Aplikasi Pemrograman Dinamis Untuk Penyejajaran Sekuens DNA Dengan Algoritma Needleman-Wunsch

Rayhan Alghifari Fauzta 13519039
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
rayfazt@gmail.com

***Abstrak*—Penerapan ilmu keinformatikaan di bidang biologi telah menghasilkan berbagai inovasi dalam upaya ilmuwan untuk memahami struktur pembentuk makhluk hidup. Salah satu kemajuan yang dihasilkan adalah teknik penyejajaran antara dua sekuens DNA menggunakan komputer. Metode ini menerapkan prinsip pemrograman dinamis dalam bentuk algoritma Needleman-Wunsch. Algoritma Needleman-Wunsch beserta penggunaannya dalam sekuensing DNA akan dibahas dalam makalah ini.**

Kata kunci—pemrograman dinamis; DNA; sekuens; algoritma Needleman-Wunsch

I. PENDAHULUAN

Bioinformatika adalah ilmu yang mempelajari penerapan teknik komputasional untuk mengelola dan menganalisis informasi biologis. Bidang ini mencakup penerapan metode-metode matematika, statistika, dan informatika untuk memecahkan masalah-masalah biologis, terutama dengan menggunakan sekuens DNA dan asam amino serta informasi yang berkaitan dengannya. Contoh topik utama bidang ini meliputi basis data untuk mengelola informasi biologis, penyejajaran sekuens (sequence alignment), prediksi struktur untuk meramalkan bentuk struktur protein maupun struktur sekunder RNA, analisis filogenetik, dan analisis ekspresi gen.

Kemajuan teknik biologi molekular dalam mengungkap sekuens biologis dari protein (sejak awal 1950-an) dan asam nukleat (sejak 1960-an) mengawali perkembangan basis data dan teknik analisis sekuens biologis. Basis data sekuens protein mulai dikembangkan pada tahun 1960-an di Amerika Serikat, sementara basis data sekuens DNA dikembangkan pada akhir 1970-an di Amerika Serikat dan Jerman. Penemuan teknik sekuensing DNA yang lebih cepat pada pertengahan 1970-an menjadi landasan terjadinya ledakan jumlah sekuens DNA yang berhasil diungkapkan pada 1980-an dan 1990-an, menjadi salah satu pembuka jalan bagi proyek-proyek pengungkapan genom, meningkatkan kebutuhan akan pengelolaan dan analisis sekuens, dan pada akhirnya menyebabkan lahirnya bioinformatika.

Salah satu bidang riset utama bioinformatika adalah penyejajaran sekuens. Penyejajaran sekuens adalah metode untuk menata urutan rantai DNA, RNA, atau protein sedemikian rupa sehingga daerah dengan tingkat kecocokan yang tinggi antara dua sampel sekuens dapat diidentifikasi. Kecocokan ini mengindikasikan adanya hubungan fungsional, struktural, atau evolusioner dari kedua sekuens.

ID	Sequence
10	ATCGGTGACTATCG
20	CATCGTTCATCG
30	CATCGTGAAGT
40	TCGTGATTG
50	GCGTGATTC

Gambar 1. Contoh penyimpanan sekuens DNA dalam basis data biologis. (Sumber: https://www.researchgate.net/figure/Example-of-a-DNA-sequence-database_tbl1_232722321)

Saat ini, sebagian besar algoritma penyejajaran sekuens didasarkan pada prinsip pemrograman dinamis. Algoritma yang umum digunakan antara lain algoritma Needleman-Wunsch, algoritma Smith-Waterman, algoritma BLAST, algoritma FASTA, dan algoritma Ukkonen. Dari kelima metode tersebut, algoritma Smith-Waterman banyak digunakan untuk dua sekuens yang cenderung tidak mirip secara keseluruhan namun memiliki kecocokan di beberapa area yang kecil (*local similarity*). Di sisi lain, algoritma Needleman-Wunsch menganalisis keseluruhan sekuens dengan derajat kecocokan yang lebih tinggi sehingga mampu menghasilkan solusi yang lebih optimal meskipun dengan kompleksitas waktu dan ruang yang lebih tinggi.

II. DASAR TEORI

A. *Program Dinamis*

Program Dinamis (*dynamic programming*) adalah suatu metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Terdapat tiga karakteristik penyelesaian persoalan dengan program dinamis yaitu:

1. Terdapat sejumlah berhingga pilihan yang mungkin,

2. Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. Menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Perbedaaan algoritma *greedy* dan program dinamis terletak pada jumlah rangkaian keputusan yang dihasilkan, greedy hanya memiliki satu, sedangkan program dinamis memiliki lebih dari satu keputusan yang dipertimbangkan.

Program dinamis memiliki sebuah prinsip yang disebut prinsip optimalitas. Prinsip optimalitas mengindikasikan jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal. Hal ini berarti jika kita bekerja dari tahap k ke tahap $k + 1$, kita dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal. Ongkos pada tahap $k + 1$ didapat dari penjumlahan ongkos yang dihasilkan pada tahap k dengan ongkos dari tahap k ke tahap $k + 1$.

Untuk dapat dikatakan sebagai persoalan program dinamis, ada beberapa karakteristik yang harus dipenuhi oleh persoalan tersebut. Karakteristik tersebut adalah:

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*) yang pada setiap tahap hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada tahap tersebut.
3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk setiap status pada tahap $k + 1$.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

Program dinamis memiliki dua pendekatan yaitu maju (*forward* atau *top-down*) dan mundur (*backward* atau *bottom-up*). Misalkan x1, x2, ..., xn menyatakan peubah (*variable*) keputusan yang harus dibuat masing-masing untuk tahap 1, 2, ..., n. Maka:

1. Program dinamis maju. Program dinamis bergerak mulai dari tahap 1, terus maju ke tahap 2, 3, dan seterusnya sampai tahap n. Runtunan peubah keputusan adalah x1, x2, ..., xn.
2. Program dinamis mundur. Program dinamis bergerak mulai dari tahap n, terus mundur ke tahap n – 1, n – 2, dan seterusnya sampai tahap 1. Runtunan peubah keputusan adalah xn, xn-1, ..., x1.

B. Penyejajaran Sekuens

Penyejajaran sekuens (*sequence alignment*) adalah proses penyusunan atau pengaturan dua atau lebih sekuens sehingga persamaan sekuens-sekuens tersebut tampak nyata. Hasil dari proses tersebut juga disebut sebagai *sequence alignment* atau *alignment*. Baris sekuens dalam suatu alignment diberi sisipan (umumnya dengan tanda "–") sedemikian rupa sehingga kolom-kolomnya memuat karakter yang identik atau sama di antara sekuens-sekuens tersebut.

Contoh *alignment* DNA dari dua sekuens pendek DNA yang berbeda, "ccatcaac" dan "caatgggcaac" (tanda "|" menunjukkan kecocokan atau match di antara kedua sekuens) adalah sebagai berikut:

```
ccat---caac
|  ||   ||||
caatgggcaac
```

Penyejajaran sekuens merupakan metode dasar dalam analisis sekuens. Metode ini digunakan untuk mempelajari evolusi sekuens-sekuens dari leluhur yang sama (*common ancestor*). Ketidakcocokan (*mismatch*) dalam *alignment* diasosiasikan dengan proses mutasi, sedangkan kesenjangan (*gap*, tanda "–") diasosiasikan dengan proses insersi atau delesi. Penyejajaran sekuens memberikan hipotesis atas proses evolusi yang terjadi dalam sekuens-sekuens tersebut. Misalnya, kedua sekuens dalam contoh alignment di atas bisa jadi berevolusi dari sekuens yang sama "ccatgggcaac". Dalam kaitannya dengan hal ini, *alignment* juga dapat menunjukkan posisi-posisi yang dipertahankan (*conserved*) selama evolusi dalam sekuens-sekuens protein, yang menunjukkan bahwa posisi-posisi tersebut bisa jadi penting bagi struktur atau fungsi protein tersebut.

Penyejajaran sekuens juga digunakan untuk mencari sekuens yang mirip atau sama dalam basis data sekuens. BLAST adalah salah satu metode alignment yang sering digunakan dalam penelusuran basis data sekuens. BLAST menggunakan algoritma heuristik dalam penyusunan *alignment*. Beberapa metode lain yang merupakan pendahulu BLAST adalah metode Needleman-Wunsch dan Smith-Waterman. Metode Needleman-Wunsch digunakan untuk menyusun *alignment* global di antara dua atau lebih sekuens, yaitu *alignment* atas keseluruhan panjang sekuens tersebut. Metode Smith-Waterman menghasilkan *alignment* lokal, yaitu *alignment* atas bagian-bagian dalam sekuens. Kedua metode tersebut menerapkan pemrograman dinamis dan hanya efektif untuk alignment dua sekuens (*pairwise alignment*).

C. Algoritma Needleman-Wunsch

Algoritma Needleman-Wunsch merupakan implementasi program dinamis (*dynamic programming*), yaitu suatu algoritma untuk memecahkan masalah yang bekerja dengan menguraikan solusi menjadi sekumpulan tahap (*stage*) dimana solusi dari

persoalan tersebut dapat dipandang atau dipertimbangkan. Algoritma Needleman-Wunsch digunakan untuk menentukan tingkat kesamaan atau kecocokan dua buah teks. Algoritma ini juga digunakan untuk menemukan alignment yang memiliki nilai optimal pada global alignment di dua buah sekuen. Algoritma ini diciptakan oleh Saul Needleman dan Christian Wunsch pada tahun 1970.

Dalam bioinformatika, tingkat perbedaan (*mismatch*) biasanya ditentukan oleh matriks nilai (*score matrix*). Misalnya dalam algoritma program dinamis

$A' = (a'1, a'2, ..., a'n')$

$B' = (b'1, b'2, ..., b'n')$

merupakan *alignment* optimal untuk sekuens berpasangan

$A = (a1, a2, a3, ..., ana)$

$B = (b1, b2, b3, ..., bnb)$.

Langkah pengerjaan algoritma Needleman-Wunsch adalah sebagai berikut:

1. Inisialisasi matriks.

   Misal terdapat dua sekuens $A = a1, a2, ..., an$ dan $B = b1, b2, ..., bm$, maka dapat dibuat matriks nilai berukuran (n+1) x (m+1) dengan n adalah banyak baris yang menyatakan panjang sekuens pertama, dan m adalah banyak kolom yang menyatakan panjang sekuens kedua. Kemudian isi baris pertama dan kolom pertama matriks nilai dengan nilai *gap penalty*. *Gap penalty* merupakan nilai yang diperoleh ketika dilakukan perbandingan antara residu dalam sebuah sekuens dengan karakter kosong (*gap*) dalam sekuens lainnya.

2. Pengisian matriks.

   Misalkan matriks nilai disebut matriks S, maka rumus untuk elemen dari matriks S adalah

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(a_i, b_j) \\ S(i-1,j) - d \\ S(i,j-1) - d \end{cases} \quad (1)$$

   dengan:

   S(i -1, j – 1) = elemen matriks S di diagonal kiri atas

   S(i, j – 1) = elemen matriks S di kiri S(i, j)

   S(i – 1, j) = elemen matriks S di atas S(i, j)

   $s(a_i, b_j)$ = elemen matriks substitusi di residu i pada sekuens a dan residu j pada sekuens b

   d = *gap penalty*.

   Model *gap* diasumsikan linear, sehingga:

$$s(a_i, b_j) = \begin{cases} skor\ match, & a_i = b_j \\ skor\ mismatch, & a_i \neq b_j \end{cases} \quad (2)$$

3. Traceback

   Setelah matriks nilai telah terisi sepenuhnya, maka skor *alignment* maksimum dari dua buah sekuens adalah nilai dari elemen matriks nilai yang berada di paling kanan bawah yaitu S(n + 1, m + 1) = s(n, m) = s(i, j). Skor *alignment* adalah jumlah dari semua nilai substitusi ditambah dengan jumlah dari semua *gap penalty*.

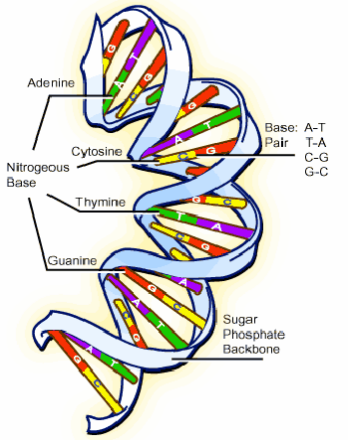
4. Menentukan hasil *alignment*

   Penentuan hasil *alignment* dimulai dengan menotasikan pasangan DNA sebagai a_i, b_j jika alur mundurnya dimulai dari a_i, b_j ke sudut kiri atas. Langkah berikutnya adalah menyisipkan simbol virtual pada sekuens vertikal dengan notasi (a_i, -) jika alur mundurnya horizontal. Jika alur mundurnya vertikal, simbol virtual disisipkan pada sekuens horizontal dengan notasi (-, b_j).

D. DNA

Deoxyribonucleic acid (DNA) adalah suatu materi yang terdapat pada tubuh manusia dan semua makhluk hidup yang diwarisi secara turun temurun. Semua sel pada tubuh memiliki DNA yang sama dan sebagian besar terdapat pada nukleus. DNA juga dapat ditemukan pada mitokondria. DNA merupakan biomolekul yang berupa asam nukleat yang berfungsi untuk menyimpan informasi genetik pada suatu organisme. DNA merupakan polimer yang terdiri dari tiga komponen utama yaitu gugus fosfat, gula deoksiribosa dan basa nitrogen. Basa-basa nitrogen tersebut adalah adenin (A), sitosin (C), guanin (G), dan timin (T). Adenin berikatan dengan timin dan sitosin berikatan dengan guanin.

DNA berbentuk untai ganda (*double helix*) yang disatukan oleh ikatan hidrogen antara basa-basa di dalam kedua untai tersebut. Pada struktur untai ganda, orientasi rantai nukleotida pada satu untai berlawanan dengan orientasi untai nukleotida lainnya. Rangkaian DNA ini memiliki kode-kode yang merepresentasikan ciri fisik makhluk hidup, cara bekerja sel dan lain-lain. Kerja sel dapat dilihat dari protein yang dihasilkan dari kombinasi DNA ini. DNA akan mengalami transkripsi menjadi RNA (*Ribo-nucleic acid*), yaitu rangkaian nukleotida seperti halnya DNA, namun nukleotida timin diganti dengan urasil (U).



Gambar 2. Struktur *double helix* pada DNA (Sumber: https://www.researchgate.net/figure/DNA-double-Helix-22_fig4_228359468)

Fungsi DNA adalah untuk bereplikasi dan mensintesis protein. Replikasi diperlukan untuk memberikan informasi yang sama pada tiap sel baru ketika terjadi pembelahan. Dalam proses sintesis protein, DNA menyediakan informasi genetik yang diperlukan oleh sel untuk dapat berfungsi secara fungsional dan struktural. Terkadang, bisa terjadi perubahan yang tidak direncanakan pada suatu bagian DNA yang disebut mutasi.

III. Implementasi Dan Pengujian

Langkah-langkah penyejajaran sekuens dengan algoritma Needleman-Wunsch seperti yang telah ditulis di dasar teori adalah inisialisasi matriks, pengisian matriks, dan *traceback*. Implementasi kode program untuk masing-masing langkahnya adalah sebagai berikut:

1. Inisialisasi matriks

Misalkan sekuen 1 berukuran m dan sekuen 2 berukuran n. Maka dibentuk matriks nilai berukuran (m+1) x (n+1).

```
def mat_zeroes(rows, cols):
    mat = []
    for x in range(rows):
        mat.append([])
        for y in range(cols):
            mat[-1].append(0)
    return mat
```

Potongan kode tersebut akan membuat matriks dengan seluruh elemennya bernilai nol dengan ukuran *rows* baris dan *cols* kolom.

2. Pengisian matriks

Pengisian matriks nilai S menggunakan rumus pada persamaan (2) dengan skor match, mismatch, dan gap yang telah ditentukan masing-masing sebesar 1, -1, dan -1. Sebelum melakukan pengisian matriks, dicari terlebih dahulu matriks substitusi dengan persamaan:

$s(0, j) = -jxd$,

$s(i, 0) = -ixd$, dan

$s(0,0) = 0$

Implementasi program pengisian matriks substitusi adalah sebagai berikut:

```
def match_score(a, b):
    if a == b:
        return 1   #skor match
    elif a == '-' or b == '-':
        return -1   #gap penalty
    else:
        return -1   # skor mismatch
```

Selanjutnya akan dihitung matriks nilai menggunakan persamaan (1) dengan implementasi sebagai berikut:

```
def score_matrix(seq1, seq2):
    # panjang sekuens
    n = len(seq1)
    m = len(seq2)

    # inisialisasi matriks skor
    score = mat_zeroes(m + 1, n + 1)

    # Hitung skor
    # Isi kolom pertama
    for i in range(0, m + 1):
        score[i][0] = -1 * i

    # Isi baris pertama
    for j in range(0, n + 1):
        score[0][j] = -1 * j

    # Isi seluruh value sisa
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            # Hitung skor
            match=score[i - 1][j - 1] +
match_score(seq1[j-1],seq2[i-1])
            delete = score[i - 1][j] - 1
            insert = score[i][j - 1] - 1
            # Cari skor maksimum
            score[i][j] = max(match,delete,insert)
```

3. Traceback

Setelah matriks nilai terisi, berikutnya dilakukan langkah *traceback* dari titik awal S(*len(seq1)*, *len(seq2)*) hingga titik akhir S(0,0). Jika $S(i,j) = S(i-1,j-1) + s(a_i, b_j)$ maka lintasannya $:(i,j) \rightarrow (i-1,j-1)$. Implementasi *traceback* adalah sebagai berikut:

```
def traceback(seq1, seq2, score):
    # variabel untuk menyimpan alignment
    align1 = ""
    align2 = ""

    # mulai dari sel kanan bawah matriks
    i = len(seq1)
    j = len(seq2)
```

```
    while i > 0 and j > 0:
        score_current = score[i][j]
        score_diagonal = score[i-1][j-1]
        score_up = score[i][j-1]
        score_left = score[i-1][j]

        # Cek sel yang sedang dihitung
lalu update nilai i dan j
        if score_current == score_diagonal + match
_score(seq1[j-1], seq2[i-1]):
            align1 += seq1[j-1]
            align2 += seq2[i-1]
            i -= 1
            j -= 1
        elif score_current == score_up - 1:
            align1 += seq1[j-1]
            align2 += '-'
            j -= 1
        elif score_current == score_left - 1:
            align1 += '-'
            align2 += seq2[i-1]
            i -= 1

    # Selesai tracing hingga sel kiri atas
    while j > 0:
        align1 += seq1[j-1]
        align2 += '-'
        j -= 1
    while i > 0:
        align1 += '-'
        align2 += seq2[i-1]
        i -= 1

    # normalisasi urutan sekuens
    align1 = align1[::-1]
    align2 = align2[::-1]

    return(align1, align2)
```

Satu fungsi tambahan yang diimplementasikan adalah fungsi untuk menghitung homologi. Homologi dalam ilmu biologi adalah kesamaan pada beberapa spesies yang dihasilkan dari leluhur yang sama. Dalam kasus penyejajaran sekuens DNA, homologi dihitung dengan rumus:

$$Homologi = (banyaknya\ nukleotida\ kedua\ sekuens\ yang\ sama \div panjang\ sekuens\ setelah\ disejajarkan) \times 100$$

Implementasi kode untuk menghitung homologi adalah sebagai berikut:

```
def homology(align1, align2):
    result = 0
    for i in range(0, len(align1)):
        if align1[i] == align2[i]:
            result += 1
    result = ((result) / len(align1)) * 100
    return result
```

Pengujian dilakukan dengan dua data sekuens DNA *Avian coronavirus* yang menyebabkan bronkitis pada burung. Data didapat dari situs National Center for Biotechnology Information (NCBI), National Library of Medicine (NLM), Amerika Serikat. Data uji ditunjukkan sebagai berikut:

Sekuens 1: *Avian infectious bronchitis virus (strain D1466) peplomeric protein gene encoding the S1 and S2 subunits*.

```
ATGTGGGCATCGTTACTGTCAGTAGTGACTCTTTTGTTTGCTTT
AAGTGAATGTAGTATAGTAGGTGAAAATTACACATACTATTACC
AGAGTCAGTTTAGGCCGCCTAATGGCTGGCATAAACATGGTGGA
GCCTATCTTGTAACCAATGAAACTGACATATCCTATAATGGTGT
GTCTTGTACTGTGGGTACAATAAAAGGCGGCATTGTCATTAATG
AGAGTGCTATATCTTTTGTTACAAAAACACCCATTGCTTGGTCA
GCCAACGGCGTTTGCACTACATATTGTAATTACTCCAGCTTATA
TGTGTTTGTTACCCATTGTGGGGGCAGCGGACATACTAGTTGTA
TTATAAATACAAATCGCATAGGCGAGATTGTTTTAGGTGTTAAA
GACTTTTCTGGTAACTGGATTTATAATCGTACTATAAAGGCTAT
TGGTCCGTATAGTAAATTTACAGCCTGGCAATGTCTTGCTAATT
TTACCAGTGTGTTTCTAAACGGCAACCTTGTGTATAGTTCTAAC
TTTACGGAGGATGTTGCAGCGGGTGGTGTTTATGCTAAAAGCGT
CAATGGTCTAAAACGTAGAATTATGAAGGACACTGATGTTTTGG
CATATTTTGTAAATGGCACTGCTGTTGAAGTGATTGTTTGTGAT
GACAGCCCTAGAGGTAGGTTAGCATGTCAGTATAATACAGGAAA
TTTTACTGATGGGTTATACCCTTTCGTAAGTTACAATGTAGTTA
ATAATAGTGTTGTTGTTTATGAGGTTATTAGTACTACAACTTAT
GGTAAACTTAACAACATTACTTTTCATAATGAAACTGGTGCACC
ACCTGCAGGTTCTAATGTTGCTAATTTTATTAAATATCAGACGC
ATGTGGTGCCTGAAGGTTTTGTTAGGCTCAATTTTTCTTTCTTG
TCTACTTACAGGTATCAGGAGTCTGATTTTACTTATGGTTCTTA
TCATAAGGCTTGTAATTTTAGACTAGAAAGTATTAATAATGGTT
TAATGTTTAATACTTTAAGTGTTTCTATTAGCTATGGACCACTT
AAGGGTTCTTGTAAGCAGTCAGTATTTAATCGTAAAGCAACATG
CTGTTATGCCTATAAATATCCCACTAATGGGGTTCAAGAGTGTA
AGGGTGTTTATAATGGAGAACGCAATACTAAATTTGAATGCGGG
CTTCTTGTATTTATAGACAAGACTGATGGTTCACGCATAATAAC
TGCAGAAAAACCACCTGTTTATACTACTAATTTTACTAATAATA
TTGTTGTTGGTAAGTGTGTTAATTATAATATTTATGGCAGGTAT
GGCCAAGGCGTCATTAGTAATATAACTACTGAAGCATTTGGATT
TTTACAGGGAGATGGTTTGGTCATCTTGGACACTGCTGGTTCTA
TAGATATTTTTTCTGTTAAGGATGGGCCACTCACACATTATTAC
AAAATTAATCCTTGTAATGATGTAAATCAACAATATGTAGTGTC
AGGAGGAAATATAGTTGGTCTTCTCACATCTAGTAATGAGACTG
GCTCTATTCAGTTAGAAGATCAGTTTTATATTAAACTCACTAAT
AGCACTCGTAGGCATAGGAGA
```

Sekuens 2: *Avian infectious bronchitis virus (strain V1397) peplomeric protein gene encoding the S1 and S2 subunits.*

```
ATGTTGGCACAGTTACTGTTAGCAGTGACTCTTTTGTCTGCTTT
AGGTGAATGTAGTATAGTAGGTGAAAATTACACATACTATTACC
```

```
AGAGTCAGTTTAGACCGCCTAATGGCTGGCATAAACATGGTGGA
GCCTATCTTGTAGTTAATGAAACTGATATATCCTATGATGCTGC
GTCTTGTACTGTGGGTACAATAAAAGGCGGCATTGTCATTAATG
AGAGTGCTATATCTTTTGTTACTAAAACACCTATTGCTTGGTCA
GCTCAAGGCGTTTGCACTACATATTGTAATTATTCCAGCCTATA
TGTGTTTGTAACCCATTGTGGGGGCCGTGGACATAATAGTTGTA
TTATAAATACAAATCGCATAGGCGAGATTGTTTTAGGTGTTAAA
TCCTTTTCTGGTAACTGGATTTATAATCGCACTATACAGGCTAC
TGGTCCGTATAGTAAATTTACAGCCTGGCAATGTCTTGCTAATT
TTACCAGTGTGTTTCTAAATGGCAACCTTGTGTATAGTTCTAAC
TTTACGGAGGATGTTGCAGCGGCTGGTGTTTATGCTAAAACAGT
CAATGGTCTAAAACGTAGAATTATGAAGGACACTGATGTTTTGG
CATATTTTGTAAATGGCACTGCTGTTGAAGTGATTGTTTGTGAT
GACAACCCTAAAGGTAGGTTAGCATGTCAGTATAATACAGGAAA
TTTTACTGATGGGTTATACCCTTTCGTAAGTAATAATGTAGTTA
ATGATAGTGTTGTTGTTTATGATGTTATTAGTACTACAACTTAT
GGTAAACTTAACAACATTACTTTCCATAATGAAACTAGTGCACC
ACCTGCAGGTTCTAATGTTGCTAATTTTATTAAATATCAGACGC
ATGTGTTGCCTGAAGGTTTTGTTAGGCTTAATTTTTCTTTCTTG
TCTACTTACAGGTATCAGGAGTCTGATTTTACTTATGGTTCTTA
TCATAAGGCTTGTAATTTTAGACTAGAAAGTATTAATAATGGTT
TAATGTTTAATACTTTAAGTGTTTCTATTAGCTATGGACCACTT
AAGGGTTCTTGTAAGCAGTCAGTGTTTAATCATAAAGCAACGTG
CTGTTATGCCTATAAATATCCCACTAATGGGGTTCAAGAGTGTA
AGGGTGTTTATAATGGAGAACGCAATACTAAATTTGAATGCGGG
CTTCTTGTATTTATAGACAAGACTGATGGTTCACGCATAATAAC
TGCAGAAAAACCACCTGTTTATACTACTAATTTTACTAATAATA
TTGTTGTTGGTAAGTGTGTTAATTATAATATTTATGGTAGGTAT
GGCCAAGGCGTCATTAGTAATGTAACTACTGAAGCATTTGGTTT
TTTAGAGGGAGATGGTTTGGTCATCTTGGACACTGCTGGTTCTA
TAGATATTTTTGTTGTTAGGGATGGTCCATTCACACATTATTAC
AAGATTAATCCTTGTAATGATGTAAATCAACAATATGTAGTGTC
AGGAGGAAATATAGTTGGTCTTCTCACATCTAGTAATGAGACTG
GCTCTATTCAGTTAGAAGATCAGTTTTATATTAAACTCACTAAT
AGCACCCGTAGGCATCGGAGA
```

Pengujian dilakukan dengan nilai *match* = 1, nilai *mismatch* = -1, dan nilai *gap penalty* = -1. Nilai ini dapat divariasikan untuk kemungkinan mendapat hasil yang berbeda. Hasil pengujian yang didapat adalah sebagai berikut:

```
Align 1: …ATC-GTT…AGC-CAA…AGC-GTC…TTT-TCT…

Align 2: …GCA-CAG…CAA-GGC…AAA-CAG…TGT-TGT…

Homologi: 96.33312616532007
```

Dari pengujian diatas, terlihat bahwa terdapat beberapa mutasi yang membedakan kedua sekuens. Hal ini sudah terlihat dari nilai homologi yang tidak mencapai 100%, dengan jelas menandakan adanya perbedaan urutan nukleotida. Hasil pengujian memberitahukan ada lima tempat di sekuens nukleotida yang mengalami "pergeseran". Meskipun begitu, salah satu kekurangan implementasi saat ini adalah tidak adanya kemampuan untuk mencari mutasi persis ada di nukleotida mana. Pengetahuan yang didapat dari pengujian hanya mampu memberitahu dua sekuens DNA sama persis atau tidak. Perlu adanya modifikasi lebih lanjut pada program untuk mengetahui secara persis indeks nukleotida mana saja yang mengalami perubahan/mutasi.

IV. KESIMPULAN

Algoritma Needleman-Wunsch sebagai contoh penerapan pemrograman dinamis adalah salah satu metode populer dalam melakukan penyejajaran sekuens DNA. Algoritma ini memiliki keunggulan salah satunya karena mampu menganalisis keseluruhan sekuens dengan derajat kecocokan yang lebih tinggi dibanding algorima lain seperti Smith-Waterman. Hal ini membuat solusi yang dihasilkan lebih optimal dengan *trade-off* kompleksitas waktu dan ruang yang diperlukan cenderung lebih tinggi. Hal ini membuktikan salah satu konsep strategi algoritma yaitu pemrograman dinamis memiliki tempat yang penting di bidang biologi komputasional, dengan kegunaannya mencakup penyocokan sekuens DNA, pelipatan protein, pengikatan protein-DNA, dan permasalahan bioinformatika maupun bidang lain yang berhubungan dengan optimisasi.

UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan terima kasih dan rasa syukur yang sebesar-besarnya kepada Tuhan Yang Maha Esa karena atas berkat rahmat dan hidayahnya penulis dapat menyelesaikan tugas makalah ini dengan baik. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir, Bapak Rila Mandala, Ibu Nur Ulfa Maulidevi, dan Bapak Dwi Hendratmo Widiyantoro selaku dosen pengajar mata kuliah IF2211 Strategi Algoritma yang telah memberikan banyak ilmu yang bermanfaat sebagai bekal penulisan makalah ini. Penulis juga mengucapkan terima kasih sebesar besarnya untuk keluarga dan sanak saudara yang telah memberi dukungan penuh dalam perkuliahan dan kehidupan serta teman-teman program studi Teknik Informatika yang bersama-sama menjalani perkuliahan pada Semester II Tahun 2020/2021 di program studi Teknik Informatika tercinta ini.

REFERENSI

[1] Jiang, X., Fu, X., Dong, G., & Li, H. (2017). *Research on Pairwise Sequence Alignment Needleman-Wunsch Algorithm*. *141*(Icmmcce), 1041–1046. https://doi.org/10.2991/icmmcce-17.2017.187.

[2] Muhamad, F. N., Ahmad, R. B., Asi, S. M., & Murad, M. N. (2018). Performance Analysis of Needleman-Wunsch Algorithm (Global) and Smith-Waterman Algorithm (Local) in Reducing Search Space and Time for Dna Sequence Alignment. *Journal of Physics: Conference Series*, *1019*(1). https://doi.org/10.1088/1742-6596/1019/1/012085.

[3] Boes, O., & Lenaerts, T. (2014). Improving the Needleman-Wunsch algorithm with the Dynamine predictor, Informatique.

[4] Pop, M., & Salzberg, S. L. (2008). Bioinformatics challenges of new sequencing technology. Trends in genetics, 24(3), 142-149.

[5] Munir, Rinaldi. (2009). Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Program Studi Teknik Informatika Institut Teknologi Bandung.

[6] https://www.ncbi.nlm.nih.gov/, diakses 9 Mei 2021.

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, 11 Mei 2021

Rayhan Alghifari Fauzta 13519039