Program Dinamis untuk Menyelesaikan Sequence Alignment

Harum Lokawati 13515109
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13515109@std.stei.itb.ac.id

***Abstract*—Sequence Alignment merupakan salah satu topik dalam *bioinformatics* yang digunakan untuk menganalisis rantai asam amino pada DNA, RNA, ataupun protein. Hasil *alignment* dapat digunakan sebagai bahan analisis kesamaan atau keterhubungan secara fungsional, structural, maupun revolusi dari struktur yang berbeda-beda. Algoritma yang dapat digunakan untuk menyelesaikan masalah ini salah satunya adalah dynamic programming, yaitu dengan 2-dimensional DP. Pengimplementasian dynamic programming dilakukan dengan membuat fungsi sebagai *cost* yang digunakan untuk menentukan keputusan tiap langkahnya. Alogirtma lebih lengkap akan dibahas pada makalah ini.**

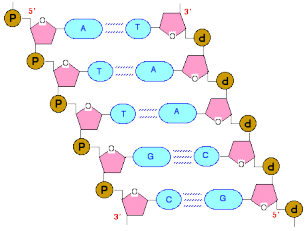
Keywords—dynamic programming, bioinformatics, DNA, algorithm, sequence alignment.

I. Pendahuluan

Semakin pesatnya perkembangan teknologi dan ilmu pengetahuan membuat manusia semakin dimudahkan oleh penemuan-penemuan yang telah dilakukan. Informatika merupakan salah satu bidang keilmuan dan keteknikan dimana memiliki perkembangan yang sangat pesat terlihat dari teknologi-teknologi yang ada di masyarakat. Keilmuan dalam informatika juga dimanfaatkan untuk mengembangkan penelitian dalam bidang keilmuan lain, salah satunya adalah bioinformatika.

Sequence Alignment merupakan salah satu topik dalam bioinformatika yang dapat diimplementasikan dengan algoritma dynamic programming. Rantai asam amino yang tersusun unik terkadang memiliki banyak kemiripan dengan struktur protein lain. Seperti pada makhluk hidup yang tersusun atas DNA yang unik, namun banyak species yang memiliki kemiripan tertentu. Hal tersebut memunculkan pertanyaan tentang keterhubungan antara struktur pada spesies yang berbeda. Oleh karena itu, dibutuhkan *tool* yang dapat mempermudah analisis dari banyak struktur yang berbeda.

Sumber:http://www.chemguide.co.uk/organicprops/aminoacids/dna1.html

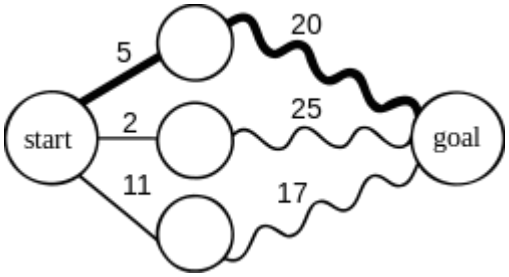


II. Dasar Teori

A. Program Dinamis

Program dinamis merupakan salah satu algoritma untuk memecahkan permasalahan kompleks dengan membagi permasalahan menjadi langkah-langkah dan menggabungkan solusi dari hasil tiap langkah yang dibuat. Salah satu sifat dari program dinamis adalah adanya keterhubungan antara sub-massalah hingga masalah yang lebih besar, dimana keterhubungan tersebutlah yang digunakan untuk membangun persamaan sebagai solusi, atau yang disebut *Bellman equation* [1]. Langkah-langkah pemecahan masalah dengan program dinamis secara umum yaitu:

1. Mendefinisikan sub-masalah
2. Menentukan rekursi dana atau fungsi berkaitan dengan sub-masalah tersebut,
3. Menyatukan solusi



Sumber: www.wikipedia.org

Jika dilihat, program dinamis memiliki pendekatan mirip dengan greedy, namun perbedaanya adalah pada program dinamis dapat dihasilkan lebih dari satu solusi. Hal tersebut dikarenakan untuk setiap langkah yang dibuat, dapat dimungkinkan adanya lebih dari satu keputusan.

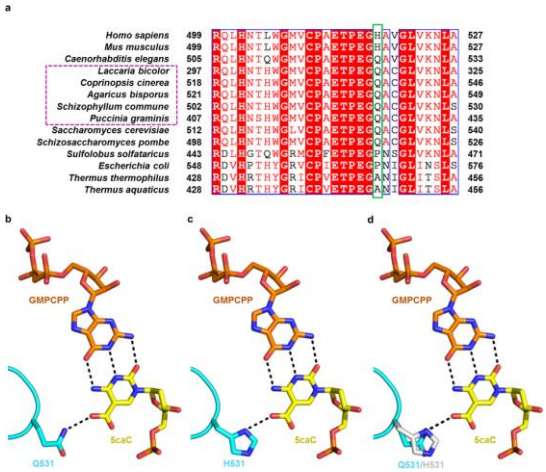
Langkah-langkah yang didefinisikan akan digunakan untuk membentuk rangkaian keputusan dengan menggunakan prinsip optimalitas. Prinsip Optimalitas yaitu “jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal” [2]. Oleh karena itu, kita dapat bekerja tahap demi tahap untuk

membentuk solusi optimal dengan menentukan hasil optimal dari setiap tahap.

Secara umum, program dinamis terdiri dari beberapa jenis yaitu: *1 dimensional DP, 2 dimensional DP, interval DP, tree DP,* dan *subset DP* [3]. Pada topik kali ini, program dinamis yang akan digunakan adalah *2 dimensional DP* yaitu dengan menggunakan struktur data array dua dimensi (matriks). Beberapa contoh permasalahan yang dapat diselesaikan dengan program dinamis adalah *Longest Common Subsequence (LCS), Shortest Path, Capital Budgeting, Knapsack,* dll.

B. Sequence Alignment

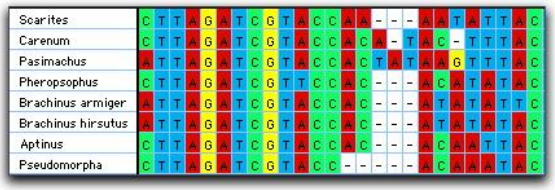
DNA, RNA, dan protein memiliki struktur yang terdiri dari asam amino dan memiliki keunikan tertentu. Sequence Alignment merupakan salah satu topik dalam bioinformatika untuk menyusul dua atau lebih rantai DNA, RNA, maupun protein. Sequential Alignment digunakan untuk mengidentifikasi kesamaan dari serangkaian struktur penyusun untuk mengetahui keterhubungan fungsional, structural ataupun revolusinya [4]. Jika terdapat dua sekuens yang memiliki kesamaan pada sebagian besar strukturnya dan terdapat sebagian kecil strukturnya yang berbeda dapat diiterpretasikan sebagai mutasi dan jeda/*gap* yang ada sebagai *indels* karena adanya *insertion* atau *deletion* pada strukturnya. Oleh karena itu, dengan adanya sequence alignment, dapat dilihat perbedaan dan persamaan dari sekuens yang digunakan untuk menganalisa revolusi rantai asam amino pada strukturnya. Sequence Alinment juga digunakan dalam informatika, yaitu pada *edit distance cost* dari *natural language processing.*



Sumber: www.nature.com

III. PROGRAM DINAMIS

Penyelesaian untuk menyelaraskan keterurutan DNA adalah dengan mencocokkan setiap struktur pada 2 atau lebih rantai DNA. Pada pencocokan struktur DNA, akan dilakukan pendekatan dengan menentukan perbedaan, persamaan, dan jeda dari setiap strukturnya. Hal tersebutlah yang membedakan Sequential Alignment dengan Longest Common Sequenze. Pada permasalahan LCS, hanya ada dua kasus yaitu ketika karakter sama atau tidak sama, namun pada Sequential Alignment terdapat 3 kasus yaitu karakter sama, karakter berbeda, dan jeda. Contoh Sequential Alignment:



Sumber: www.sequence-alignment.com

Berdasarkan sifat, bentuk solusi, maka dibentuk ide awal untuk menyelesaikannya, yaitu:

- Dilakukan pencocokan setiap karaker
- 3 keputusan untuk setiap pencocokan, yaitu:
  - Hasil pencocokan menunjukkan karakter yang sama
  - Hasil pencocokan menunjukkan karakter yang berbeda dan diputuskan untuk membiarkan kedua karakter berbeda.
  - Hasil pencocokan menunjukkan karakter yang berbeda sehingga diputuskan untuk memberikan jeda, sehingga salah satu dari kedua rangkaian dikosongkan.
- Dibutuhkan konstanta tertentu untuk dapat memberikan keputusan sesuai dengan ketiga kasus diatas, yaitu: konstanta jika kedua karakter sama, berbeda dan untuk yang salah satunya adalah jeda.
- Konstanta dibuat sesuai dengan prinsip optimalitas sesuai dengan pendekatannya.
- Konstanta diatas, digunakan untuk pendekatan dua buah rangkaian, untuk jumlah yang lebih banyak, tentu diperlukan parameter lain untuk

Penyelesaian untuk dua sekuens, misal X dengan panjang m dan Y dengan panjang n, akan didapatkan dengan membuat matriks berukuran $m \times n$. Sedangkan nilai untuk setiap elemen matriks diisi dengan fungsi tertentu, yaitu:

A. Fungsi dan Solusi

Misal fungsi f(i,j) adalah nilai dari pencocokan karakter ke i dari sekuens X dengan karakter ke j dari sekuens Y yang akan dimasukkan ke dalam *matriks[i,j].* Untuk menentukan nilai dari *f(i,j)*, diperlukan hasil pencocokan sebelumnya dan konstanta tertentu, yaitu:

1. Konstanta untuk $X[i]=Y[j]$
2. Konstanta untuk $X[i]\neq Y[j]$
3. Konstanta untuk jeda(-)

Sehingga nilai *f(i,j)* dapat ditentukan dengan fungsi:

$$f(i,j) = \begin{cases} Konstanta(X[i] = Y[j]), & X[i] = Y[j] \; dan \; i = j = 0 \\ Konstanta(X[i] \neq Y[j]), & X[i] \neq Y[j] \; dan \; i = j = 0 \\ f(i, j-1) + Konstanta\, Jeda, & i = 0, j \neq 0 \\ f(i-1, j) + Konstanta\, Jeda, & i \neq 0, j = 0 \end{cases}$$

$$f(i,j) = max \begin{Bmatrix} f(i-1, j-1) + Konstanta(X[i] = Y[j]), & X[i] = Y[j] \\ f(i-1, j-1) + Konstanta(X[i] \neq Y[j]), & X[i] \neq Y[j] \\ f(i-1, j) + Konstanta\, Jeda \\ f(i, j-1) + Konstanta\, Jeda \end{Bmatrix} \; untuk \;\; i, j \neq 0$$

Nilai f(i,j) tersebut akan digunakan untuk mengisi elemen dari Matriks indeks [i,j] dan panah yang digunakan untuk *backtracking.*

B. *Matriks*

Matriks berukuran sesuai dengan panjang sekuens digunakan untuk menyimpan nilai fungsi yang telah didefinisikan. Selain itu matriks tersebut juga digunakan untuk *backtrack* hasil yang *alignment* yang didapat. .

f(i-1 , j-1)	*f(i , j-1)*
f(i-1 , j)	*f(i , j)*

Misal terdapat dua sekuens yaitu X = “CTTAGATCG” dan Y = “ATTAC”, dengan nilai similarity = 1, non-similarity = 0, dengan gap = -1, maka akan didapat matriks sbb:

	C	T	T	A	G	A	T	C	G
A	↖ 0	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8
T	↑ -1	↖ 1	↖ 0	← -1	← -2	← -3	↖ -4	← -5	← -6
T	↑ -2	↖ 0	↖ 2	← 1	← 0	← -1	↖ -2	← -3	← -4
A	↑ -3	↑ -1	↑ 1	↖ 3	← 2	↖ 1	← 0	← -1	← -2
C	↑ -4	↑ -2	↑ 0	↑ 2	↖ 3	2	↖ 1	↖ 1	← 0

Misal solusinya dalam bentuk dua string S1 dan S2, dimulai dari i = 8, j = 4, maka dari tabel matriks diatas dapat dilakukan *backtracking* yaitu :

1. i = 8, j = 4| Panah = ←
   S1 = G | S2 = -
2. i = 7, j = 4| Panah = ↖
   S1 = CG | S2 = C-
3. i = 6, j = 3| Panah = ←
   S1 = TCG | S2 = -C-
4. i = 5, j = 3| Panah = ↖
   S1 = ATCG | S2 = A-C-
5. i = 4, j = 2| Panah = ←
   S1 = GTCG | S2 = -A-C-
6. i = 3, j = 2| Panah = ←
   S1 = AGATCG | S2 = --A-C-
7. i = 2, j = 2| Panah = ↖
   S1 = TAGATCG | S2 = T- -A-C-
8. i = 1, j = 1| Panah = ↖ |
   S1 = TTAGATCG | S2 = TT- -A-C-
9. i =0, j =0| Panah=↖
   S1 = CTTAGATCG | S2=ATT- -A-C-

Sehingga diadapat solusi :

C T T A G A T C G
A T T – – A – C –

C. *Algoritma*

Secara umum algoritmanya adalah dengan mengeset nilai matriks berukuran m x n dengan nilai fungsi f(i,j) lalu dapat dilakukan *backtracking* untuk mendapatkan solusi.

```
Procedure SetAlignment (
          input/output M:array[1..m]of
               array[1..n] of integer,
          Input/output Arrow: array[1..m]of
                    array[1..n] of integer,
          input/output X,Y: array of char,
          input m,n= integer,
          input Similar,NonSimilar,Gap = integer)
ALGORITMA:
j traversal [0..n]
  i traversal [0..m]
     SetValue(M,Arrow,i,j,Similar,NonSimilar,Gap)
```

Algoritma untuk menyelesaikan Sequential Alignment untuk 2 rantai DNA dibagi menjadi 2, yaitu

1. SetValue
   Menentukan nilai elemen matriks dan nilai matriks arrow(panah) yang dimanfaatkan untuk *backtrack.* Secara umum, fungsi ini berisi implementasi fungsi dari program dinamis yang telah didefinisikan. Matriks yang kedua adalah Arrow untuk menyimpan arah dari nilai maksimal yang didapatkan dari indeks sebelumnya.

```
Procedure SetValue (
          input/output M:array[1..m]of
               array[1..n] of integer,
          Input/output Arrow: array[1..m]of
                    array[1..n] of integer,
          input/output X,Y: array of char,
          input i,j = integer,
          input Similar,NonSimilar,Gap = integer)
ALGORITMA:
if((i=0)AND(j=0))then
  if(X[i]=Y[j]) then
    M[i][j] <- Similar
  else
    M[i][j] <- NonSimilar
  Arrow[i][j] <- 1
else if(i=0)then
  M[i][j] <- M[i][j-1]+Gap
  Arrow[i][j]  <- 2
else if(j=0)then
  M[i][j] <- M[i-1][j]+Gap
  Arrow[i][j]  <- 3
else
  if(X[i]=Y[j) then
    M[i][j] <- max(M[i-1][j-1]+Similar,
                   M[i-1][j]+Gap,
                   M[i][j-1]+Gap)
  else
    M[i][j] <- max(M[i-1][j-1]+NonSimilar,
                   M[i-1][j]+Gap,
                   M[i][j-1]+Gap)
  if(M[i][j]=M[i-1][j]+Gap) then
    Arrow[i][j] <- 2 {panah ke kiri}
  else if(M[i][j]=M[i][j-1]+Gap)
    Arrow[i][j] <- 3 {panah ke atas}
  else
    Arrow[i][j] <- 1 {panah diagonal}
```

2. BackTrack : menelusuri hasil dari indeks belakang hingga terbentuk solusi yaitu 2 string. Jika arah mengarah ke diagonal, ada dua kemungkinan yaitu kedua karakter sama atau berbeda. Sedangkan untuk arah panah ke atas atau ke kiri, maka salah satu dari solusinya diisi dengan jeda (-).

```
Procedure BackTrack(
          Input/output Arrow: array[1..m]of
                       array[1..n] of integer,
          input X,Y : array of char,
          input m,n : integer,
          input/output S1,S2: string)
KAMUS
i,j: integer
arrowval: integer
ALGORITMA:
i = m-1;
j = n-1;
arrow = 0;
while((i>=0)AND(j>=0)) do
  arrow = M.Elmt[i][j].arrow;
  if(arrow=1)then   {diagonal}
    S1 <- X[i] + S1
    S2 <- Y[j] + S2;
    i <- i-1
    j <- j-1
  else if(arrow=2) then {up}
    S1 <- "-" + S1
    S2 <- Y[j] + S2
    j <- j-1
  else
    S1 <- X[i] + S1;
    S2 <- "-" + S2;
    i <- i-1
```

D. Kompleksitas

Kompleksitas waktu dari algoritma ini, sudah cukup mangkus dibandingkan dengan *brute force*. Untuk dua masukan dengan panjang karakter m dan n,maka akan dilakukan m x n pencocokan ;

O(m x n).

IV. Implementasi dan Hasil

A. Struktur Data

Implementasi dari algoritma *dynamic programming* ini dibuat dalam bahasa java yang dibagi menjadi 3 kelas yaitu: Matriks.java, Cell.java, dan Alignment,java.

Input berupa dua buah *string* dan keluaran berupa dua buah *string*. Kelas Matriks berisi matriks berukuran m x n yang merupakan ukuran *string input* dengan element berupa objek Cell. Kelas Cell berisi *value* dan *arrow* yaitu nilai fungsi dan panah untuk *backtracking*.

B. Input dan Output

1. m = 9, n = 7

```
Sekuens 1 : CTTAGATCG
Sekuens 2 : ATTACCG
 0,1|-1,3|-2,3|-3,3|-4,3|-5,3|-6,3|-7,3|-8,3|
-1,2| 1,1| 0,1|-1,3|-2,3|-3,3|-4,1|-5,3|-6,3|
-2,2| 0,1| 2,1| 1,3| 0,3|-1,3|-2,1|-3,3|-4,3|
-3,2|-1,2| 1,2| 3,1| 2,3| 1,1| 0,3|-1,3|-2,3|
-4,2|-2,2| 0,2| 2,2| 3,1| 2,1| 1,1| 1,1| 0,3|
-5,2|-3,2|-1,2| 1,2| 2,1| 3,1| 2,1| 2,1| 1,1|
-6,2|-4,2|-2,2| 0,2| 2,1| 2,1| 3,1| 2,1| 3,1|
Sequential Alignment:
CTTAGATCG
ATT--ACCG
Execution time is 0.000 miliseconds
```

1. m = 40, n = 14

```
-23,2|-21,1|-19,1|-17,2|-15,2|-13,2|-11,1|-10,
-24,2|-22,2|-20,2|-18,1|-16,2|-14,1|-12,2|-11,
-25,2|-23,2|-21,2|-19,2|-17,1|-15,2|-13,2|-12,
Sequential Alignment:
CTTAGATCGGATCGATGGGAAACTTATTTATAGAGACGAT
ATTA-CTCGG-TCG-TTGGAAA-TTA--T-T----A-G--
Execution time is 1.000 miliseconds
```

3. m = 30, n = 26

```
-24,2|-22,2|-20,2|-18,1|-16,2|-14,1|-12,2|-
-25,2|-23,2|-21,2|-19,2|-17,1|-15,2|-13,2|-
Sequential Alignment:
CTTAGATCGGATCGATGGGAAACTTATTTA
ATTA-CTCGG-TCG-TTGGAAA-TTATTAG
Execution time is 0.000 miliseconds
```

4. m = 1600, n = 1040

```
Sekuens 1 : 1600
Sekuens 2 : 1040
Sequential Alignment:
CTTAGATCGGATCGATGGGAAACTTATTTATAGAGACGA
ATTA-CTCGG-TCG-TTGGAAA-TTA--T-T----A-GA
Execution time is 12 miliseconds
```

5. m = 3200, n = 2080

```
Sekuens 1 : 3200
Sekuens 2 : 2080
Sequential Alignment:

ATTA-CTCGG-TCG-TTGGAAA-TTA--T-T--
Execution time is 86 miliseconds
```

6. m = 4800, n = 3120

```
Sekuens 1 : 4800
Sekuens 2 : 3120
Sequential Alignment:

ATTA-CTCGG-TCG-TTGGAAA-TTA--T-T----A-G
Execution time is 2062 miliseconds
```

C. Analisis

Dari hasil percobaan, didapat waktu eksekusi program sebagai berikut :

No	Ukuran seq1	Ukuran seq2	waktu
1	9	7	0 ms
2	30	26	1 ms
3	40	14	1 ms
4	80	28	1 ms
5	800	280	3 ms
6	1600	1040	12 ms
7	3200	2080	86 ms
8	4800	3120	2062 ms

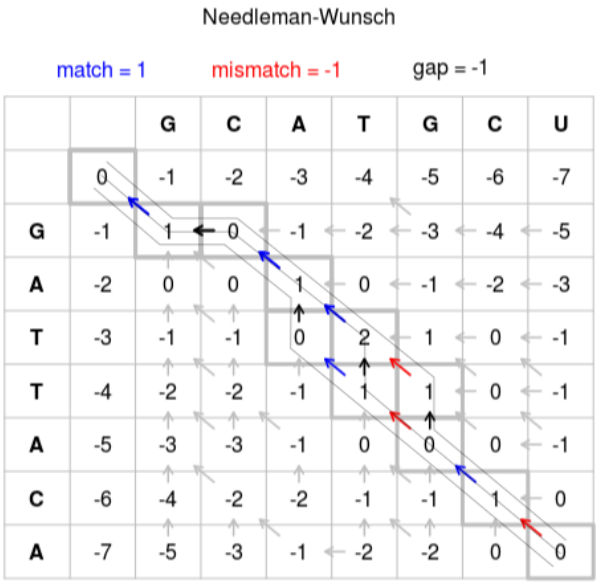
Terlihat bahwa waktu yang dihasilkan sangat signifikan ketika ukuran sekuens besar. Dari algoritma diatas seharusnya kompleksitas waktu O(mxn) dan dapat diselesaikan secara polinomial. Namun, pada implementasinya banyak faktor lain yang mempengaruhi sehingga tidak sesuai dengan perkiraan awal.

Selain itu,perlu adanya optimasi pada penggunaan struktur data, karena untuk menyelesaikan *multisequence alignment*, jika tetap menggunakan matriks, maka akan butuh alokasi *memory* yang besar.

D. Pengembangan

Algoritma yang diimplementasikan pada makalah ini terbatas untuk dua sekuens. Karena untuk melakukan *multisequence alignment* dibutuhkan *constraint* lain yang melibatkan seluruh sekuens, dan tentu kompleksitasnya lebih besar. Sehingga, pengembangan yang bisa dilakukan dari algoritma program dinamis diatas adalah dengan meminimalkan penggunaan struktur data yang besar dan penambahan fungsi pembatas sehingga pengecekan tidak harus dilakukan sebanyak m x n kali. Karena hanya dibutuhkuan satu hasil paling optimal, maka sebenarnya struktur data yang dibuat bisa diminimalkan dengan menggunakan array dinamis dan menyimpan nilai backtracking dalam struktur data tertentu. Selain itu perlu dikembangan lagi untuk lebih dari 2 sekuens (*multisequence alignment)*

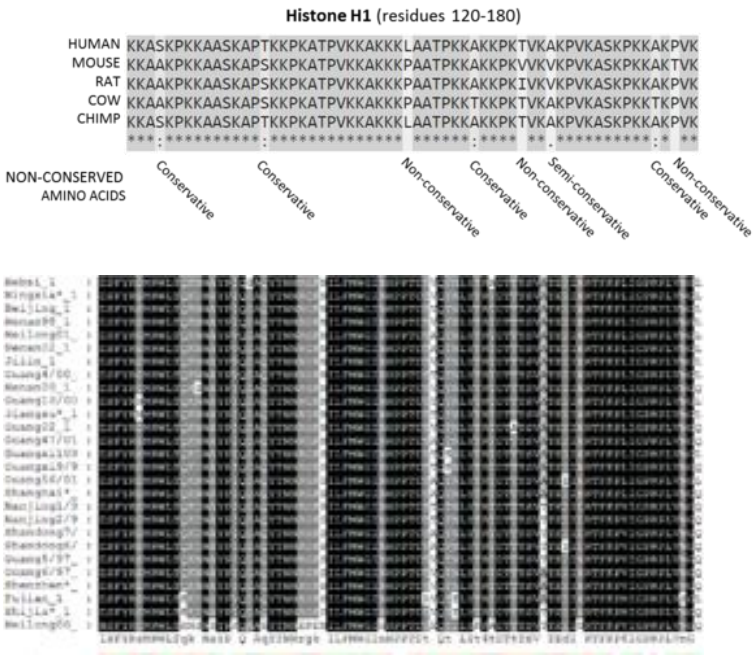
Beberapa algoritma lain yang dapat digunakan untuk menyelesaikan masalah ini adalah algoritma Needleman-Wunsch , dengan gambaran sbb:



Algoritma ini lebih fokus ke *natural langueange processing* yaitu *optimal matching.*

Selain itu, banyak pengembangan lain yang mengarah ke *natural language processing* seperti *edit distance cost* yang mungkin dapat dikembangkan untuk bidang non-biologi.

Tool yang biasa digunakan untuk melakukan *sequence alignment* adalah clustalO:



Sumber : Wikipedia.org

V. KESIMPULAN DAN SARAN

Kesimpulan dari makalah ini adalah algoritma diatas tidak cukup mangkus dalam *memory space*, dan cukup mangkus dalam waktu. Dibandingkan dengan brute force yaitu dengan membentuk seluruh kemungkinan *alignment*, maka algoritma ini jauh lebih cepat dan leboih efektif. Namun hanya terbatas pada dua sekuens saja. Sehingga, diperlukan pengembangann untuk *multisequence alignment.*

UCAPAN TERIMA KASIH

Pertama-tama penulis ingin mengucapkan syukur kepada Tuhan Yang Maha Esa karena rahmat dan hidayahnya, penulis dapat menyelesaikan makalah ini. Penulis juga ingin berterima kasih kepada orang tua dan teman seperjuangan yang selalu memberi dukungan dan semangat. Tak lupa penulis ucapkan terima kasih kepada Dr. Masayu Leylia Kodhra, S.T., M.T., Dr. Nur Ulfa Mulidevi S.T., M.Sc., dan Dr.Ir.Rinaldi Munir,M.T. atas bimbingan dan ilmu yang diberikan selama ini.

REFERENSI

[1] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. (2001), Introduction to Algorithms (2nd ed.), MIT Press & McGraw–HillJ. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] Slide Kuliah Dynamic Programming, 2015

[3] Park Jaehyun. (2015) Slide Dynamic Programming, CS 97SI, Stanford Sniversity

[4] Mount DM. (2004). *Bioinformatics: Sequence and Genome Analysis* (2nd ed.). Cold Spring Harbor Laboratory Press: Cold Spring Harbor, NY.

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,18 Mei 2017

Harum Lokawati
13515109