DYNAMMIC PROGRAMMING
DALAM MENENTUKAN ARTI URUTAN UNTAIAN GEN

David Soendoro

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Alamat: Jalan Ganeca No. 10 Bandung
e-mail: david.soendoro@gmail.com

ABSTRAK

Gen yang ditemukan dalam protein atau *DNA* (*Deoxyribonucleic acid*) merupakan urutan kode yang memberi arti sebagai sifat dasar makhluk hidup. Sebagai komponen terkecil kehidupan, pengartian *DNA* sangatlah sulit dan memiliki basis data yang sangat besar. Salah satu metode memastikan arti suatu sifat *DNA* adalah dengan membandingkannya dengan *DNA* yang telah diverifikasi sebelumnya yang disebut dengan *sequence alignment* atau penyusunan terurut. Seiring bertambahnya jumlah *DNA* yang diverifikasi maka lamanya proses pencarian juga meningkat dengan tajam. *Dynamic programming* dengan berbasiskan algoritma Needleman-Wunsch merupakan cara tercepat yang telah ditemukan untuk mengetahui arti suatu untaian protein tersebut karena dapat mengenali apakah untaian yang diproses akan menuju pada untaian yang dibandingkan atau tidak. Pembandingan ini dilakukan dengan menghitung kesamaan tiap jenis gen *DNA* yaitu A, T, G, atau C dan jaraknya. Jarak perlu diperhitungkan karena informasi yang dibawa *DNA* tidak seperti informasi buatan manusia, 2 buah untaian berbeda bisa saja memiliki arti yang sama karena terdapat banyak *junk DNA* yang merupakan dasar teori evolusi, lewat algoritma ini juga kita dapat menentukan kapan tepatnya terjadi evolusi makhluk hidup secara fungsional, struktural, bahkan pola hidup.

Kata kunci: Dynamic Programming, Needleman-Wunsch, sequence alignment, DNA, gen.

1. PENDAHULUAN

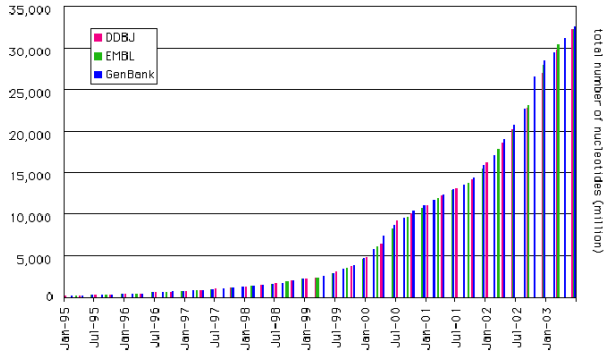
Sequence alignment merupakan hal yang akan dibahas pada makalah ini, *sequence alignment* berasal dari bahasa Inggris yang arti semantiknya adalah penjajaran urutan. Inti dari proses ini adalah mencari "kemiripan" antar dua buah untaian protein atau bisa dikatakan rentetan informasi. Kemiripan tersebut perlu diindikasikan agar kita dapat mengetahui sifat dari untaian protein tersebut karena untaian protein terus berubah akibat dari evolusi, mutasi, dan berbagai pengaruh kimia atau biologi lainnya.

```
AAB24882        TYHMCQFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPT 60
AAB24881        --------------------YECNQCGKAFAQHSSLKCHYRTHIGEKPYECNQCGKAFSK 40
                                     ****: .***:  * *:** * :****.:* ********..

AAB24882        PSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHTGEKPYE-CNQCGKAFAQ- 116
AAB24881        HSHLQCHKRTHTGEKPYECNQCGKAFSQHGLLQRHKRTHTGEKPYMNVINMVKPLHNS 98
                 **** *:***********:***:**.: .***************    :  *.: :
```

Gambar 1 – Contoh dari *sequence alignment* 2 buah protein zinc dari jari manusia

Bagaimanapun, dalam pembelajaran tentang gen para ilmuwan terus menyimpan arti untaian protein yang telah terverifikasi ke dalam basis data. Seiring dengan majunya ilmu genetis, basis data gen pun meningkat secara eksponensial, dapat dilihat peningkatan ukuran basis data yang diambil dari GenBank, DDBJ (DNA DataBank of Japan), dan EMBL (European Molecular Biology Laboratory).



Gambar 2 – Perkembangan basis data gen dari Januari 1995 hingga Januari 2003

Basis data sejumlah besar ini harus diakses seluruhnya pada saat mencoba mengidentifikasi sebuah untaian protein baru, dapat dibayangkan berapa lama waktu yang dibutuhkan bila kita tidak memiliki algoritma yang mangkus.

Menjawab tantangan di atas, kita dapat menggunakan algoritma *Dynamic Programming* Needleman-Wunsch dalam mengenali untaian tersebut. Adapun algoritma ini menggunakan prinsip matematika sederhana yaitu menggunakan matriks singularitas berdasarkan hasil penelitian di mana gen A mudah

bermutasi menjadi G dan C menjadi T begitu pula sebaliknya sehingga membentuk matriks seperti tersebut

-	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

Gambar 3 – Matriks singularitas gen

Selanjutnya apabila kita membandingkan 2 buah untaian kita dapat menjumlahkan perbedaan-perbedaannya semakin besar nilainya maka semakin besar kemungkinan kedua untaian tersebut memiliki sifat yang sama atau merupakan hasil mutasi dari untaian yang lain.

2. METODE

2.1 Algoritma Needleman-Wunsch

Kita dapat menemukan kemiripan antar untaian gen dengan menggunakan algoritma Needleman-Wunsch. Algoritma Needleman-Wunsch adalah algoritma yang merepresentasikan kemiripan dua buah untaian informasi secara keseluruhan dengan membuat kedua buah untaian menjadi sebuah tabel. Adapun algoritma ini terdiri dari 3 tahap penting yakni:

1. Inisialisasi
2. Mengisi Matriks
3. Runut Balik (*Traceback*)

Hal yang harus diperhitungkan juga adalah jarak kosong atau gap pada untaian gen, oleh karenanya sesuai dengan prinsip *Dynamic Programming* maka harus ada beberapa hitungan matematis dalam tiap pengambilan keputusan, begitu pula dengan algoritma Needleman-Wunsch. Berikut adalah persamaan yang akan digunakan:

$$S_{i,j} = \text{diambil dari matriks singularitas} \quad (1)$$

$$w = \text{ongkos jarak kosong (gap)} \quad (2)$$

dan

$$M_{i,j} = \text{MAXIMUM}[M_{i-1,\,j-1} + S_{i,j},\ M_{i,j-1} + w,\ M_{i-1,j} + w] \quad (3)$$

Apabila terdapat tabel ditulis seperti contoh berikut:

Tabel x Judul tabel ditulis dengan huruf Times New Roman, 9 pt, Bold, center

2.2 Inisialisasi

Langkah pertama dari membuat *global alignment dynamic programming* adalah dengan membuat matriks dengan besar (M+1) x (N+1) di mana M adalah panjang untaian pertama dan N adalah panjang untaian kedua.

Contoh mengenai metode sequence alignment dengan algoritma Needleman-Wunsch akan diberikan secara berurut sepanjang makalah ini. Berikut adalah contoh inisialisasi dari 2 buah untaian gen di mana:

Gen 1: GAATTCAGTTA

Gen 2: GGATCGA

Ongkos jarak kosong (w) = -5

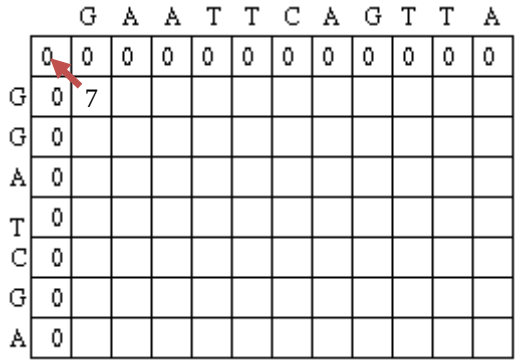
		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Gambar 4 - Inisialisasi Matriks

2.3 Mengisi Matriks

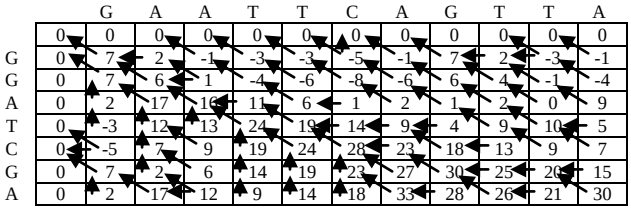
Dalam melakukan pengisian matriks terdapat beberapa cara, cara paling mudah dan awal digunakan dalam algoritma Needleman-Wunsch adalah mengisi penuh seluruh matriks dari kiri atas dengan kompleksitas O(mn). Pada perkembangannya terdapat algoritma Hirschberg yang mereduksi kompleksitas pengisian matriks ini menjadi O(m+n) namun algoritma ini tidak akan dibahas di sini, pada metode yang tertulis akan digunakan metode pengisian dari kiri atas.

Sesuai dengan matriks singularitas pada Gambar 3 dan persamaan (3) kita akan dapat mengisikan matriks tersebut. Langkah pertama adalah mencari $M_{1,1}$ = MAXIMUM[(0 + 7), (0 + -5), (0 + -5)] = 7 maka $M_{1,1}$ = 7.



Gambar 5 - Pengisian Matriks 1,1

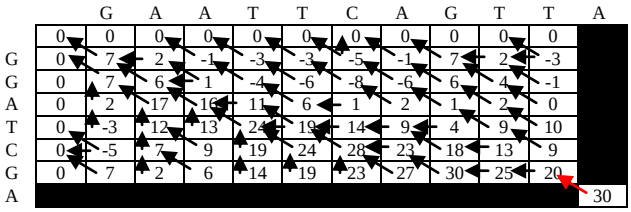
Karena nilai maksimum berasal dari $M_{i-1,j-1} + S_{i,j}$ maka nilai *traceback* yang disimpan adalah ke kiri atas, apabila nilai berasal dari $M_{i,j-1} + w$ maka *traceback* adalah atas dan $M_{i-1,j} + w$ maka *traceback* adalah kiri. Matriks akan terus diisi hingga seluruh matriks terpenuhi.



Gambar 6 - Hasil pengisian penuh matriks

2.4 Runut Balik (*Traceback*)

Setelah mengisi penuh matriks maka langkah berikutnya yang harus kita lakukan adalah mencari persamaan terbaik dengan menyusuri matriks tersebut dari kanan bawah ke kiri atas untuk mencari nilai maksimum

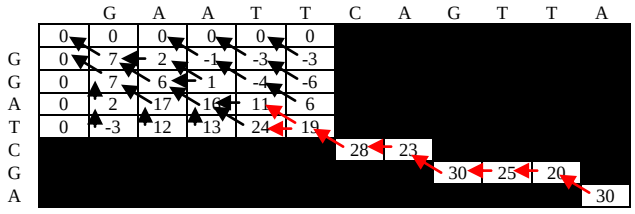


Gambar 7 – Runut balik pertama

Langkah pertama dari runut balik adalah dari kanan bawah ke kiri atasnya sesuai dengan runut balik yang telah didapatkan dari pemenuhan matriks. Kemudian kita dapat memastikan bahwa letak alignment yang cocok dari untaian pertama dan kedua adalah A di akhir berada pada urutan yang sama, atau kita dapatkan:

```
A
|
A
```

Selanjutnya runut balik akan mengikuti lajur yang telah dibuat sebelumnya, apabila ada beberapa kemungkinan nilai maksimal maka kedua nilai tersebut harus ditelusuri karena memang algoritma *dynamic programming* dapat memberikan lebih dari 1 solusi.



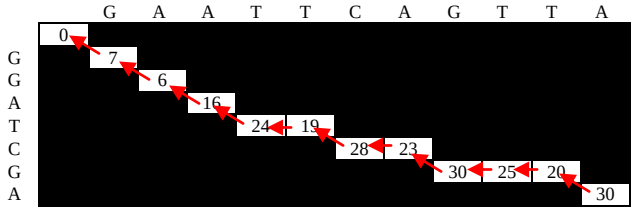
Gambar 8 – Runut balik pada persimpangan

Dari runut balik sejauh ini kita telah memperoleh

```
T C A G T T A
| |   |     |
T C _ G _ _ A
```

Di mana apabila kita menemui persimpangan ke kiri atas mencatatnya dan bila persimpangan ke kiri atau ataus kita menganggapnya sebuah *gap*. Berikut di bawah ini merupakan kedua solusi yang diperoleh dari *sequence alignment* kedua untaian tersebut.

Jawaban pertama:



Gambar 9 – Kemungkinan pertama

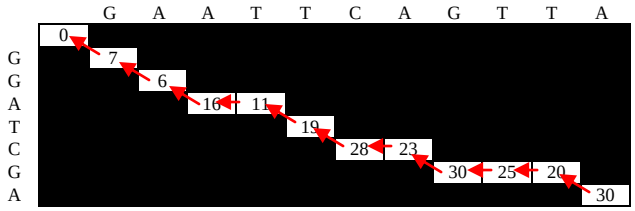
Untaian termirip pertama

```
G A A T T C A G T T A
|   | |   |   |     |
G G A T _ C _ G _ _ A
```

Memiliki skor:

0 + 7 – 1 + 10 + 8 – 5 + 9 – 5 + 7 – 5 – 5 + 10 = **30**

Jawaban kedua:



Gambar 10 – Kemungkinan kedua

Untaian termirip kedua

```
G A A T T C A G T T A
|   |   | |   |     |
G G A _ T C _ G _ _ A
```

Memiliki skor:

0 + 7 – 1 + 10 – 5 + 8 + 9 – 5 + 7 – 5 – 5 + 10 = **30**

3. IMPLEMENTASI

3.1 Implementasi Sederhana

Implementasi dapat dibuat dalam format GUI ataupun *command prompt*, bagaimanapun yang coba penulis buat di sini adalah sebuah implementasi algoritma Needleman-Wunsch sederhana dengan antarmuka *command prompt* dengan menggunakan bahasa Java dengan masukkan dua buah untaian informasi dalam format *string* dan keluaran berupa hasil *sequence alignment* seperti pada bab 2.

Nama File: Needleman-Wunsch.Java

```
0       0       0       0       0       0       0       0
0       7       7       2       -3      -5      7       2
0       2       6       17      12      7       2       17
0       -1      1       16      13      9       6       12
0       -3      -4      11      24      19      14      9
0       -3      -6      6       19      24      19      14
0       -5      -8      1       14      28      23      18
0       -1      -6      2       9       23      27      33
0       7       6       1       4       18      30      28
0       2       4       2       9       13      25      26
0       -3      -1      0       10      9       20      21
0       -1      -4      9       5       7       15      30

GAATTCAGTTA

GGA-TC-G--A
```

Gambar 11 - Tampilan program

Adapun beberapa fungsi penting yang harus dibuat dalam mengimplementasi algoritma ini adalah:

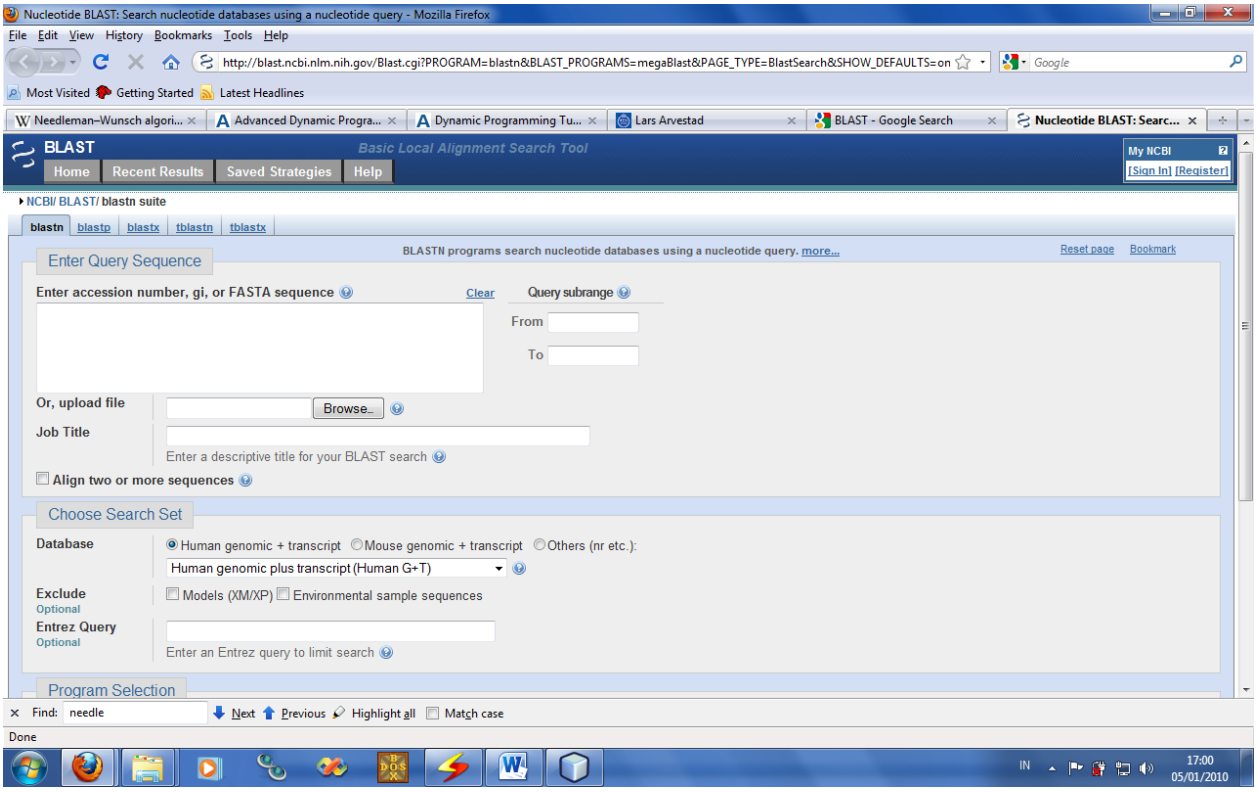
- **getAlignments**
  masukkan:
  untaian pertama, *array of integer* 1 dimensi dari masing-masing untaian dan *array of integer* 2 dimensi dari kedua untaian.
  keluaran:
  untaian hasil *sequence alignment.*
  deskripsi:
  runut balik *array of integer* 2 dimensi dari kedua untaian untuk mencari jawaban *sequence alignment*.
- **calculateMatrix**
  masukkan:
  array of integer untaian pertama dan kedua.
  keluaran:
  array of integer 2 dimensi dari kedua untaian yang telah diperhitungkan.
  deskripsi:
  mengisi seluruh matriks dengan nilai yang tepat sesuai rumus yang dijelaskan di bab 2.
- **convertStringToArr**
  masukkan:
  untaian gen
  keluaran:
  array of integer yang merepresentasikan untaian gen (contoh: A=0, G=1, C=2, T=3).
  deskripsi:
  mengkonversi untaian gen menjadi *array of integer* untuk mempermudah perhitungan.
- **simular**
  masukkan:
  dua buah integer yang merepresentasikan gen
  keluaran:
  mengambil nilai dari matriks singularitas berdasarkan gen yang diminta.
  deskripsi:
  mengembalikan nilai dari matriks singularitas sesuai dengan masukkan dua buah gen yang dibandingkan.

3.2 BLAST

BLAST atau singkatan dari **B**asic **L**ocal **A**lignment **S**earch **T**ool merupakan program yang telah ada dan digunakan untuk membandingkan sekuens informasi biologi, bermacam kegunaan dalam riset gen telah dibuat berdasarkan program tersebut. BLAST juga merupakan program sequence alignment pertama di dunia (dibuat oleh Myers E. dkk. Pada tahun 1990), yang menggunakan pengembangan algoritma Smith-Waterman dan memiliki sedikit kemiripan dengan algoritma Needleman-Wunsch.

Pada tahun 2009, BLAST kembali diluncurkan namun kini khusus untuk untaian protein dan diberi nama CS-BLAST. Penggunaan BLAST sangat banyak, hingga saat ini beberapa kegunaannya antara lain: identifikasi spesies, membuat pohon filogeni, memetakan DNA, pembandingan, dan melokasikan domains.

BLAST dapat diakses di http://blast.ncbi.nlm.nih.gov/Blast.cgi, berikut tampilan dari BLAST



Gambar 12 - Tampilan BLAST

4. KESIMPULAN

Bioinformatika merupakan salah satu cabang ilmu Teknik Informatika yang sangat berkembang belakangan ini bersamaan dengan berkembangnya ilmu bioteknologi dan genetika. Penemuan-penemuan yang sebelumnya tidak dibayangkan seperti rekayasa genetika kini dapat kita raih dalam masa depan yang tidak lama lagi (*near future*). Terima kasih untuk bioinformatika yang mempercepat pendataan untaian-untaian gen dan mempercepat kemampuan kita belajar genetika secara eksponensial sejak awal 1990an (berdasarkan statistik pada bab 1).

Terutama bagi para ahli bidang Teknik Informatika, ini merupakan kesempatan besar dan membuktikan besarnya andil Teknik Informatika dalam perkembangan kehidupan baik sains maupun lingkungan sekitar. Bayangkan suatu saat kita dapat membuat tanaman tahan hama, unggul, dan sering berbuah sekaligus karena kita telah berhasil memetakan gen tanaman tersebut.

Algoritma Needleman-Wunsch, Smith-Waterman, dan program BLAST merupakan beberapa penemuan yang telah ada pada dunia bioinformatika dan dapat terus kita kembangkan. Dengan mempelajari algoritma-algoritma tersebut penulis yakin suatu saat nanti akan tumbuh algoritma-algoritma baru yang berguna dan dapat mengubah dunia menjadi lebih baik.

Sekian kesimpulan dari makalah yang penulis buat, apabila pada makalah terdapat kesalahan baik disengaja maupun tidak agar dimaafkan.

REFERENSI

[1] Sergio Anibal de Carvalho Junior, "*Sequence Alignment Algorithms*", Department of Computer Science School of Physical Sciences & Engineering King's College London, 2003, 2-7.

[2] Kraulis, Per, "*Molecular Bioinformatics 2001, Uppsala University*", Stockholm Bioinformatics Center, 2001.

[3] http://blast.ncbi.nlm.nih.gov/Blast.cgi waktu akses: 5 Januari 2010 17.00 WIB

[4] http://ww.wikipedia.org/wiki/BLAST waktu akses: 5 Januari 2010 17.00 WIB

[5] http://ww.wikipedia.org/wiki/Sequence_alignment waktu akses: 5 Januari 2010 17.00 WIB