APLIKASI *DYNAMIC PROGRAMMING* DALAM ALGORITMA NEEDLEMAN-WUNSCH UNTUK PENJAJARAN DNA DAN PROTEIN

Dian Perdhana Putra - 13507096

Teknik Informatika ITB
Jl. Ganesha 10 Bandung
e-mail: if17096@students.if.itb.ac.id

ABSTRAK

Dynamic programming adalah salah satu algoritma yang digunakan untuk menemukan nilai optimal dari suatu permasalahan. Dalam dynamic programming, pemecahan suatu masalah dibagi menjadi beberapa langkah (step) atau tahapan (stage) sedemikian hingga solusi dari sebuah persoalan dapat dipandang sebagai serangkaian keputusan yang saling berkaitan.

Penerapan dynamic programming yang amat penting dalam bidang bioinformatika adalah penjajaran sequence DNA dan protein. Salah satu penerapan dynamic programming adalah pada algoritma Needleman-Wunsch untuk mencari best alignment dari penjajaran dua buah sequence. Makalah ini akan membahas tentang penggunaan algoritma Needleman-Wunsch, berikut dengan contoh penggunaannnya dalam penjajaran sequence DNA dan protein.

Kata kunci: *dynamic programming*, bioinformatika algoritma *Needleman-Wunsch*, *sequence*, *alignment*, *DNA*, protein.

1. PENDAHULUAN

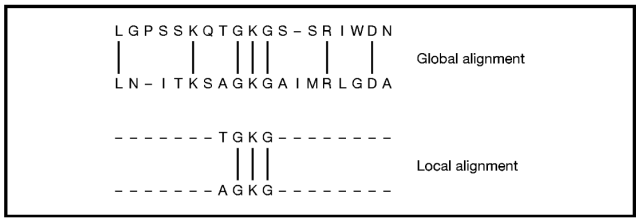
1.1 DNA dan Asam Amino

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. DNA dapat melakukan replikasi DNA, di mana ketika replikasi DNA ini dilakukan, terbentuklah DNA-DNA baru yang memiliki informasi genetik yang serupa dengan induknya.

Asam amino adalah suatu senyawa organik yang tersusun dari karbon, hidrogen, oksigen, dan nitrogen. Asam amino merupakan pembentuk dari protein dalam tubuh manusia dan diperlukan untuk metabolisme. Ada 20 macam asam amino yang diperlukan untuk membentuk protein dalam tubuh manusia, yaitu *Alanine (Ala,A), Arginine (Arg,R), Asparagine (Asn,N), Aspartate (Asp,D), Cysteine (Cys,C), Glutamine (Gln,Q), Glutamate (Glu,E), Glycine (Gly,G), Histidine (His,H), Isoleucin (Ile,I), Leucine (Leu,L), Lysine (Lys,K), Methionine (Met,M), Phenylalanine (Phe,F), Proline (Pro,P), Serine (Ser,S), Threonine (Thr,T), Tryptophan (Trp,W), Tyrosine (Tyr,Y),* dan *Valine (Val,V).* DNA menciptakan instruksi-instruksi yang diperlukan untuk sintesis asam amino.

1.2 Penjajaran Barisan

Sequence Alignment (penjajaran barisan) adalah prosedur untuk menjajarkan dua buah barisan (*sequence*) dari DNA atau protein dengan tujuan mencari kesamaan di antara barisan-barisan tersebut atau untuk membuktikan bahwa kedua *sequence* yang dibandingkan berasal dari *sequence* yang sama. Ada dua macam metode *sequence aligment*, yaitu *global alignment* dan *local alignement*.



Gambar 1. Perbedaan antara *global alignment* dan *local alignment* dari dua buah *sequence* [5]

Dalam *global alignment*, penjajaran dilakukan untuk keseluruhan *sequence*, yaitu dengan menggunakan sebanyak mungkin karakter dalam DNA. Sedangkan *local alignment* adalah menjajarkan sebagian dari *sequence-sequence*, biasanya bagian yang memiliki tingkat kemiripan yang cukup tinggi.

Dalam bioinformatika, *dynamic programming* digunakan sebagai metode untuk menjajarkan dua *sequence* protein atau DNA. *Dynamic programming* digunakan karena metode ini dapat memberikan hasil *alignment* yang paling optimal di antara *sequence-sequence* yang diberikan.

2. SKEMA PENILAIAN DAN MATRIKS SUBTITUSI

Residu (*residue*) adalah karakter dalam sebuah *sequence DNA*. Sebagai contoh : *sequence* ADEFGTW, maka residunya adalah A, D, E, F, G, T, dan W.

Misal, terdapat dua buah *sequence*, yaitu *sequence* I dan *sequence* II. *Subtitution score* adalah nilai yang didapat ketika sebuah residu dalam *sequence* I dibandingkan dengan residu dalam *sequence* II. Penalti celah kosong (*gap penalty*) adalah nilai yang diperoleh ketika dilakukan perbandingan antara residu dalam sebuah *sequence* dengan karakter kosong (*gap*) dalam *sequence* lainnya.

Skema penilaian (*scoring scheme*) adalah metode penilaian untuk mendapatkan nilai penjajaran (*alignment score*) dari penjajaran dua buah *sequence*. Skema penilaian ini bersifat *residue based*, yaitu perhitungan dari *alignment score* dipengaruhi oleh *residue substitution score*. *Alignment score* dari dua buah *sequence* adalah jumlah dari semua *substitution score* ditambah dengan jumlah dari semua *gap penalties*.

Contoh skema penilaian sederhana :
Diberikan nilai `+1` untuk *match* (residu dalam *sequence* I cocok dengan residu dalam *sequence* II), nilai `-1` untuk *mismatch* (residu dalam *sequence* I tidak cocok dengan residu dalam *sequence* II), dan nilai `0` untuk *gap penalty*. Misalkan terdapat *alignment* :

```
ATGGCGT
ATCGA-T
```

alignment score = `+1+1-1+1-1+0+1` = `2`

Matriks subtitusi (*substitution matrix*) adalah suatu matriks kesamaan (*similarity matrix*) yang digunakan untuk menyatakan *residue substitution score*. Matriks substitusi berukuran n × n, dimana n = 4 untuk DNA dan n = 20 untuk protein. Contoh dari matriks substitusi ini adalah BLASTN (*Basic Local Alignment Search Tool Nucleotide*) untuk DNA dan BLOSUM (*BLOck SUbstitution Matrix*) untuk protein.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Gambar 2 : BLOSUM62 *subtitution matrix*

3. PENCARIAN *BEST ALIGNMENT*

Best alignment adalah *alignment* yang memiliki nilai (*score*) tertinggi. Ada dua macam cara untuk menemukan *best alignment*, yaitu dengan menggunakan cara *brute force* dan menggunakan algoritma Needleman-Wunsch.

3.1 CARA *BRUTE FORCE*

Cara *brute force* untuk menemukan *best alignment* :

1. Cari semua *alignment* yang mungkin dari dua buah *sequence*, kemudian hitung *alignment score* untuk masing-masing *alignment* tersebut.
2. Pilih *alignment* dengan *alignment score* yang paling tinggi sebagai *best alignment*.

Jumlah *alignment* yang mungkin dari dua buah *sequence* dengan panjang N adalah :

$$\frac{2^{2^N}}{\sqrt{\pi N}} \qquad (1)$$

Untuk N yang sangat besar, pencarian *best alignment* dengan menggunakan cara *brute force* akan memakan waktu yang sangat lama.

3.2 ALGORITMA NEEDLEMAN-WUNSCH

Algoritma Needleman-Wunsch ditemukan pada tahun 1970 oleh Saul Needleman dan Christian Wunsch. Algoritma Needleman-Wunsch digunakan untuk menemukan *alignment* yang memiliki nilai optimal pada *global alignment* dari dua buah *sequence*. Untuk mencari *best alignment* menggunakan algoritma Needleman-Wunsch, digunakan sebuah matriks, yaitu matriks nilai (*score matrix*).

Algoritma ini dibagi menjadi tiga tahap :

1. **Inisialisasi :**
   Pilih matriks substitusi yang sesuai dengan jenis *sequence* yang dibandingkan, contoh *BLOSUM62 subtitution matrix* bila *sequence* yang dibandingkan adalah *sequence* protein. Misalkan *sequence* A memiliki panjang M dan *sequence* B memiliki panjang N, maka buat matriks nilai berukuran (M+1) × (N+1). Kemudian isi baris pertama dan kolom pertama matriks nilai dengan nilai dari *gap penalty*.

2. **Mengisi matriks :**
   Misalkan matriks nilai disebut matriks D, maka rumus untuk elemen dari matriks D, $D(i,j)$ adalah :

$$D(i,j) = \begin{cases} D(0,0) = 0 \\ max \begin{cases} D(i-1,j-1) + s(x_i, y_i) \\ D(i-1,j) + g_x \\ D(i,j-1) + g_y \end{cases} \end{cases} \quad (2)$$

dimana :

$s(x_i, y_i)$: elemen matriks substitusi di residu i pada *sequence* x dan residu j di *sequence* y.

$D(i-1, j-1)$: elemen matriks D di diagonal kiri atas .

$D(i, j-1)$: elemen matriks D di kiri $D(i,j)$.

$D(i-1, j)$: elemen matriks D di atas $D(i,j)$.

g_x : *gap penalty* untuk *sequence* x.

g_y : *gap penalty* untuk *sequence* y.

3. *Traceback Step*

Setelah matriks nilai berukuran (M+1) × (N+1) telah terisi sepenuhnya, maka *alignment score* maksimum dari dua buah *sequence* adalah nilai dari elemen paling kanan bawah dari matriks nilai,
yaitu $D(M+1, N+1)$.

Lakukan *traceback step* dengan menerapkan *dynamic programming* di sini, mulai dari $D(M+1, N+1)$.
hingga ke elemen pertama tabel, $D(0,0)$.

Misalkan ada dua buah *sequence*, yaitu *sequence* A dan *sequence* B, maka prosedur untuk melakukan *traceback step* adalah :

1. Buat dua buah *string*, disebut `alignmentA` dan `alignmentB`, untuk menyimpan *alignment* yang terbentuk. Jika *sequence* A lebih panjang dari *sequence* B, maka panjang `alignmentA` dan `alignmentB` sama dengan panjang *sequence* A. Sebaliknya, jika *sequence* B lebih panjang dari *sequence* A, maka panjang `alignmentA` dan `alignmentB` sama dengan panjang *sequence* B. Inisisalisasi `alignment` dan `alignmentB` dengan String kosong (“”).
2. Buat variabel pencacah untuk menyimpan *alignment score*, yaitu `alignmentScore`. Inisialisasi dengan 0.
3. Buat empat buah variabel bertipe bilangan bulat ,yaitu :
   `score`, untuk menyimpan $D(i,j)$
   `scoreDiag`, untuk menyimpan $D(i-1, j-1)$
   `scoreKiri`, untuk menyimpan $D(i, j-1)$
   `scoreAtas`, untuk menyimpan $D(i-1, j)$
4. Buat dua buah variable iterasi, `i` dan `j`. Inisialisasi `i` dengan nilai dari panjang *sequence* A dan `j` dengan nilai dari panjang *sequence* B.
5. Selama (`i`≥`0`) dan (`j`≥`0`), lakukan pengulangan sebagai berikut :
   - Jika `score=scoreDiag` + $S(A_{i-1}, B_{j-1})$, maka :
     `alignmentA` ← A(`i-1`) + `alignmentA`
     `alignmentB` ← B(`j-1`) + `alignmentB`
     `alignmentScore`← `alignmentScore` + $S(A_{i-1}, B_{j-1})$
     `i` ← `i - 1`
     `j` ← `j - 1`
   - Jika `score = scoreAtas` + g_x maka :
     `alignmentA` ← A(`i-1`) + `alignmentA`
     `alignmentB` ← “-” + `alignmentB`
     `alignmentScore`← `alignmentScore` + $S(A_{i-1}, B_{j-1})$
     `i` ← `i - 1`
   - Jika `score = scoreKiri` + g_y maka :
     `alignmentA` ← “-” + `alignmentA`
     `alignmentB` ← B(`j-1`) + `alignmentB`
     `alignmentScore`← `alignmentScore` + $S(A_{i-1}, B_{j-1})$
     `j` ← `j - 1`
6. Tuliskan *alignment* yang didapatkan.

3.2.1 Contoh penerapan algoritma Needleman-Wunsch

Misalkan, terdapat :
Sequence A : `IDEA`
Sequence B : `WIKIPEDIA`
Gap penalty untuk *sequence* A : -10
Gap penalty untuk *sequence* B : -10
Matriks substitusi, *S* (dibuat dengan mengacu pada *BLOSUM62 substitution matrix*) :

Tabel 1: Matriks subtitusi *S*

	D	**E**	**W**	**A**	**P**	**I**	**K**
D	6	2	-4	-2	-1	-3	-1
E	2	5	-3	-1	-1	-3	-1
W	-4	-3	11	-3	-4	-3	-3
A	-2	-1	-3	4	-1	-1	-1
P	-1	-1	-4	-1	7	-3	-1
I	-3	-3	-3	-1	-3	4	-3
K	-1	1	-3	-1	-1	-3	5

1. Inisialisasi

Panjang *sequence* A: 4
Panjang *sequence* B : 9
Buat matriks nilai *D* berukuran 5 × 10, indeks dimulai dari 0, isi dengan *gap penalties* :

Tabel 2 : Matriks nilai *D* awal

		W	I	K	I	P	E	D	I	A
	0	-10	-20	-30	-40	-50	-60	-70	-80	-90
I	-10									
D	-20									
E	-30									
A	-40									

Tabel 3 : Matriks nilai D yang telah terisi penuh

		W	I	K	I	P	E	D	I	A
	0	-10	-20	-30	-40	-50	-60	-70	-80	-90
I	-10	-3	-6	-16	-26	-36	-46	-56	-66	-76
D	-20	-13	-6	-7	-17	-27	-34	-40	-50	-60
E	-30	-23	-16	-5	-10	-18	-22	-32	-42	-51
A	-40	-33	-24	-15	-6	-11	-19	-24	-33	-38

2. Mengisi matriks

Untuk $D(1,1)$:

W

	0	-10
I	-10	

$D(0,0) = 0$
$D(0,1) = -10$
$D(1,0) = -10$
$S(I,W) = -3$

Maka :

$$D(1,1) = max\Big(\big(D(0,0) + (-3)\big), \big(D(0,1) + (-10)\big), \big(D(1,0) + (-10)\big)\Big)$$
$$D(1,1) = max\,(-3,-20,-20)$$
$$D(1,1) = -3$$

Untuk $D(1,2)$:

W **I**

	-10	-20
I	-3	

$D(0,1) = -10$
$D(0,2) = -20$
$D(1,1) = -3$
$S(I,I) = 4$

Maka :

$$D(1,2) = max\Big(\big(D(0,1) + 4\big), \big(D(0,2) + (-10)\big), \big(D(1,1) + (-10)\big)\Big)$$
$$D(1,2) = max\,(-6,-30,-13)$$
$$D(1,2) = -6$$

Pengisian matriks dilakukan hingga seluruh elemen matriks nilai terisi. Matriks yang terbentuk adalah sebagai berikut :

3. *Traceback Step*

Setelah matriks nilai D terisi penuh, maka *alignment score* optimal adalah -38, yaitu nilai dari D(4,9). *Traceback step* dilakukan untuk mendapatkan *alignment* dari *sequence* A dan *sequence* B yang memiliki *alignment score* sama dengan -38.

Untuk $D(4,9)$:

A

	-42	-51
A	-33	-38

$score = -38$
$score\ diagonal = -42 + S(A,A) = -42 + 4 = -38$
$score\ atas = -51 + gap\ sequence\ II$
$= -51 + (-10) = -61$
$score\ kiri = -33 + gap\ sequence\ I = -33 + (-10)$
$= -43$
Karena $score = score\ diagonal = -38$, maka
$i = i - 1 = 4 - 1 = 3$
$j = j - 1 = 9 - 1 = 8$
Sehingga sel selanjutnya adalah $D(3,8)$.

Untuk $D(3,8)$:

I

	-40	-50
E	-32	-42

$score = -42$
$score\ diagonal = -40 + S(I,E) = -40 + 3 = -37$
$score\ atas = -50 + gap\ sequence\ II$
$= -50 + (-10) = -60$
$score\ kiri = -32 + gap\ sequence\ I = -32 + (-10)$
$= -42$
Karena $score = score\ kiri = -42$, maka
$j = j - 1 = 8 - 1 = 7$
Sehingga sel selanjutnya adalah $D(3,7)$.

Traceback step dilakukan hingga elemen terakhir yaitu $D(0,0)$.

Tabel 4 : *best alignment* dari *sequence* A dan *sequence* B pada matriks nilai D

		W	I	K	I	P	E	D	I	A
	0	-10	-20	-30	-40	-50	-60	-70	-80	-90
I	-10	-3	-6	-16	-26	-36	-46	-56	-66	-76
D	-20	-13	-6	-7	-17	-27	-34	-40	-50	-60
E	-30	-23	-16	-5	-10	-18	-22	-32	-42	-51
A	-40	-33	-24	-15	-6	-11	-19	-24	-33	-38

Hasil *Alignment*

Sequence A : ---I-DE-A

| |

Sequence B : WIKIPEDIA

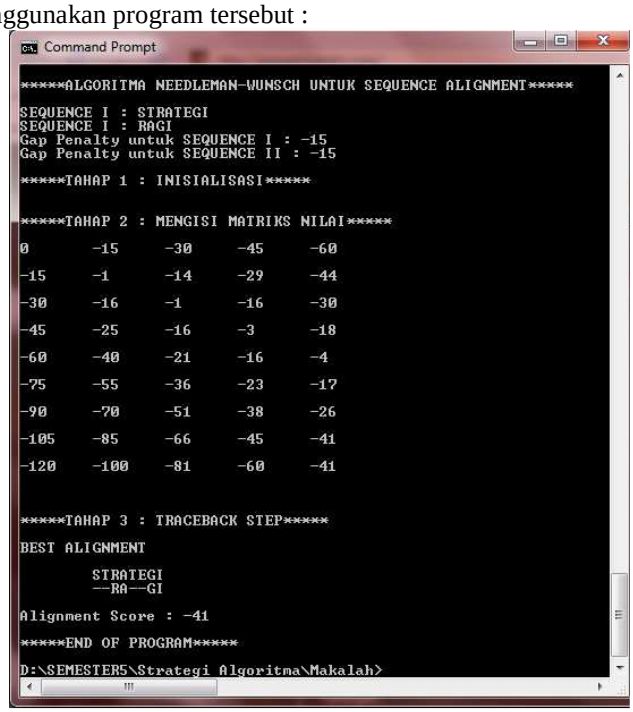
Alignment score =

$S(-,W) + S(-,I) + S(-,K) + S(I,I) + S(-,P)$
$+S(D,E) + S(E,D) + S(-,I) + S(A,A)$
$= (-10) + (-10) + (-10) +$
$4 + (-10) + 2 + 2 + (-10) + 4$
$= -38.$

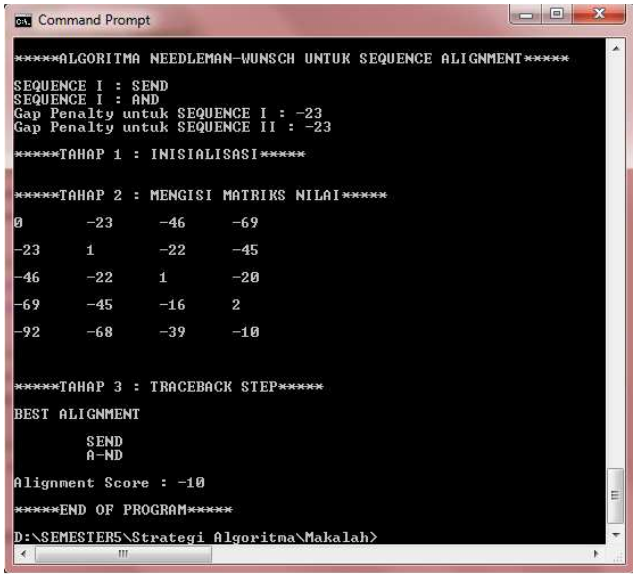
4. HASIL EKSPERIMEN

Penulis membuat program sederhana dalam bahasa Java yang mendemonstrasikan *alignment* untuk dua buah *sequence* protein. *Source code* program ini dapat diunduh di http://students.if.itb.ac.id/~if17096/ProgramStrategiAlgoritma.zip . Dalam program ini, matriks substitusi yang digunakan adalah *BLOSUM62 substitution matrix*.

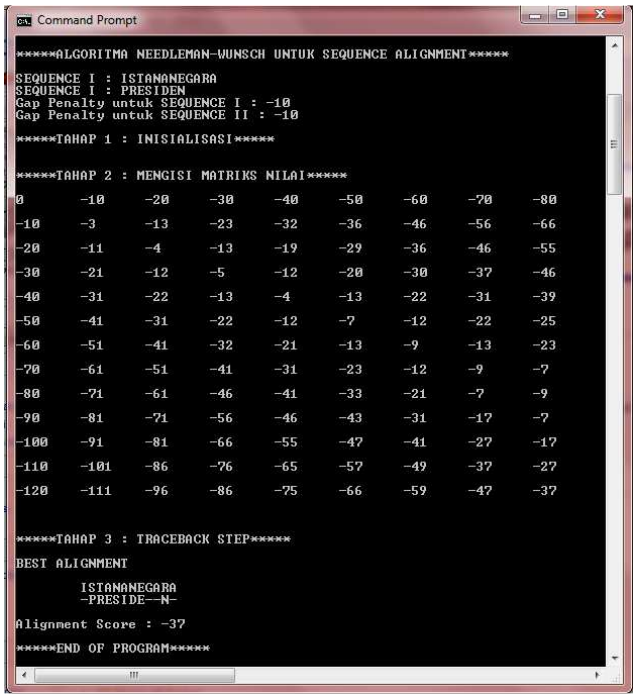
Berikut adalah beberapa hasil percobaan dengan menggunakan program tersebut :



Gambar 3 : Hasil Percobaan I



Gambar 4 : Hasil Percobaan II



Gambar 5 : Hasil Percobaan III

5. ANALISIS

Dari hasil percobaan, terbukti bahwa algoritma NeedlemanWunsch dapat digunakan untuk menemukan *best alignment*. Penyelesaian masalah dalam *traceback step* dibagi menjadi beberapa tahap atau langkah. Jumlah langkah yang terjadi sesuai dengan panjang dari *sequence* yang memiliki panjang terbesar.

Jika *sequence I* memiliki panjang m dan *sequence II* memilki panjang n, maka untuk tahap inisialisasi matriks nilai, kita perlu memasukkan nilai dari kolom pertama sebanyak m kali dan baris pertama sebanyak n kali. Pada tahap inisialisasi ini, kompleksitasnya adalah $O(m+n)$. Pada tahap mengisi matriks nilai, terdapat kalang bersarang, sehingga untuk mengisi matriks nilai, kompleksitas waktunya adalah $O(mn)$. Untuk *traceback step*, ada dua langkah (*step*) yang dilakukan yaitu menandai elemen dan menemukan *final path*. Untuk tahap menandai elemen, perpindahan yang mungkin dilakukan adalah sebanyak m baris dan n kolom, sehingga kompleksitasnya adalah $O(m+n)$. Sedangkan untuk menemukan *final path*, langkah ini melibatkan maksimum n langkah (dengan asumsi $n \geq m$). Oleh karena itu kompleksitasnya adalah $O(n)$. Jadi kompleksitas total dari algoritma Needleman Wunsch adalah :

$$O(m+n) + O(mn) + O(m+n) + O(n) = O(mn).$$

Pencarian best *alignment* dengan menggunakan algoritma brute force memiliki kompleksitas sebesar $O(m!\,n!)$ sehingga dengan demikian algoritma Needleman-Wunsch yang memiliki kompleksitas $O(mn)$ akan jauh lebih mangkus untuk m dan n yang sangat besar.

6. KESIMPULAN

1. *Dynamic programming* memiliki penerapan yang amat beragam dalam berbagai bidang keilmuan, salah satunya dalam bidang bioinformatika, yaitu pada algoritma Needleman-Wunsch untuk menemukan *best alignment* pada penjajaran *sequence* DNA dan protein.
2. Algoritma Needlaman-Wunsch dapat digunakan untuk mencari *best alignment* dari dua buah *sequence*, yaitu *alignment* yang memiliki *alignment score* maksimum.
3. Penggunaan algoritma Needleman-Wunsch untuk mencari best *alignment* dari dua buah *sequence* DNA atau protein akan jauh lebih efektif dan efisien daripada menggunakan *brute force*, terutama untuk *sequence-sequence* yang memiliki panjang sangat besar.
4. Algoritma Needleman-Wunsch memiliki kompleksitas $O(mn)$, sedangkan brute force memilki kompleksitas $O(m!\,n!)$.
5. Algoritma Needleman-Wunsch adalah algoritma yang cukup bagus untuk menemukan *best alignment* dari dua buah *sequence*. Di masa yang akan datang, algoritma ini dapat dikembangkan lagi sehingga memberikan hasil yang lebih baik.

7. REFERENSI

[1] Munir, Rinaldi. “Strategi Algoritma”. Teknik Informatika, Bandung, 2009.

[2] Böckenhauer, Hans-Joachim and Dirk Bongartz.“*Algorithmic Aspects of Bioinformatics*”, Springer, New York, 2007.

[3] Baxevanis, Andreas D. and B.F. Francis Oullette.”*BIOINFORMATICS* : *A Practical Guide to the Analysis of Genes and Proteins*, John Wiley & Sons,New York,2001.

[4] Lesk, Arthur.”*Introduction to Bioinformatics*”,Oxford University Press Inc,New York, 2002.

[5] Mount, David.”*Bioinformatics : Sequence and Genome Analysis*”,Cold Spring Harbor Laboratory Press,New York,2001.

[6] Barnes, Michael and Ian C. Gray.”*Bioinformatics for Geneticists*”, John Wiley & Sons,New York,2003.

[7] Jonassen, Inge and Kim Junhyong.”*Algorithms in Bioinformatics*”, Springer, New York, 2005.

[8] Cormen, Thomas H. et al.”*Introduction to Algorithms 2nd ed*”,The MIT Press,Cambridge.2001

http://www.avatar.se/molbioinfo2001/dynprog/dynamic.html
diakses pada 19 Desember 2009, pukul 20.13.

http://www.avatar.se/molbioinfo2001/dynprog/adv_dynamic.html
diakses pada 19 Desember 2009, pukul 20.35.

http://en.wikipedia.org/wiki/Dynamic_programming
diakses pada 18 Desember 2009 pukul 21.00

http://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm
diakses pada 18 Desember 2009 pukul 21.15