Aplikasi *Divide and Conquer* dalam Algoritma Hirschberg untuk Mengidentifikasi Mutasi pada DNA

Achmad Baihaqi - 13508030[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*if18030@students.if.itb.ac.id*

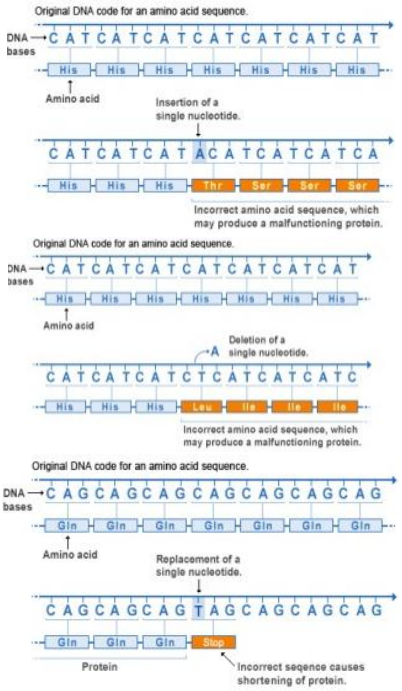
***Abstrak*—Penemuan DNA merupakan salah satu penemuan terbesar dalam bidang biologi. Siapa sangka, dibalik ukuran DNA yang kecil ini ternyata menyimpan informasi yang besar dalam tubuh manusia. Tidak ada kehidupan tanpa DNA ini. Tetapi akhir-akhir ini diketahui bahwa ada beberapa penyakit yang terjadi akibat kelainan DNA. Kelainan ini disebabkan mutasi. Mutasi adalah perubahan untaian DNA yang disebabkan oleh faktor-faktor eksternal. Karena untaian DNA ini sangat kecil dan panjang, maka sulit sekali untuk mengidentifikasi apa mutasi yang terjadi sehingga penyakit tersebut bisa terjadi. Tetapi dengan hadirnya komputer maka pekerjaan peneliti biologi ini lebih ringan karena bisa dibantu oleh komputer dengan kecepatannya. Tetapi, apa yang harus dilakukan komputer untuk membantunya? Disinilah peran penting algoritma-algoritma yang telah ditemukan ahli informatika. Dengan algoritma yang diterapkan pada komputer, masalah ini bisa tertangani dengan cepat. Salah satu algoritma untuk menyelesaikan permasalahan peneliti biologi ini adalah algoritma Hirschberg yang ditemukan oleh Dan Hirschberg. Algoritma ini menggunakan teknik *Divide and Conquer* dalam menyelesaikan permasalahan. Makalah ini akan membahas bagaimana algoritma Hirschberg ini bisa menyelesaikan masalah ini dibantu dengan teknik algoritma-algoritma lain. Pada akhir makalah penulis memberikan contoh program komputer untuk implementasi algoritma ini dalam menyelesaikan masalah tersebut.**

Kata kunci: Hirschberg, *Divide and Conquer*, Program Dinamis, *Edit-distance*, Mutasi, DNA

I. PENDAHULUAN

Asam deoksiribonukleat (*deoxyribonucleic acid*), atau biasa disebut DNA, adalah biomolekul yang berupa asam nukleat (terdapat dalam inti sel atau *nucleus*), yang berfungsi untuk menyimpan informasi genetik pada suatu organisme. DNA berbentuk untai ganda (*double helix*) yang disatukan oleh ikatan hidrogen antara basa-basa di dalam kedua untai tersebut. Basa-basa tersebut adalah Adenin (A), Sitosin (C), Guanin (G), dan Timin (T). Adenin berikatan dengan Timin, dan Sitosin berikatan dengan Guanin [2].

Suatu untaian DNA bisa saja berubah karena faktor-faktor tertentu. Perubahan urutan untaian DNA ini dinamakan mutasi. Mutasi terdiri dari macam-macam jenis. Akan tetapi pada makalah ini hanya dibahas mutasi pada skala kecil atau disebut juga dengan muasi titik (*point mutation*). Maksud dari skala kecil ini, yaitu mutasi yang terjadi perubahan pada basa-basa nitrogen penyusun DNA. Mutasi pada skala kecil ini terbagi menjadi 3 jenis yaitu insersi, delesi, dan substitusi. Insersi terjadi ketika suatu basa nitrogen mengisi diantara urutan basa nitrogren sehingga kerangka urutan basa nitrogen tersebut mundur ke belakang dan otomatis asam amino yang akan dibentuk juga akan berubah. Delesi terjadi jika terdapat pengurangan satu atau lebih pasangan basa nitrogren pada untaian DNA. Sedangkan substitusi terjadi jika terdapat pergantian suatu basa nitrogen menjadi basa nitrogren lain yang berbeda [8].



Gambar 1 Insersi (atas), Delesi (tengah), dan Substitusi (bawah)

Mutasi ini akan berpengaruh pada urutan protein yang dibentuk oleh DNA tersebut, sehingga bisa menyebabkan kelainan pada organisme yang mengalami mutasi tersebut.

Untuk mengidentifikasi jenis mutasi ini dapat dilakukan dengan cara membandingkan untaian DNA yang termutasi dengan untaian DNA yang asli. Akan

tetapi karena untaian DNA yang sesungguhnya sangat panjang, maka dibutuhkan suatu teknik untuk mengidentifikasi mutasi DNA tersebut secara optimal. Optimal dalam hal ini adalah optimal dalam penggunaan memori dan optimal dalam waktu eksekusi. Algoritma yang akan digunakan adalah teknik *divide and conquer* pada algoritma Hirschberg.

II. METODE

A. Penjajaran Barisan (Sequence Alignment)

Untuk mencari jenis mutasi yang terjadi pada suatu untaian DNA, maka harus dilakukan penjajaran antar untaian yang akan dibandingkan. Penjajaran barisan adalah prosedur untuk menjajarkan dua buah barisan (*sequence*) dari DNA dengan tujuan mencari kesamaan di antara barisan-barisan tersebut atau untuk membuktikan bahwa kedua *sequence* yang dibandingkan berasal dari *sequence* yang sama [2].

```
FTFTALILLAVAV
F--TAL-LLA-AV
```

Gambar 2 Penjajaran Barisan

B. Algoritma Hirschberg

Algoritma ini merupakan salah satu algoritma yang optimal untuk menyelesaikan penjajaran barisan. Algoritma ini adalah versi *divide and conquer* dari algoritma Needleman-Wunsch dengan fungsi yang yang sama yaitu untuk mencari penjajaran barisan [9].

Ide dari algoritma ini adalah dengan membagi-bagi string yang dibandingkan hingga terdapat salah satu string yang mengandung satu karakter atau sudah kosong. Apabila hal demikian terjadi, maka penjajaran akan sangat mudah dicari. Ini adalah basis dari rekursifitas algoritma ini. Terdapat beberapa kemungkinan apabila salah satu string mengandung satu karakter atau sudah kosong. Penulis misalkan, terdapat dua string yaitu A dan B. A anggap saja untaian DNA yang asli dan B adalah untaian DNA yang termutasi. Masing-masing karakter penyusun string kita anggap sebagai basa nitrogen penyusun DNA. Kemungkinan tersebut diantaranya:

1. Apabila A adalah string kosong, berarti terjadi mutasi jenis insersi. Hal ini karena pada B masih mengandung suatu untaian DNA yang tidak jelas asalnya darimana. Ilustrasinya:
   A = “”
   B = “ACC”
   Hasil penjajarannya menjadi:
   A = “---“
   B = “ACC”
   Tanda ‘-‘ berarti terjadi mutasi jenis delesi atau insersi.
2. Apabila B adalah string kosong, berarti terjadi mutasi jenis delesi. Hal ini karena pada A masih terdapat untaian DNA yang tiba-tiba hilang pada B. Ilustrasinya:
   A = “AATC”
   B = “”
   Hasil penjajarannya menjadi:
   A = “AATC”
   B = “----“
3. Apabila A hanya terdiri dari satu karakter sedangkan B tidak kosong, maka kemungkinannya yang bisa terjadi adalah jenis mutasi substitusi, insersi, dan substitusi disertai insersi.
   a. Substitusi pada kondisi diatas bisa terjadi apabila masing-masing string terdiri dari satu karakter, misalkan:
      A = “C”
      B = “G”
      Hasil penjajarannya menjadi:
      A = “C“
      B = “G”
      Substitusinya adalah karakter C digantikan oleh karakter G.
   b. Insersi bisa terjadi bila jumlah karakter B lebih dari A, misalkan:
      A = ‘C’
      B = ‘GCGA’
      Hasil penjajarannnya menjadi:
      A = ‘-C—‘
      B = ‘GCGA’
   c. Substitusi disertai insersi bisa terjadi bila jumlah karakter pada B lebih dari A dan di dalam B tidak ada karakter yang sama dengan A, misalkan:
      A = “C”
      B = “GTAA”
      Hasil penjajarannya menjadi:
      A = “C---“
      B = “GTAA”
      Terjadi substitusi dari karakter C menjadi G dan sisanya adalah insersi. Penentuan karakter yang terinsersi selalu diawal string.

Untuk menghasilkan basis yang selalu memenuhi kemungkinan-kemungkinan diatas, maka proses *divide* atau pembagiannya harus diatur. Pembagiannya dilakukan dengan membagi string menjadi dua bagian yang **tidak selalu** sama panjang.

C. Dynamic Programming Algorithm (DPA) untuk Edit-Distance

Algoritma DPA untuk *edit-distance* ini yang akan dipakai untuk proses *divide* pada algoritma Hirschberg. Algoritma ini sebenarnya digunakan untuk mencari jarak kemiripan dua string. *Edit-Distance* disini adalah jumlah titik mutasi yang dibutuhkan untuk mengubah suatu string menjadi string lain [7]. Misalkan d adalah fungsi dari algoritma ini maka jika terdapat dua string yaitu s1 dan s2, maka fungsi dari algoritma ini yaitu:

d(‘’,’’) = 0 (basis)
d(s,’’) = d(‘’,s) = |s| (basis)
d(s1+ch1, s2+ch2) =
    min(d(s1,s2) + if ch1=ch2 then 0 else 1,
        d(s1+ch1,s2)+1,

d(s1,s2+ch2)+1) (rekurens)

Dua persamaan pertama sudah pasti benar, sehingga yang perlu dijelaskan adalah persamaan ketiga. Pada persamaan ketiga ch1 dan ch2 adalah karakter terakhir dari s1 dan s2. Jika ch1 sama dengan ch2, maka tidak ada tambahan jarak antar s1 dan s2. Tetapi apabila ch1 berbeda dengan ch2 maka terdapat tambahan jarak 1. Kemungkinan lain adalah penghapusan ch1 dan penggantian s1 menjadi s2+ch2, jaraknya adalah d(s1,s2+ch2)+1. Kemungkinan terakhir adalah penggantian s1+ch1 menjadi s2, kemudian insersi ch2, jaraknya adalah d(s1+ch1,s2)+1. Dari ketiga kemungkinan tersebut dicari jarak yang paling minimum.

Dengan metode rekurensi diatas, ini akan melambatkan penghitungan, karena kita tahu bahwa algoritma rekursif serperti diatas diselesaikan dengan waktu eksponensial, sehingga lambat untuk string terdiri dari banyak karakter. Dari fungsi diatas dapat dilihat bahwa d(s1,s2) hanya bergantung pada d(s1`,s2`) yang s1` lebih kecil satu karakter dari s1 atau s2` lebih kecil satu karakter dari s2. Dengan karakteristik ini, bisa dipakai teknik program dinamis (*dynamic programming*) untuk menyelesaikan permasalahan ini.

Untuk menyimpan hasil dari tiap tahap dari program dinamis ini, kita akan memakai matriks dua dimensi dengan ukuran m[0..|s1|][0..|s2|]. Dari matriks ini berarti, s2 disejajarkan dengan kolom matriks, dan s1 disejajarkan dengan baris matriks. Metode penyimpananannya sebagai berikut ini.

m[i][j] = d(s1[1..i],s2[1..j])

Inisialisasinya:
m[0][0] = 0
m[i][0] = i, dengan i = 1..|s1|
m[0][j] = j, dengan j = 1..|s2|

Tahap selanjutnya:
m[i,j] = min(m[i-1][j-1] + if s1[i]=s2[j] then 0 else 1,
m[i-1][j] + 1,
m[i,j-1] + 1), dengan i=1..|s1|, j=1..|s2|

Dengan teknik program dinamis ini baris pada m[i][..] hanya bergantung pada baris m[i-1][..]. Kompleksitas algoritma ini adalah O(|s1|*|s2|), lebih baik dari eksponensial. Hasil jarak dapat diperoleh dari m[i,j] dengan i=|s1| dan j=|s2|.

D. Proses Divide Algoritma Hirschberg

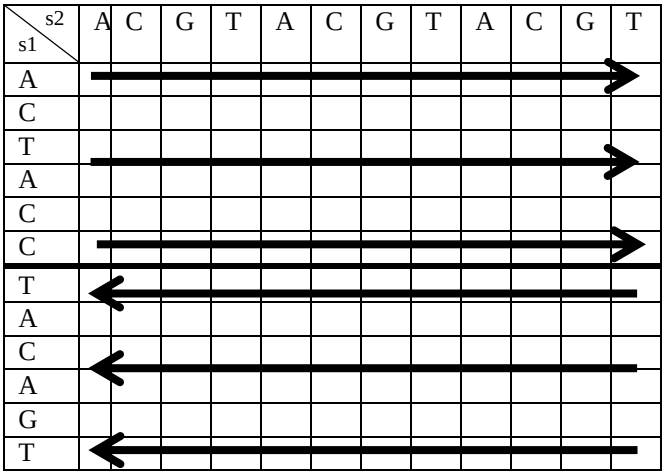
Setelah penjelasan panjang diatas, bagaimana algoritma diatas dapat membantu proses *divide* pada algoritma hirschberg. Ingat bahwa satu baris pada matriks ‘m’ bisa dihitung dengan baris sebelumnya. Dengan demikian tidak perlu matriks dengan panjang |s1|*|s2| untuk menyimpannya. Hanya diperlukan minimal matriks sebesar 1*|s2| untuk menyimpan hasil *edit-distance*. Algoritma DPA untuk *edit-distance* diatas bisa berjalan secara mundur dari m(|s1|+1,|s2|+1) menuju m(1,1), dengan hasil jarak yang sama. Lihat bahwa d(reverse(s1),reverse(s2)) = d(s1,s2). *Reverse* disini adalah membalik suatu string dari belakang ke depan.

Karakteristik algoritma DPA untuk *edit-distance* yang bisa berjalan dalam dua arah ini yaitu maju dan mundur, dimanfaatkan pada algoritma hirschberg. Pada proses *divide* atau pembagian, dengan memanfaatkan matriks m dari hasil DPA, pembagian harus dilakukan dengan dua cara, yaitu membagi baris s1 dan membagi kolom s2. Misalkan s1=’ACTACCTACAGT’ dan s2=’ ACGTACGTACGT’. Maka matriksnya akan seperti berikut.

s1 \ s2	A	C	G	T	A	C	G	T	A	C	G	T
A												
C												
T												
A												
C												
C												
T												
A												
C												
A												
G												
T												

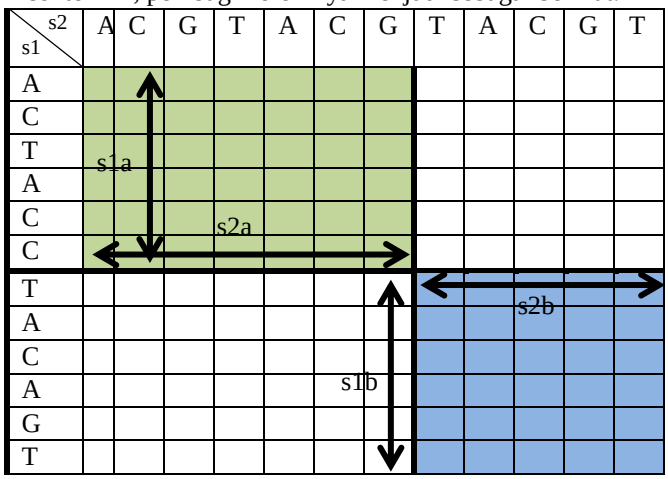
Gambar 3 Matriks *Edit-distance*

Untuk membagi baris s1 dapat dilakukan dengan membagi dua sama rata sesuai panjangnya seperti gambar 3. Misalkan hasil pembagiannya s1a pada bagian atas, dan s1b pada bagian bawah. Sedangkan untuk membagi kolom s2, maka digunakan algoritma DPA *edit-distance* untuk menghasilkan semua nilai jarak (*edit-distance*) pada bagian tengah baris matriks. Untuk contoh diatas harus diketahui nilai jarak (*edit-distance*) pada baris ke-6, dengan cara menggunakan fungsi DPA unutk *edit-distance* di atas dengan string pertama adalah s1a dan string kedua adalah s2 dengan arah maju. Baris selanjutnya yaitu baris ke-7, dengan cara menggunakan fungsi DPA untuk *edit-distance* dengan string pertama adalah s1b dan string kedua adalah s2 dengan arah mundur.



Gambar 4 Proses maju dan mundur pada algoritma DPA

Setelah didapatkan hasil semua jarak pada baris ke tengah yaitu baris ke-6 dan ke-7, untuk membagi kolomnya, kita telusuri setiap kolom cari yang jumlah nilai jarak pada baris tengah dengan nilai minimum. Dalam contoh ini, misalkan nilai jarak pada baris ke-6 disimpan dalam fwd[1..|s2|] dan nilai jarak pada baris ke-7 disimpan dalam rev[1..|s2|]. Telusuri dari i=1..|s2|, cari min(fwd[i]+rev[i]). Setelah ditemukan minimumnya, posisi minimum ini sebagai pembagi kolomnya. Pada contoh ini, pembagi kolomnya menjadi sebagai berikut.



Gambar 5 Pembagian s1 dan s2

Pembagian s2 tersebut menjadi s2a untuk sebelah kiri dan s2b untuk sebelah kanan.

E. Proses Conquer Algoritma Hirschberg

Proses *conquer*-nya adalah menyelesaikan 2 buat area pada matriks. Area pertama yaitu menyelesaikan dengan algoritma hirschberg dengan string pertama adalah s1a dan string kedua adalah s2a. Area yang kedua yaitu menyelesaikan dengan algoritma hirschberg dengan string pertama adalah s1b dan string kedua adalah s2b.

Dengan menggabungkan setiap string hasil penjajaran yang didapat pada tahap basis dari rekursif, maka didapatkan string total hasil penjajaran dan dapat diketahui jenis mutasi apa saja yang terjadi.

F. Kompleksitas Algoritma Hirschberg

Memori yang digunakan dalam algoritma *edit-distance* adalah O(|s2|). Sedangkan waktu yang digunakannya adalah O(|s1|*|s2|), tetapi rekursif harus dijalankan ketika dipakai di algoritma Hirschberg. Pada saat menjalankan rekursif yang kedua, program hanya menjalankan separuh dari rekursif yang pertama, sehingga membutuhkan waktu O(|s1|*|s2|/2). Kemudian pada rekursif selanjutnya, program menjalankan separuh dari rekursif sebelumnya dan seterusenya, sehingga total waktu yang dibutuhkan yaitu O(|s1|*|s2|*(1+1/2+1/4+…)) yang mana sama dengan O(2*|s1|*|s2|). Penggunaan rekursif ini mengurangi kompleksitas memori yang digunakan dari O(|s1|*|s2|) menjadi O(|s2|) tetapi waktunya menjadi dua kali lipat [6].

III. IMPLEMENTASI

Implementasi yang dibuat oleh penulis menggunakan output berupa *console* dengan menggunakan bahasa pemrograman C#. Output dari program ini adalah menampilkan jenis mutasi yang terjadi, statistik mutasi, dan penjajaran dua untaian DNA. Penulis menggunakan suatu contoh fragmen untain DNA dengan jumlah 132 basa nitrogren sebagai DNA yang asli. Kemudian untuk DNA yang termutasi, penulis menggunakan contoh fragmen DNA dengan jumlah 133 basa nitrogen. Hasil dari pengujian program sebagai berikut.

Nama file: dna.cs

Jenis mutasi yang terjadi yaitu:

```
Terjadi insersi T pada urutan ke-3
Terjadi insersi A pada urutan ke-14
Terjadi substitusi C menjadi A pada urutan ke-19
Terjadi insersi G pada urutan ke-25
Terjadi substitusi T menjadi C pada urutan ke-27
Terjadi substitusi A menjadi C pada urutan ke-28
Terjadi substitusi G menjadi C pada urutan ke-37
Terjadi substitusi T menjadi C pada urutan ke-38
Terjadi substitusi G menjadi C pada urutan ke-59
Terjadi substitusi T menjadi G pada urutan ke-64
Terjadi substitusi C menjadi A pada urutan ke-88
Terjadi substitusi A menjadi T pada urutan ke-104
Terjadi substitusi A menjadi G pada urutan ke-105
Terjadi substitusi C menjadi G pada urutan ke-117
Terjadi delesi C pada urutan ke-124
Terjadi delesi C pada urutan ke-125
Terjadi substitusi C menjadi A pada urutan ke-132
Terjadi substitusi C menjadi A pada urutan ke-133
Terjadi substitusi C menjadi A pada urutan ke-134
```

Gambar 6 Jenis mutasi hasil pengujian

Statistik mutasinya:

```
-Jumlah insersi: 3
-Jumlah delesi: 2
-Jumlah substitusi: 14
-Jumlah total mutasi: 19
```

Gambar 7 Statistik mutasi

Hasil penjajarannya:

```
Hasil alignment:
TT-GACTCACCCA-TCAACAACCG-CTATGTATTTCGTACATTACTGCCAGTCACCATGAATATTGTACGGTACCATAAATACTTGACCACCTGTAGTAC
ATAAAAACTCAACCCACATCAAACCCCCCCCCCCC
TTTGACTCACCCAATCAAAAACCGGCCCTGTATTTCCCACATTACTGCCAGTCACCATCAATAGTGTACGGTACCATAAATACTTGAACACCTGTAGTAC
ATATGAACTCAACCCAGATCAAA--CCCCCCAAAC
```

Gambar 8 Hasil penjajaran

DNA yang atas adalah DNA yang asli, sedangkan yang bawah adalah DNA yang termutasi.

Adapun beberapa fungsi penting yang harus dibuat dalam mengimplementasikan algoritma ini:

a. alignHirschberg

Deskripsi:

Mencari penjajaran antar dua untaian DNA dan mencari jenis-jenis mutasi yang terjadi.

Masukan:

- p1 adalah indeks awal string 1
- p2 adalah indeks akhir string 1
- q1 adalah indeks awal string 2
- q2 adalah indeks akhir string 2

Keluaran:

Menampilkan jenis mutasi yang terjadi

b. fwdDPA

Deskripsi:

Mencari jarak (*Edit-distance*) antar 2 string dengan algoritma program dinamis secara maju.
Masukan:
- p1 adalah indeks awal string 1
- p2 adalah indeks akhir string 1
- q1 adalah indeks awal string 2
- q2 adalah indeks akhir string 2

Keluaran:
Menghasilkan matriks yang merupakan nilai jarak (*edit-distance*) pada bagian tengah baris matriks penjajaran.

c. revDPA

Deskripsi
Mencari jarak (*Edit-distance*) antar 2 string dengan algoritma program dinamis secara mundur dari belakang ke depan.
Masukan:
- p1 adalah indeks awal string 1
- p2 adalah indeks akhir string 1
- q1 adalah indeks awal string 2
- q2 adalah indeks akhir string 2

Keluaran:
Menghasilkan matriks yang merupakan nilai jarak (*edit-distance*) pada bagian tengah baris matriks penjajaran.

Pada awal pemanggilan, fungsi alignHirschberg dipanggil dengan format:

alignHirschberg(0,string1.length,0,string2.length)

IV. KESIMPULAN

Algoritma ini bisa digunakan oleh para peneliti rekayasa genetika untuk mengidentifikasi mutasi yang terjadi pada untaian DNA. Meskipun ada algoritma lain yang dapat digunakan dengan fungsi yang sama, tetapi algoritma ini memiliki beberapa kelebihan dibanding algoritma lain. Kelebihan ini yaitu terletak pada penggunaan memori yang sangat efisien, sehingga cocok dipakai untuk komputer yang memorinya tidak terlalu "super" meskipun harus dibayar dengan kecepatan penyelesaian yang lebih lama. Bayangkan saja apabila panjang untaian DNA yang akan dianalisa sepanjang 100.000 rantai. Perlu diketahui bahwa panjang untaian DNA sebenarnya biasanya sepanjang 100 rantai sampai 100.000 rantai [6]. Misalkan saja untuk penyimpanan 1 rantai atau 1 basa nitrogen disimpan dalam 1 byte karakter. Pada algoritma lain contohnya algoritma Needleman-Wunsch, algoritma ini memiliki kompleksitas memori O(|s1|*|s2|) dan kompleksitas waktu O(|s1|*|s2|). Memori yang dibutuhkan dengan algoritma Needleman-Wunsch ini yaitu 100.000 byte x 100.000 byte. Angka ini mencapai 9,3 GB. Sedangkan memori RAM yang dijual saat ini kebanyakan maksimal 4 GB, sehingga untuk peneliti yang tidak punya komputer yang memorinya "super" akan kesulitan dalam menganalisis masalah ini. Akan tetapi dengan algoritma Hirschberg, memori yang dibutuhkan hanya 100.000 byte, ini jauh lebih "irit" dari algoritma lain dan dapat dipakai oleh komputer-komputer biasa. Untuk masalah waktu mungkin algoritma Hirschberg memakan dua kali lipat lebih dari yang lain, tetapi untuk peneliti yang spesifikasi komputernya "biasa" waktu ini masih bisa ditunggu, ketimbang peneliti harus memodifikasi komputernya sehingga bisa dipasang memori RAM yang besar atau menyewa komputer super dengan biaya yang mahal.

Dengan kelebihan dan kekurangan algoritma ini, semoga para peneliti dalam bidang informatika bisa berlomba-lomba untuk mencari algoritma yang seefisien mungkin dalam hal waktu dan memori untuk menyelesaikan masalah analisa mutasi DNA ini. Ini adalah PR besar bagi ahli informatika, supaya bisa membantu para peneliti rekayasa genetika. Mungkin saja suatu saat dengan teknologi dan algoritma analisis DNA ini, suatu kelainan atau penyakit yang disebabkan oleh mutasi DNA bisa terobati. *Reward* yang besar diberikan untuk ahli informatika meskipun tidak bisa menangani penyakit itu secara fisik, tetapi dengan algoritmanya bisa sangat membantu.

Demikian kesimpulan dari makalah yang penulis buat apabila ada kesalahan dalam tulisan makalah ini mohon dimaklumi dan dimaafkan.

REFERENSI

[1] R. Munir, "Strategi Algoritma", Teknik Informatika, Bandung, 2009.
[2] D. P. Putra, "Apilkasi Dynamic Programming dalam Algoritma Needleman-Wunsch untuk Penjajaran DNA dan Protein". Bandung, Teknik Informatika ITB, 2009.
[3] http://ghr.nlm.nih.gov/handbook/illustrations/insertion, waktu akses 29 November 2010, 16:30 WIB
[4] http://ghr.nlm.nih.gov/handbook/illustrations/deletion, waktu akses 29 November 2010, 16:30 WIB
[5] http://ghr.nlm.nih.gov/handbook/illustrations/nonsense, waktu akses 29 November 2010, 16:30 WIB
[6] http://www.csse.monash.edu.au/%7Elloyd/tildeAlgDS/Dynamic/Hirsch/, waktu akses 29 November 2010, 16:30 WIB
[7] http://www.csse.monash.edu.au/%7Elloyd/tildeAlgDS/Dynamic/Edit/ , waktu akses 29 November 2010, 16:30
[8] http://www.gudangmateri.com/2010/07/mutasi-gen-dan-mutasi-kromosom.html, waktu akses 29 November 2010, 16:30
[9] http://en.wikipedia.org/wiki/Hirschberg_algorithm, waktu akses 29 November 2010, 16: 30

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, 29 April 2010

Achmad Baihaqi
13508030