasan untuk DNA sequence dapat digunakan sebuah variabel X-drop yang dalam pengimplementasiannya dapat menggunakan dynamic programming atau algoritma *Greedy*. Penyelarasan dua buah DNA sequence A dan B bertujuan untuk mendapatkan susunan sequence yang berbentuk $a1a2\ldots ai$ dan $b1b2\ldots bj$, untuk sebuah $i \le M$ dan $j \le N$ yang memiliki score tertinggi. Untuk sebuuah *i* dan *j*, score tertinggi untuk sebuah susunan sequence didefinisikan dengan *S(i,j)* dimana ditambahkan mat > 0 untuk setiap kolom susunan dengan nucleotide yang identik, mis <0 untuk setiap kolom dengan nucleotide yang berbeda, dan ind < 0 untuk setiap kolom dimana nucleotide dipasangkan dengan symbol gap. *S(0,0)* harus menghasilkan nilai 0 karena susunan dengan 0 kolom berarti tidak ada susunan yang terbentuk. Semua penjelasan diatas dirangkum dalam persamaan dibawah ini.

$$S(i,j) = \max \begin{cases} S(i-1, j-1) + mat & \text{if } i > 0,\ j > 0 \text{ and } a_i = b_j \\ S(i-1, j-1) + mis & \text{if } i > 0,\ j > 0 \text{ and } a_i \ne b_j \\ S(i, j-1) + ind & \text{if } j > 0 \\ S(i-1, j) + ind & \text{if } i > 0. \end{cases}$$

Algoritma *greedy* untuk penyusunan DNA Sequence berkaitan dengan pengukuran nilai perbedaan antara dua buah sequence DNA, bukan nilai kesamaan antara keduanya. Dengan kata lain, near-identify dari dua buah sequence dijelaskan dengan menggunakan sebuah angka positif yang mana semakin kecil nilai angka tersebut maka semakin dekat identitas dua buah sequence tersebut.

Dengan demikian, fungsi untuk menghitung near-identify dua buah sequence, yakni *D(i,j)*, merupakan nilai atau angka terkecil yang menyatakan perbedaan dua buah sequence *i* dan *j*.

Di bawah ini merupakan satu contoh perhitungan nilai score dari dua buah DNA sequence.

```
...A...G...        ...A-...G...
...C...T...        ...-C...G...
```

Gambar 5 Perhitungan Dua Buah DNA Sequence

Gambar pada contoh menunjukkan transformasi yang dialami oleh sebuah DNA sequence. Pada contoh ini, simbol titik menyatakan suatu susunan DNA yang cocok diantara dua buah sequence tersebut.

Dari transformasi yang terlihat pada contoh di atas, dihasilkan perbedaan nilai dari kedua sequence tersebut, distance, ekivalen dengan,

$$2 \times mis = 2 \times ind + mat$$

atau

$$ind = mis - mat$$

dimana, *ind = indels*
mis = mismatches
mat = matches.

Dengan mengetahui persamaan tersebut, dapat diturunkan bahwa setiap *a1a2…ai* dan *b1b2…bj* yang mempunyai nilai perbedaan *d* akan memiliki nilai score,

$$S'(i+j,d) = (i+j) \times mat/2 - d \times (mat - mis)$$

Dari persamaan tersebut dapat dilihat bahwa dengan meminimasi nilai *d* kita akan memaksimalkan nilai *score* yang didapat. Dengan melalui ini, maka dapat dirumuskan bahwa

$$S(i,j) = S'(i=j, D(i,j))$$

Persamaan di atas telah sebelumnya dibuktikan oleh Zheng Zhang, Scott Schwartz, Lukas Wagner, dan Webb Miller. Dari persamaan tersebut terlihat bahwa *S(i,j)* harus di-*prune* pada saat kondisi tertentu. Kondisi tersebut adalah saat menemukan sebuah posisi *(i,j)* pada sebuah diagonal k dimana *D(i,j) = d*. Namun jika kondisi ini tidak terpenuhi maka proses *prune* tidak perlu dilakukan.

1. $i \leftarrow 0$
2. while $i < \min\{M, N\}$ and $a_{i+1} = b_{i+1}$ do $i \leftarrow i + 1$
3. $R(0,0) \leftarrow i$
4. $T' \leftarrow T[0] \leftarrow S'(i+i, 0)$
5. $d \leftarrow L \leftarrow U \leftarrow 0$
6. repeat
7. $\quad d \leftarrow d + 1$
8. $\quad d' \leftarrow d - \left\lfloor \frac{X + mat/2}{mat - mis} \right\rfloor - 1$
9. $\quad$ for $k \leftarrow L - 1$ to $U + 1$ do
10. $\quad\quad i \leftarrow \max \begin{cases} R(d-1, k-1) + 1 & \text{if } L < k \\ R(d-1, k) + 1 & \text{if } L \le k \le U \\ R(d-1, k+1) & \text{if } k < U \end{cases}$
11. $\quad\quad j \leftarrow i - k$
12. $\quad\quad$ if $i > -\infty$ and $S'(i+j, d) \ge T[d'] - X$ then
13. $\quad\quad\quad$ while $i < M$, $j < N$ and $a_{i+1} = b_{j+1}$ do
14. $\quad\quad\quad\quad i \leftarrow i + 1$; $j \leftarrow j + 1$
15. $\quad\quad\quad R(d,k) \leftarrow i$
16. $\quad\quad\quad T' \leftarrow \max\{T', S'(i+j, d)\}$
17. $\quad\quad$ else $R(d,k) \leftarrow -\infty$
18. $\quad T[d] \leftarrow T'$
19. $\quad L \leftarrow \min\{k : R(d,k) > -\infty\}$
20. $\quad U \leftarrow \max\{k : R(d,k) > -\infty\}$
21. $\quad L \leftarrow \max\{L, \max\{k : R(d,k) = N + k\} + 2\}$
22. $\quad U \leftarrow \min\{U, \min\{k : R(d,k) = M\} - 2\}$
23. until $L > U + 2$
24. report T'

Gambar 6 Algoritma *Greedy* untuk Penyelarasan DNA Sequence

Terdapatnya proses pruning pada penyelarasan DNA Sequence menggunakan algoritma Greedy tidak boleh mempengaruhi hasil akhir yang seharusnya diperoleh. Oleh karena itu, perlu dibuktikan bahwa proses pruning sekalipun dilakukan (kondisi terjadinya pruning ditemukan) tidak boleh mempengaruhi hasil akhir. Hal ini juga sudah dibuktikan oleh Zheng Zhang, Scott Schwartz, Lukas Wagner, dan Webb Miller pada jurnalnya

IV. KESIMPULAN

Penggunaan algoritma *Greedy* dalam penyelarasan DNA sequence dengan memanfaatkan variable x-drop akan bekerja untuk melakukan suatu penyelarasan pengurutan DNA dengan membuat suatu urutan yang terurut mengecil (*descending*) berdasarkan penomoran dari tiap elemen-elemen DNA, sehingga penyelesaian pernasalahan dalam penyelarasan DNA sequence menggunakan algoritma *greedy* adalah salah satu solusi yang dapat digunakan

REFERENSI

[1] Munir. Rinaldi, *Diktat Kuliah IF3051 Strategi Algoritma*,, Bandung, 2009, hal : 41-49

Tanggal akses : 8 Desember 2011
[2] Allon G. Percus, Torney. David C, "*Greedy* Algorithm for Optimized DNA Sequencing", unpublished.
Tanggal akses : 8 Desember 2011
[3] David R. Powell, David L. Dowe, Llyod Allison, Dix. Trevor I, "Discovering Simple DNA Sequence by Compression", unpublished.
Tanggal akses : 8 Desember 2011
[4] Zheng Zhang, Scott Schwartz, Lukas Wagner,Miller. Webb, "A *Greedy* Algorithm for Alligning DNA Sequences" Journal of Computational Biology A Journal of Computational Molecular Cell Biology.
Tanggal akses : 8 Desember 2011
[5] http://en.wikipedia.org/wiki/DNA_sequencing
Tanggal akses : 8 Desember 2011
[6] http://seqcore.brcf.med.umich.edu/doc/educ/dnapr/sequencing.html
Tanggal akses : 8 Desember 2011
[7] http://id.wikipedia.org/wiki/deoksiribo-nukleat.html
Tanggal akses : 8 Desember 2011
[8] http://ghr.nlm.nih.gov/handbook/basics/dna
Tanggal akses : 8 Desember 2011
[9] http://www.absoluteastronomy.com/topics/Greedy_algorithm
Tanggal akses : 8 Desember 2011

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Desember 2011

Dita Anindhika - 13509023