Perbandingan Needleman-Wunsch dan Lempel-Ziv dalam Teknik Global Sequence Alignment: Keunggulan Faktorisasi Sempurna

Mikhael Avner Malendes [#1], Hendra Bunyamin[*2]

Jurusan S1 Teknik Informatika, Fakultas Teknologi Informasi

Universitas Kristen Maranatha, Jalan Prof. Drg. Surya Sumantri No.65, Bandung 40164

[1]haemometer@gmail.com

[2]hendra.bunyamin@it.maranatha.edu

Abstract — **Essentially research on bioinformatics discusses the changing of the DNA information, and marking the mutation for the DNA. Comparing DNA and finding out how two DNA can have similarities, bioinformatics utilizeds algorithms that work on global alignment and local alignment. The global alignment compares all the characters in a sequence; on the other hand, the local one only takes a piece of characters from the alignment. This study proposes two algorithms for processing the DNA sequence in global alignment that are Needleman-Wunsch and Lempel-Ziv algorithms. These algorithms work with building a scoring matrix and creating an alignment based on the matrix. We test DNA sequences randomly with the length less than 1000 characters and more than 1000 characters. Needleman-Wunsch leading with processing speed up to 1 miliseconds for less than 1000 character dataset and 42 miliseconds for more than 1000 characters dataset. However, Lempel-Ziv is leading the processing speed on specific case of perfect phrase in DNA sequence.**

Keywords— **Bioinformatics, Needleman-Wunsch, Lempel-Ziv, sequence alignment algorithms, perfect phrase**

I. PENDAHULUAN

A. Latar Belakang

Kehidupan manusia merupakan rangkaian dari berbagai macam penerimaan, pengolahan dan penyampaian informasi, baik itu cara bicara, pola hidup, sejarah, dan semua yang berhubungan dengan kegiatan manusia. Penyampaian informasi tidak hanya disampaikan di tahap komunikasi manusia pada umumnya, misalnya berbicara atau menyampaikan sesuatu secara visual, penyampaian informasi juga terjadi di tahap molekular sel, yaitu DNA. DNA adalah struktur informasi unik terkecil yang dimiliki oleh setiap organisme dan DNA berperan penting dalam seluruh kehidupan yang terjadi di muka bumi ini.

Ilmu yang mempelajari tentang DNA secara spesifik, salah satunya, adalah mikrobiologi. Mikrobiologi sudah berkembang pesat dari abad ke abad dengan meneliti bagaimana suatu informasi dari DNA tiap organisme dapat diwariskan atau bahkan dimodifikasi. Seiring dengan perkembangan ilmu mikrobiologi, kehidupan di dunia juga mengalami perkembangan dan perubahan yang kita sebut dengan evolusi. Evolusi yang kita kenal berhubungan erat dengan perubahan penyampaian informasi DNA oleh tiap organisme. Informasi DNA atau sequence dari DNA yang ada mengalami perubahan yang disebut mutasi.

Menurut Deepa Agashe dalam penelitiannya di jurnal biologi molekular dan evolusi, mutasi atau perubahan informasi sel terjadi berulang kali pada banyak varian gen [3]. Dengan semakin banyaknya perubahan informasi sel dari tiap organisme, wujud atau cara bertahan hidup sebuah organisme semakin beragam. Dalam hal ini, ilmu mikrobiologi mendapat tantangan dalam membandingkan organisme satu dengan yang lainnya; apakah suatu organisme tersebut memiliki kerabat atau hubungan erat dengan organisme yang memiliki wujud ataupun cara bertahan hidup yang mirip.

Dengan berkembangnya data tentang informasi DNA yang dimiliki, ilmu mikrobiologi saja tidak cukup dalam mengolah data dan melakukan perbandingan antar sequence DNA dari organisme-organisme yang ingin diteliti. Untuk itu, diperlukan ilmu komputasi informatika yang dapat membantu menganalisis sekuens DNA yang ada, membandingkan, dan dapat mengambil kesimpulan dari hasil analisis tersebut.

Bioinformatika adalah aplikasi dalam bidang sains biologi yang berfokus pada analisis data sekuens biologi. Bioinformatika cenderung digunakan dalam komputasi mikrobiologi dan dengan adanya bioinformatika, perkembangan pesat dari data sequence DNA yang muncul hampir tiap hari dapat diolah dan ditangani secara cepat. Bioinformatika mempunyai peran penting dalam pensejajaran atau perbandingan sekuens DNA. Dalam sequence DNA, bioinformatika menggunakan algoritma untuk mensejajarkan suatu DNA dan mencari kecocokan dari DNA tersebut sehingga kemiripan dan korelasi antar DNA dari organisme dapat dianalisis dengan baik.

Untuk melakukan pensejajaran sekuens DNA, program dari ilmu bioinformatika menggunakan berbagai macam algoritma. Pensejajaran DNA memiliki dua macam teknik, yaitu pensejajaran global ataupun lokal. Dalam pensejajaran global, seluruh karakter DNA diproses sementara dalam pensejajaran lokal hanya sebagian dari karakter DNA yang diproses [2]. Beberapa algoritma yang digunakan dalam pensejajaran lokal antara lain adalah Smith-Waterman, FASTA, BLAST, dan masih banyak algoritma yang sedang dikembangkan. Sementara untuk pensejajaran global, algoritma Needleman-Wunsch masih kerap digunakan dan juga dikembangkan agar lebih efisien [4]. Di sisi lain, pensejajaran global mengalami banyak perkembangan dan banyak algoritma-algoritma baru diciptakan dengan mengadaptasi dari beberapa algoritma dasar, salah satunya adalah metode Lempel-Ziv, yaitu metode teknik data compression yang kemudian diadaptasi untuk pensejajaran DNA [5].

Kecepatan proses dan ketepatan pensejajaran, menjadi tolak ukur penting terhadap pensejajaran yang baik oleh suatu algoritma. Untuk itu, mencari keseimbangan yang optimal terhadap kecepatan dan ketepatan pensejajaran menjadi dasar pembuatan aplikasi bioinformatika ini dengan mengimplementasikan dan menganalisis kinerja dua algoritma pensejajaran global yaitu Needleman-Wunsch dan Lempel-Ziv.

Dari studi literatur yang telah dilakukan [11,12] kinerja kedua algoritma ini belum pernah dibandingkan. Oleh karena itu, artikel ini mengangkat hal baru yaitu meneliti lebih jauh mengenai fungsi, analisis, dan perbandingan dua algoritma ini, khususnya mengenai kecepatan dan ketepatan serta implementasinya di dalam sebuah program. Harapannya adalah artikel ini dapat berkontribusi kepada bidang bioinformatika, khususnya, pemahaman tentang kinerja dua algoritma tersebut dan, umumnya, bidang ilmu informatika serta penerapannya dalam biologi seluler.

B. Rumusan Masalah

Dari latar belakang yang sudah disampaikan, maka dapat dirumuskan beberapa masalah yang menjadi dasar pembuatan program bioinformatika, yaitu:

1. Bagaimana membuat implementasi Needleman-wunsch dan Lempel-Ziv ke dalam suatu program?
2. Bagaimana menentukan algoritma yang tepat untuk tiap kasus pensejajaran DNA yang berbeda?
3. Bagaimana proses dan analisis dari kedua algoritma dapat membantu penelitian biologi molekuler?

C. Ruang Lingkup

Ruang lingkup untuk program ini meliputi beberapa hal yaitu:

1. Aplikasi ini akan memiliki dua metode pengolahan sekuens DNA berdasarkan dua algoritma yaitu Needleman-Wunsch dan Lempel-Ziv.
2. Tiap metode pengolahan sekuens DNA akan dites dari maksimal 1000 karakter hingga di atas 1000 karakter.

D. Tujuan Pembahasan

Adapun tujuan dari analisis algoritma bioinformatika, yaitu:

1. Mengembangkan aplikasi bioinformatika yang memiliki proses pengolahan data sekuens dengan algoritma Needleman-Wusnch dan Lempel-Ziv.
2. Memberikan analisis dari proses kinerja tiap algoritma untuk menentukan algoritma yang tepat di tiap kasus berbeda.
3. Memberikan analisis dari kedua algoritma dalam proses pensejajaran DNA untuk membantu memberikan hasil yang cepat dan tepat dalam penelitian biologi molekuler.

E. Manfaat Penelitian

Manfaat penelitian ini mencakup:

1. Mengaplikasikan ilmu yang telah diperoleh selama menempuh pendidikan di Universitas Kristen Maranatha dengan membuat aplikasi bioinformatika untuk memproses pensejajaran dua buah sekuens DNA dan laporan penelitian secara ilmiah dan sistematis.
2. Hasil penelitian ini diharapkan mencadi acuan bagi penyusunan program dan laporan pada pemecahan permasalahan dengan tema yang serupa.

II. TINJAUAN PUSTAKA

Algoritma Needleman-Wunsch dan Lempel-Ziv merupakan algoritma yang memproses sekuens secara *global alginment* yaitu pemrosesan keseluruhan sekuens dalam penyusunan pensejajarannya.

A. Needleman-Wunsch

Algoritma Needleman-Wunsch adalah algoritma yang ditemukan oleh Saul B. Needleman dan Christian D. Wunsch pada tahun 1970. Algoritma ini adalah salah satu algoritma populer dalam *global sequence alignment* [4]. Seiring dengan perkembangan ilmu biologi seluler dan bioinformatika, maka algoritma Needleman-Wunsch mengalami banyak pengembangan dan penyesuaian untuk digunakan pada banyak penelitian biologi seluler. Dengan banyaknya pengguna algoritma Needleman-Wunsch, algoritma ini menjadi pegangan dasar pada masa awal ilmu bioinformatika berkembang.

Algoritma Needleman-Wunsch mempunyai tiga langkah utama yang menjadi proses untuk membangun sebuah pensejajaran dari dua buah sekuens DNA. Langkah pertama

adalah *scoring matrix*, dalam langkah ini akan dibangun sebuah matriks dengan ukuran berdasarkan panjang dari sekuens DNA yang diterima ditambah dengan sebuah *gap* di depan tiap sekuens. Tanda *gap* dapat dituliskan berupa "-" atau "_", dan tanda ini yang nantinya dapat diproses untuk ilmu biologi seluler sebagai tanda mutasi atau *mutation mark* [4]. Setelah *scoring matrix* dibuat maka kolom dan baris *gap* akan diisi berdasarkan skor yang sudah ditentukan dari awal. Penentuan skor berdasarkan tiga buah *event* yaitu *match*, *mismatch*, dan *gap*.

Umumnya skor ini akan diisi dengan 1, -1, dan -1, agar perhitungan *scoring matrix* tidak mengalami proses yang panjang. Skor dapat ditentukan oleh pengguna dengan syarat bahwa nilai *match* harus kontradiksi dengan nilai *mismatch* dan *gap*. Langkah kedua adalah *traceback*, yaitu melacak kembali dari *scoring matrix* yang sudah terisi. Proses *traceback* dimulai dengan mencari nilai terbesar dari kotak matriks terluar dan melakukan *traceback* hingga ke nilai *gap*. Langkah terakhir adalah *alignment*¸ yaitu langkah penyusunan sekuens berdasarkan hasil *traceback*. Dalam langkah ini kedua buah sekuens akan disejajarkan dan kemudian hasil dari pensejajaran akan dihitung berdasarkan nilai *match*, *mismatch*, dan *gap* yang ada pada hasil *alignment* [9].

B. *Lempel-Ziv*

Algoritma Lempel-Ziv ditemukan oleh Abraham Lempel dan Jacob Ziv pada tahun 1977. Algoritma ini memiliki keunikan tersendiri, yaitu fungsi utama untuk algoritma ini adalah kompresi data (*data compression*). Lempel-Ziv kemudian disesuaikan untuk proses *pairwise sequence global alignment*, dengan tetap mengambil fungsi utamanya yaitu *factorisation phrase* dan menyesuaikannya dengan proses *scoring matrix*, *traceback*, dan *alignment* seperti pada proses algoritma Needleman-Wunsch. Proses dari *factorisation phrase* dalam Lempel-Ziv adalah dengan memotong sebuah sekuens menjadi frase-frase yang kemudian disimpan ke dalam tabel *factorisation phrase* dan dijadikan *scoring matrix* setelah sekuens tersebut dipotong menjadi frase-frase. Dalam kasus ini, Lempel-Ziv akan memproses *scoring matrix* dengan perhitungan yang sama dengan Needleman-Wunsch dengan pengecualian, frase untuk tiap kotak matriks tidak akan mengalami proses perhitungan karena sudah dianggap sebagai suatu kesatuan (*phrase*) [10].

III. Analisis Dan Perancangan Sistem

Sistem bioinformatika dirancang untuk mengolah dua buah sekuens DNA dengan kemudian mensejajarkannya sehingga hasil pensejajarannya dapat diteliti dengan sambil meneliti kecepatan masing-masing algoritma dalam memproses sekuens DNA.

A. *Proses Needleman-Wunsch*

Needleman-Wunsch adalah algoritma pensejajaran global yang masih sering digunakan dan juga mengalami banyak penyesuaian guna meningkatkan efisiensi dari algoritma ini sendiri.

Needleman-Wunsch memiliki tiga tahap dalam melakukan pensejajaran, yang pertama adalah membuat matriks penilaian (*scoring matrix*), lalu melakukan *traceback* atau melacak ulang sampai ke awal, dan terakhir yaitu mensejajarkan atau proses *alignment* [4].

Dimulai dengan membuat *scoring matrix*, Needleman-Wunsch akan memproses tiap sekuens yang ada menjadi sebuah matriks dengan menentukan skor untuk tiap *match*, *mismatch*, dan *gap*.

Skor dapat ditentukan oleh pengguna, dengan catatan bahwa nilai mismatch dan *gap* pasti berkontradiksi dengan nilai *match* (plus dan minus). Untuk contoh kali ini, kita menggunakan skor sebagai berikut:

match= +1,mismatch= -1,gap=-2

Setelah skor ditentukan, ada beberapa peraturan penting yang digunakan dalam mengisi matriks skor:

- Saat mengisi skor dari kotak samping ataupun kotak dari bawah, skor yang ditambahkan selalu *gap*.
- Untuk pengisian skor secara diagonal, skor yang ditambahkan adalah tergantung *match* atau *mismatch* dari karakter yang sedang diberikan skor.

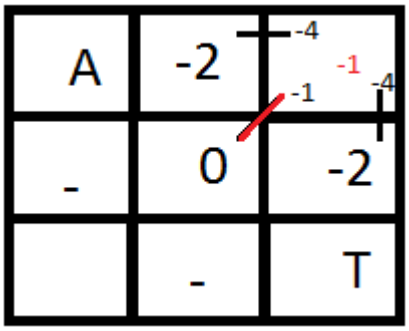
Awal dari perhitungan matriks skor ini adalah dengan mengisi nilai *gap* yang diletakkan di awal tiap sekuens. Kita lihat contoh pensejajaran dua sekuens "ATCG" dan "TCG" dalam membangun matriks skor seperti pada Gambar 1.

G	-8			
C	-6			
T	-4			
A	-2			
-	0	-2	-4	-6
	-	T	C	G

Gambar 1. Proses Awal Matriks Skor

Nilai awal *gap* yang ditandai dengan kotak abu-abu pada gambar, selalu diisi terlebih dahulu karena nilai ini merupakan nilai dasar untuk menghitung matriks skor dari sekuens yang tersedia. Dapat dilihat bahwa skor dari nilai *gap* adalah skor kelipatan -2, skor ini dapat dilihat dari angka berwarna biru pada gambar, karena sesuai aturan

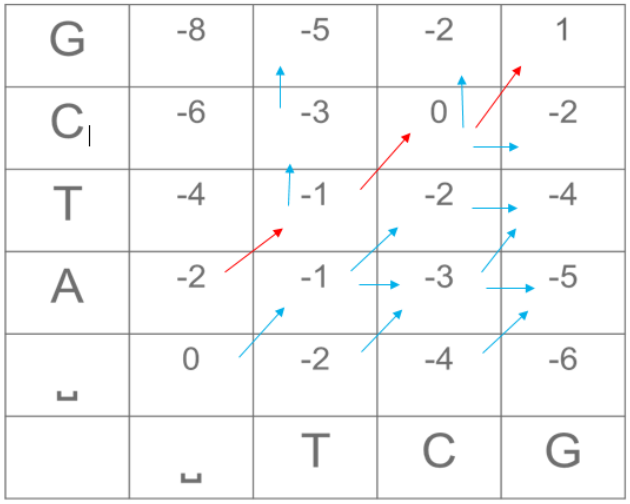
awal bahwa kotak dari samping maupun dari bawah harus ditambahkan dengan skor *gap*. Setelah semua skor *gap* terisi, maka proses dapat dimulai dengan karakter "A" dan "T". Pengisian skor ditentukan dengan menghitung dari tiap sudut baik itu samping, bawah, dan diagonal dengan mengikuti aturan di atas yaitu untuk pengisian skor dari samping dan bawah ditambah nilai gap dan pengisian skor dari diagonal ditentukan oleh nilai *match* atau *mismatch* dari karakter. Setelah mendapatkan skor dari masing-masing sudut, maka skor terbesar menjadi skor inisial di kotak tersebut Seperti yang bisa dilihat pada Gambar 2



Gambar 2. Proses Pengisian Tabel Matriks Skor

Gambar 2 menunjukkan asal dari nilai untuk pengisian kotak matriks skor. Pertama, nilai dari samping kiri didapatkan dengan aturan pengisian matriks skor, yaitu dengan menambah nilai asal (-2) dengan nilai *gap* (-2) maka didapatkan nilai -4, kemudian nilai dari bawah juga mendapatkan hasil -4 karena memiliki aturan penambahan nilai yang sama dengan nilai dari samping kiri. Untuk nilai dari diagonal, maka yang ditambah adalah nilai *match*/*mismatch* dari karakter yang bertemu di kotak tersebut (saat ini karakter yang bertemu adalah "A" dan "T"), sehingga karakter "A" dan "T" adalah mismatch (-1) ditambahkan dengan 0 sehingga mendapatkan nilai -1. Nilai yang tertinggi dari hasil penambahan baik dari samping, bawah, dan diagonal, akan diambil dan menjadi isi dari kotak matriks tersebut dan asal dari nilai tertinggi akan mendapatkan tanda (pada gambar ditandai dengan garis merah) untuk digunakan nantinya pada proses *traceback*.

Dengan demikian, maka tiap asal dari skor tertinggi akan memiliki *pointer* atau panah khusus yang menandakan asal dari skor tersebut sehingga didapat tabel matriks skor yang penuh terisi beserta *pointer* untuk tiap asal dari nilai tertinggi dalam sebuah kotak seperti pada Gambar 3:



Gambar 3. Hasil pengisian Matriks Skor

Setelah matriks skor terisi penuh, akan dilakukan *traceback* yaitu pengecekan jalur ke titik awal yang dimulai dari nilai tertinggi pada kotak luar (*outer box*). Pada Gambar 3, kita menemukan nilai tertinggi pada *outer box* adalah 1 (di kotak paling kanan bagian atas matriks), sehingga kita dapat memulai proses *traceback*. Saat ini yang menjadi patokan dalam *traceback* adalah panah merah yang menjadi tanda dari asal nilai untuk tiap kotak matriks skor. *Traceback* memiliki aturan sederhana, yaitu saat asal traceback dari arah diagonal maka pada alignment dituliskan karakter, sedangkan jika traceback berasal dari kotak bawah atau samping, maka dituliskan gap sesuai arah panah.

Proses *alignment* dilakukan bersamaan dengan proses *traceback*, yaitu saat melakukan *traceback* maka *alignment* juga ditulis sesuai karakter atau *gap* yang ditemukan sehingga berdasarkan matriks skor di atas dapat kita lakukan *alignment* seperti pada Gambar 4.

A	T	C	G
	|	|	|
-	T	C	G
-2	+1	+1	+1

Gambar 4. Hasil *Alignment* Needleman-Wunsch

Setelah proses *alignment* selesai kita dapat mencari tahu apakah perhitungan yang dilakukan sudah benar atau tidak berdasarkan skor yang diperoleh dari *alignment*. Di sini kita mendapatkan +3 (dari *match*) dan -2 (dari *gap*) berdasarkan hasil skor *alignment*, sehingga:

+3-2=1

Angka 1 adalah skor dari *alignment* dan perlu diperhatikan bahwa nilai ini akan selalu sama dengan angka terbesar dari *outer box* matriks skor.

B. Lempel-Ziv

Lempel-Ziv adalah metode *data compression* yang disesuaikan untuk pensejajaran global DNA. Awalnya metode ini hanya mencari frase dari sebuah *string* yang panjang dan membagi-bagi frase tersebut menjadi faktor-faktor dan menyimpan tiap frase tersebut. Namun Lempel-Ziv kemudian dikembangkan dan digunakan untuk pensejajaran global dengan menggunakan teknik *data compression* dan melakukan alignment seperti Needleman-Wunsch.

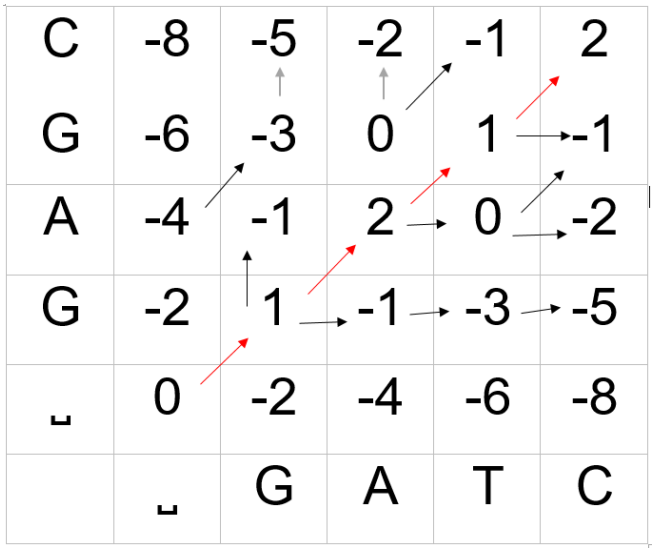
Faktorisasi dari Lempel-Ziv adalah dengan membagi-bagi suatu rangkaian string menjadi beberapa frase, sehingga string yang panjang akan terpotong-potong menjadi potongan-potongan frase. Misalnya sebuah *string* DNA "AGAACAAGGCGT" yang akan dipecah menjadi beberapa frase, seperti pada Gambar 5. Proses pembuatan frase dari Lempel-Ziv dimulai dengan menuliskan P (*phrase*) kosong, dan P digabungkan dengan C (karakter selanjutnya) di mana karakter dari sekuens dimulai dengan "A", jika karakter "A" belum tercatat dalam P, maka karakter "A" dimasukkan ke dalam P. Selanjutnya berlaku hal yang sama dengan karakter setelah "A" yaitu "G", setelah "G" masuk ke dalam P, maka karakter selanjutnya adalah "A" di mana "A" telah tersedia dalam P sehingga "A" akan ditambahkan dengan C atau karakter selanjutnya, sehingga menjadi "AA" yaitu P dengan serial number 1 ditambahkan dengan karakter A dan menjadi faktorisasi (1,A), dan "AA" yang belum tercatat di dalam P akan masuk menjadi P yang baru. Proses ini berlangsung terus menerus hingga seluruh karakter tercatat ke dalam P [8].

Serial Number	Factor	Phrase
0	Empty	(empty string)
1	(0 , A)	A
2	(0 , G)	G
3	(1 , A)	AA
4	(0 , C)	C
5	(3 , G)	AAG
6	(2 , C)	GC
7	(2 , T)	GT

Gambar 5. Lempel-Ziv Factorisation Phrase

Dengan metode ini, Lempel-Ziv akan mengelompokkan frase tersebut di dalam matriks skor dan memulai proses pengisian matriks, *traceback*, dan *alignment* sesuai dengan kinerja Needleman-Wunsch.

Contohnya pada saat membuat matriks skor dari dua sekuens "GAGC" dan "GATC", maka sekuens "GAGC" akan mengalami proses faktorisasi frase menjadi "G", "A", dan "GC" dan hasil dari frase ini akan diimplikasikan ke dalam matriks skor, dengan "GAGC" yang ditulis secara tegak lurus dengan arah bawah ke atas dan "GATC" ditulis secara mendatar, seperti pada Gambar 6.



Gambar 6. Proses Pengisian Matriks Skor Lempel-Ziv

Seperti yang kita lihat, untuk pengisian matriks skor Lempel-Ziv, *pointer* dalam tiap frase (yang ditandai dengan panah abu-abu) diabaikan karena dianggap sebuah frase sehingga hanya berfokus pada *pointer* diagonal untuk *traceback* dan *pointer* sekitarnya yang mempunyai frase yang berbeda. Dari hasil traceback maka didapat *alignment* sebagai berikut:

G	A	G	C
|	|		|
G	A	T	C
+1	+1	-1	+1

Gambar 7. Hasil Alignment Lempel-Ziv

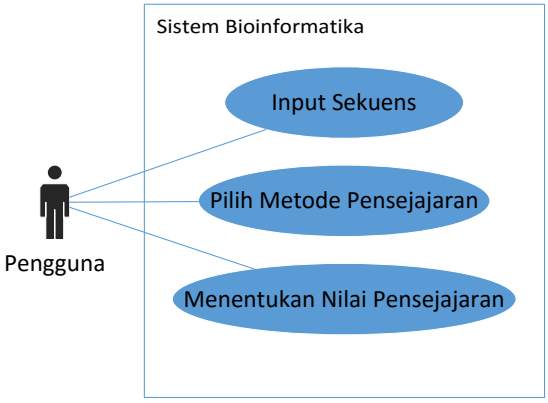
Hasil perhitungan dari *alignment* di atas adalah:

+3-1=2

Angka 2 adalah skor tertinggi dari outer box yang terdapat pada matriks skor.

C. Use Case Diagram

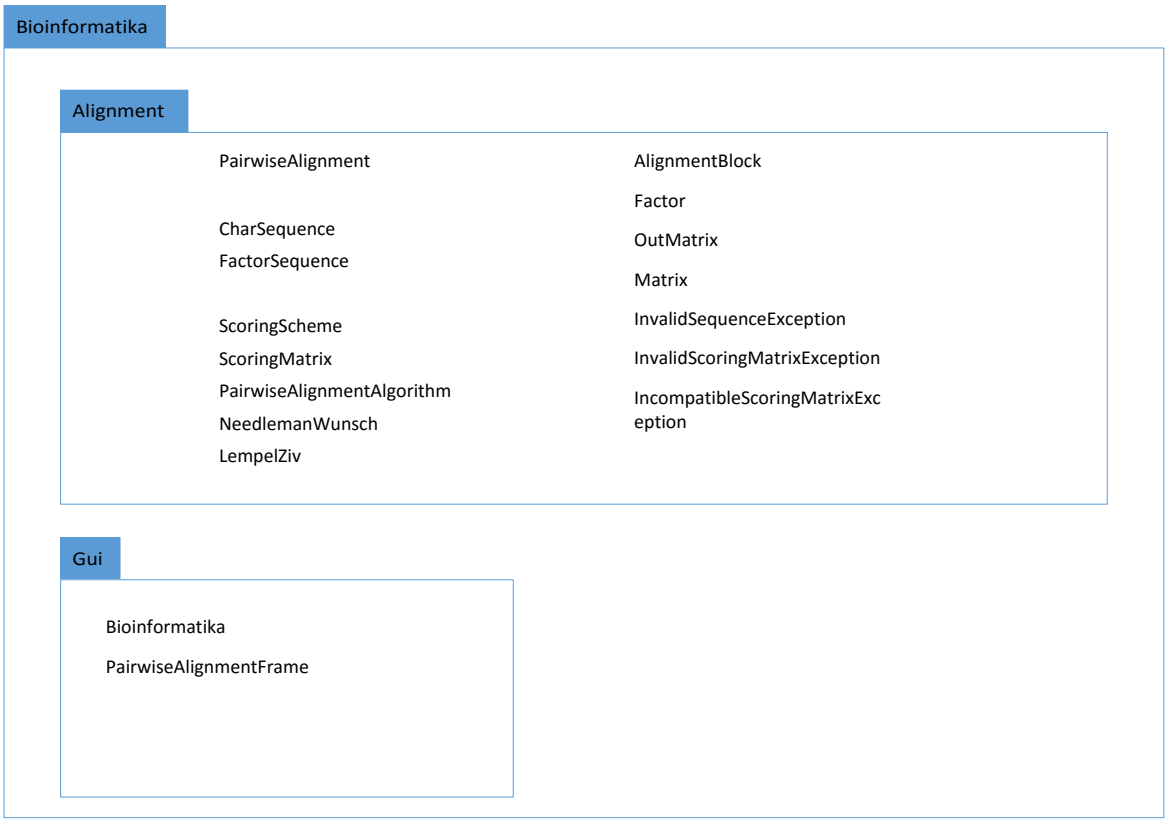
Perangkat lunak yang dimodelkan akan mempunyai diagram use case seperti pada Gambar 8.



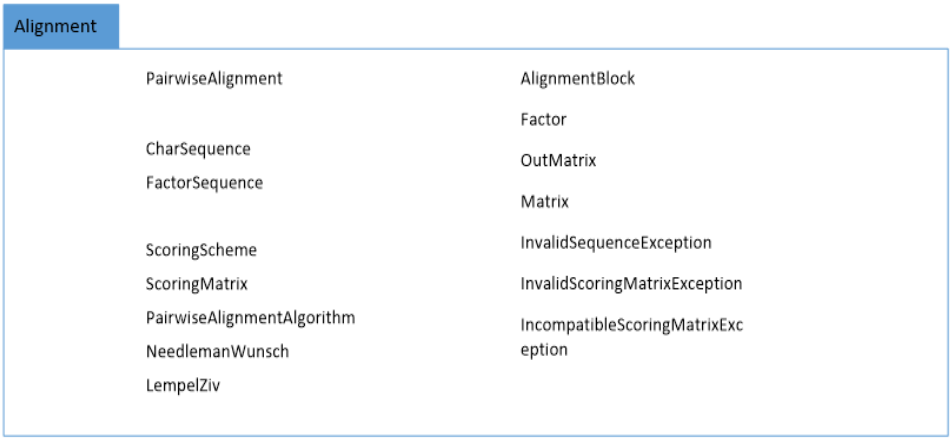
Gambar 8. Use Case Diagram Bioinformatika

D. Package dan Class Diagram

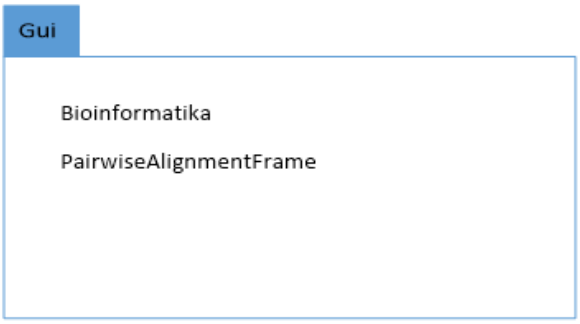
Gambar 9 menampilkan rancangan *package* dan *class diagram* dari aplikasi Bioinformatika secara keseluruhan. Untuk detilnya, Gambar 10 menggambarkan kelas-kelas yang berada dalam *package* Alignment dan Gambar 11 adalah kelas-kelas yang berada dalam *package* Gui. Adapun fungsi utama dari beberapa class untuk proses *alignment* didesain sebagai pada Gambar 12.



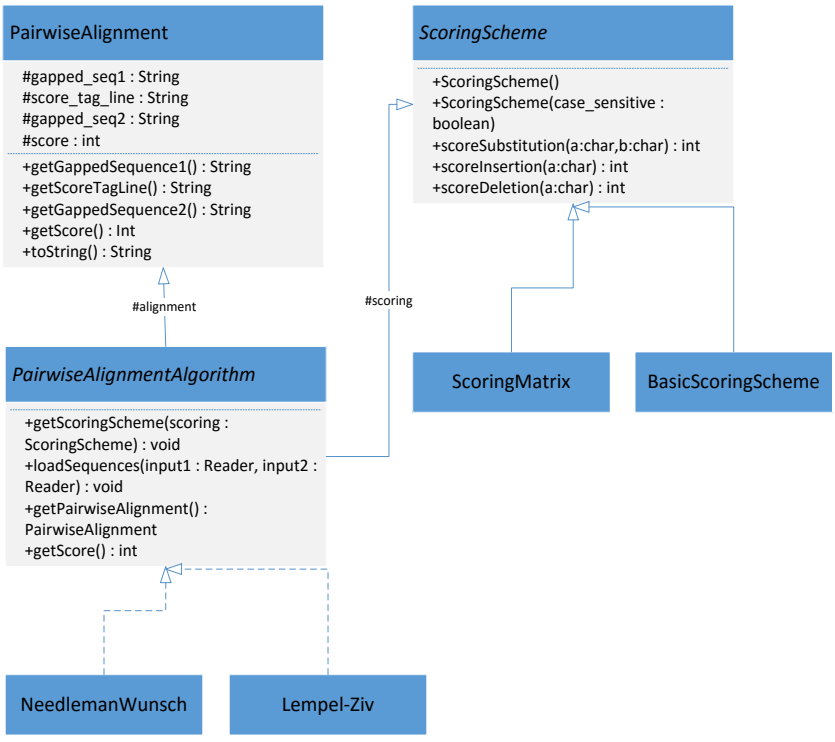
Gambar 9. *Package* Bioinformatika mencakup package alignment dan Gui (Untuk lebih jelasnya, Gambar 10 mendeskripsikan *package* Alignment dan Gambar 11 menggambarkan *package* Gui)



Gambar 10. Package Alignment yang berada di dalam package Bioinformatika



Gambar 11. Package Gui yang berada di dalam package Bioinformatika



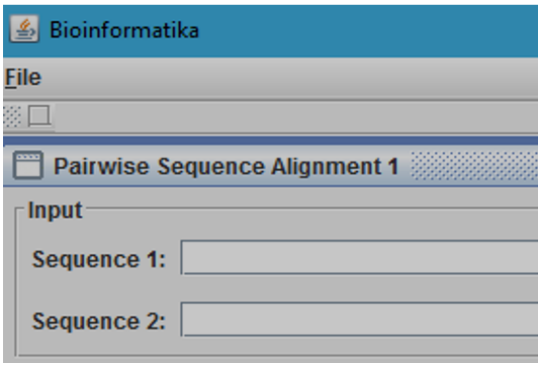
Gambar 12. *Class Diagram* Bioinformatika

IV. IMPLEMENTASI

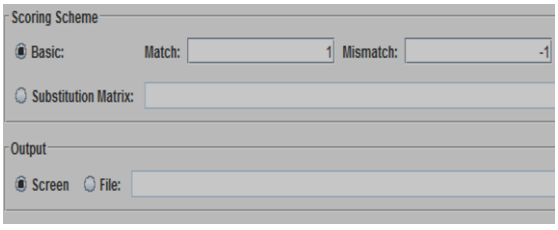
Aplikasi Bioinformatika ini memiliki tampilan utama tunggal berupa GUI (*Graphical User Interface*) yang memungkinkan pengguna untuk memilih dua buah sekuens DNA yang akan diproses kemudian pengguna dapat menentukan *scoring* scheme untuk penilaian kecocokan sequence dan aplikasi akan menampilkan pensejajaran dua buah sequence DNA dengan jumlah *match* dan skor *alignment*-nya serta waktu yang dibutuhkan untuk proses pensejajaran dua buah sequence DNA.

A. *Tampilan Awal Aplikasi*

Gambar 13 menampilkan bagian input yang menerima dua sequence. Selanjutnya, Gambar 14 menampilkan kolom *output* yang dapat dipilih oleh pengguna secara langsung apa yang ditampilkan pada layar, atau berupa *file* .txt, hal ini membantu agar sekuens yang panjang dapat terlihat lebih jelas. Pengguna dapat menentukan algoritma mana yang ingin digunakan untuk memproses alignment dari sekuens yang diberikan, dan tombol *run* untuk memulai proses *alignment*. Jika pengguna memilih *screen* sebagai hasil *output*, maka hasil *alignment* akan keluar pada layar *output* di bawah. *Tab progress* adalah proses *alignment* yang dilakukan, dari membaca sekuens hingga menyelesaikan alignment beserta dengan waktu yang dibutuhkan untuk keseluruhan proses.



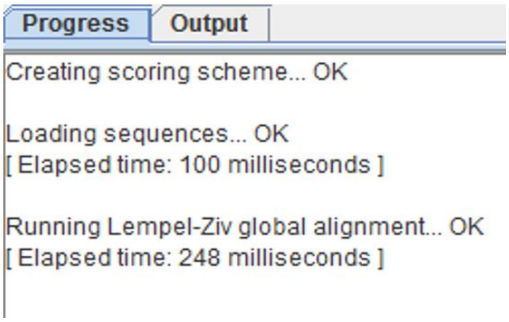
Gambar 13. Tampilan Awal Aplikasi



Gambar 14. Tampilan untuk memilih *scoring scheme* dan *output*

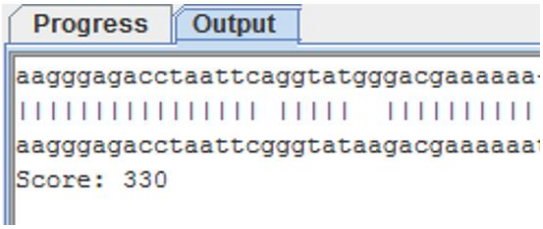
B. *Tampilan Progress dan Output*

Setelah pengguna menekan tombol *run*, algoritma yang dipilih akan berjalan dan tab progress akan menampilkan proses dari algoritma tersebut, serta waktu yang dibutuhkan untuk memprosesnya seperti tampilan pada Gambar 15, ini sangat penting mengingat bahwa kecepatan dari suatu algoritma akan didata untuk tiap masing-masing kasus *alignment* .



Gambar 15. Tampilan *Progress*

Setelah proses selesai, jika pengguna memilih screen sebagai output maka tab output akan menampilkan hasil alignment dari sekuens yang diberikan, serta skor dari alignment tersebut seperti pada Gambar 16.



Gambar 16. Tampilan *Output*

V. PENGUJIAN

Pengujian aplikasi Bioinformatika dilakukan dengan penggunaan sampel *dataset* dari NCBI (*National Center for Biotechnology Information*). *Dataset* dari NCBI mempunyai banyak varian terhadap DNA yang mencakup berbagai macam sekuens bakteria seperti *sphingomonas sp., synechococcus elongatus, beberapa varian dari campylobacter, brevundimonas sp. complete sequence* untuk sekuens dengan panjang di atas 1000, beberapa varian dari *bacillus sp., sphingobacterium thalpophilum, pseudomonas sp., mesorhizobium sp., cellulosimicrobium cellulans*, beberapa varian dari *staphylococcus sp., chryseobacterium sp., comamonas sp., variovorax sp., moraxella osloensis*, serta berbagai varian dari *geobacillus thermodenitrificans*. Dataset dibagi menjadi 2 kategori yaitu: maksimal 1000 karakter dan yang lebih daripada 1000 karakter.

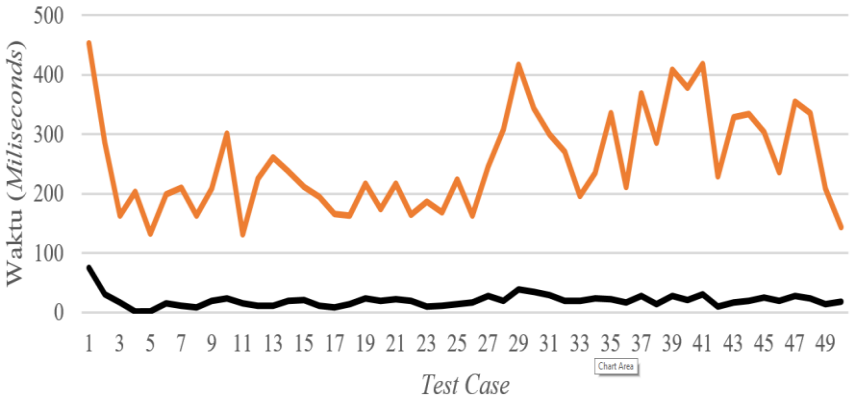
A. *Pengujian Sekuens Maksimal 1000 Karakter*

Pengujian ini dilakukan untuk 50 sekuens yang memiliki karakter kurang dari 1000. Pengujian dilakukan untuk tiap algoritma dengan pengukuran waktu pemrosesan dan skor dari sekuens yang diberikan.

TABEL I
PENGUJIAN SEKUENS MAKSIMAL 1000 KARAKTER

Nomor	Panjang Sekuens	Waktu Proses Needleman-Wunsch (Dalam *Miliseconds*)	Waktu Proses Lempel-Ziv (Dalam *Miliseconds*)	Skor
1	800	75	378	72
2	800	31	254	86
3	800	16	147	25
4	800	2	201	85
5	800	1	131	73
6	800	15	184	57
7	800	11	199	77
8	800	8	155	44
9	800	19	190	71
10	800	24	277	48
11	900	15	116	113
12	900	11	214	82
13	900	11	250	79
14	900	19	218	77
15	900	21	190	49
16	900	11	183	85
17	900	8	157	77
18	900	14	149	59
19	900	23	194	61
20	900	19	155	49
21	900	22	194	57
22	900	19	145	84
23	900	9	177	40
24	900	11	158	54
25	900	14	210	88
26	1000	16	147	960
27	1000	27	218	510
28	1000	19	288	560
29	1000	39	377	862
30	1000	34	310	477
31	1000	29	270	582
32	1000	19	251	377
33	1000	19	177	874
34	1000	24	211	421
35	1000	22	313	642
36	1000	16	195	771
37	1000	27	341	529
38	1000	14	271	719
39	1000	28	380	711
40	1000	21	357	694
41	1000	31	387	710
42	1000	10	219	814
43	1000	17	311	411
44	1000	19	315	575
45	1000	25	279	810
46	1000	19	217	418
47	1000	28	327	520
48	1000	24	311	731
49	1000	14	195	644
50	1000	18	125	741

Tabel I menunjukkan bahwa proses untuk *dataset* dengan panjang sekuens maksimal 1000, algoritma Needleman-Wunsch memiliki keunggulan dalam waktu proses, sementara skor match yang didapat oleh tiap algoritma memiliki kesamaan karena proses perhitungan match untuk tiap sekuens sama dengan perbedaan yang terdapat pada pembentukkan awal matriks, untuk Needleman-Wunsch, *scoring matrix* yang dibuat langsung sesuai karakter sekuens yang tersedia sementara Lempel-Ziv sudah melalui proses *factorisation phrase*. Tabel di atas dapat kita buat grafik untuk lebih jelas melihat kecepatan masing-masing algoritma untuk tiap panjang sekuensnya seperti pada Gambar 17.



Gambar 17. Grafik Hasil Pengujian Maksimal 1000 Karakter (grafik atas merupakan Lempel-Ziv sedangkan grafik bawah adalah Needleman-Wunsch)

Hasil penelitian dari grafik di atas menunjukkan bahwa waktu proses Needleman-Wunsch (dalam *miliseconds*) memiliki keunggulan dalam kecepatan dibandingkan dengan waktu proses Lempel-Ziv. Lempel-Ziv memiliki grafik yang cenderung tinggi untuk waktu pemrosesan dibandingkan dengan Needleman-Wunsch. Sementara skor untuk tiap *test case alignment*, meningkat seiring dengan panjang sekuens yang diberikan, karena skor *match* yang dihasilkan tergantung dari panjang sekuens, semakin panjang suatu sekuens maka skor *match* cenderung akan bertambah.

B. Pengujian Sekuens Di Atas 1000 Karakter

Pengujian ini dilakukan untuk mengetes kecepatan tiap algoritma dalam memproses sekuens yang panjang, dengan metode yang sama, dan tolak ukur pengukuran yang sama yaitu waktu dan skor dari sekuens yang diberikan. Tabel II menampilkan panjang sekuens, waktu proses, dan skor.

TABEL II
ACUAN UKURAN TEKS

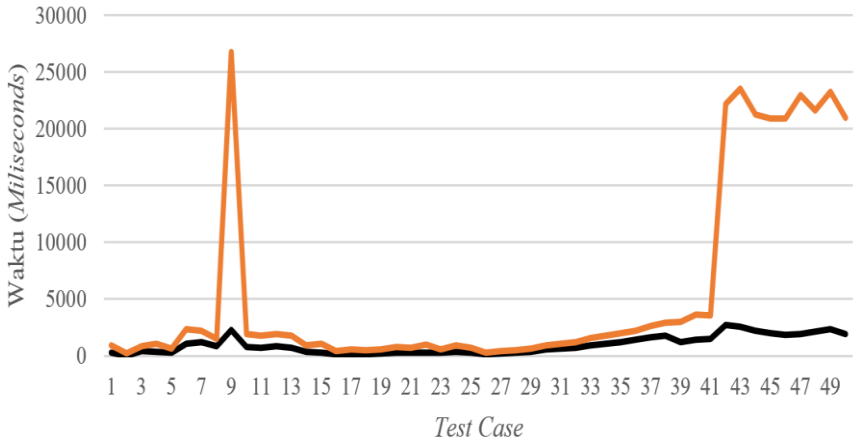
Nomor	Panjang Sekuens	Waktu Proses Needleman-Wunsch (Dalam *Miliseconds*)	Waktu Proses Lempel-Ziv (Dalam *Miliseconds*)	Skor
1	2700	314	580	570
2	1500	42	221	305
3	3800	411	442	415
4	3800	388	671	515
5	2400	320	315	425
6	5000	1071	1280	751
7	5000	1204	1007	401
8	3750	852	641	399
9	8000	2309	24446	489
10	3500	811	1088	390
11	3500	725	1051	305
12	3500	890	1004	425
13	3500	715	1094	377
14	2700	387	508	157
15	2700	315	788	229
16	1800	133	271	279
17	1800	154	391	422
18	1800	177	297	310
19	2000	241	311	471
20	2000	311	504	308
21	2000	279	431	379
22	2500	311	671	310
23	2500	305	271	339
24	3200	391	551	477
25	3200	255	479	311
26	1100	140	140	462
27	1250	207	203	467
28	1500	284	240	553
29	1750	370	287	408
30	2000	540	373	449
31	2250	611	430	552
32	2500	734	494	308
33	2750	961	587	441
34	3000	1081	684	309
35	3250	1235	757	419
36	3500	1402	824	312
37	3750	1622	1048	339
38	4000	1796	1118	413
39	5000	1214	1788	379
40	5000	1425	2210	440
41	5000	1490	2115	376
42	8000	2711	19515	481
43	8000	2588	20980	455
44	8000	2191	19067	387
45	8000	1966	18950	410
46	8000	1891	19001	344
47	8000	1911	21085	557
48	8000	2127	19505	481
49	8000	2322	20915	479
50	8000	1902	19115	360

Dari data di atas, dapat dilihat bahwa proses untuk dataset dengan panjang sekuens di atas 1000 karakter, algoritma Lempel-Ziv memiliki keunggulan di beberapa kasus sekuens yang setelah diteliti lebih lanjut, dataset sekuens yang memberikan hasil berbeda terhadap kinerja algoritma Lempel-Ziv adalah dataset dengan sekuens yang memiliki faktorisasi sempurna. Faktorisasi sempurna dapat dicontohkan oleh sekuens "A A C A C G A C G T A C G T T A C G T T A A C G T T A G A C G T T A G C".

TABEL III
CONTOH FAKTORISASI SEMPURNA LEMPEL ZIV

Serial Number	Factor	Phrase
0	Empty	(empty string)
1	(0 , A)	A
2	(1 , C)	AC
3	(2 , G)	ACG
4	(3 , T)	ACGT
5	(4 , T)	ACGTT
6	(5 , A)	ACGTTA
7	(6 , G)	ACGTTAG
8	(7 , C)	ACGTTAGC

Faktorisasi sempurna dapat dilihat dari frase yang terurut secara bertahap dengan pengulangan dari tiap frase yang sudah tercatat di atasnya dengan tambahan satu karakter selanjutnya dari tiap frase seperti pada Tabel III. Frase yang membentuk faktorisasi sempurna memungkinkan perhitungan yang lebih efisien dari algoritma Lempel-Ziv.
Tabel hasil pengujian dapat kita buat grafik untuk lebih jelas melihat kecepatan masing-masing algoritma untuk tiap panjang sekuensnya seperti pada Gambar 18.



Gambar 18. Grafik Hasil Pengujian Di Atas 1000 Karakter (grafik berwarna hitam adalah Needleman-Wunsch dan grafik lainnya adalah Lempel-Ziv)

Hasil penelitian dari grafik di atas menunjukkan bahwa waktu proses Lempel-Ziv memiliki keunggulan dalam proses alignment seperti yang kita lihat pada hasil *test dataset* dari nomor 26 hingga 38. Dataset tersebut memiliki faktorisasi sempurna dari sekuens yang diberikan sehingga Lempel-Ziv mempunyai hasil proses yang lebih cepat pada rentang *dataset* ini.

C. *Pengujian Spesifik Lempel-Ziv Factorisation Phrase Terhadap Nilai Gap*

Pengujian ini ditujukan untuk melihat bagaimana pengaruh dari proses *factorisation phrase* terhadap nilai *gap* yang dihasilkan pada saat *sequence alignment*. Data yang diuji akan sedikit berbeda karena menggunakan contoh DNA virus yang didapatkan dari NCBI. Virus yang diuji adalah *hepatitis A complete genome* dan *hepatitis B complete genome*.

Gambar 19. Hasil Lempel-Ziv Alignment Sekuens Hepatitis A dan Hepatitis B

Pada Gambar 19 dapat dilihat potongan dari hasil *alignment* dari kedua buah virus *hepatitis* A dan *hepatitis* B. Dengan sekuens yang sama, tes ini juga dijalankan dengan menggunakan algoritma Needleman-Wunsch dan hasil *alignment* yang muncul adalah sama. Dari hasil ini dapat kita lihat bahwa virus *hepatitis* A dan *hepatitis* B memiliki banyak nilai *gap* terhadap satu sama lain, sehingga dapat diambil kesimpulan bahwa *hepatitis* B telah mengalami banyak proses mutasi sehingga virus ini dikenal lebih berbahaya dibandingkan dengan virus *hepatitis* A, dan karena hasil *alignment* yang didapatkan oleh kedua buah algoritma adalah sama, maka tidak terjadi perubahan dalam proses *factorisation phrase* dari Lempel-Ziv terhadap nilai *gap*.

VI. Simpulan Dan Saran

Simpulan yang dapat diambil dari hasil pembuatan aplikasi Bioinformatika ini adalah:

1. Aplikasi Bioinformatika dapat memproses alignment dari dua buah sekuens DNA yang dipilih dengan

algoritma Needleman-Wunsch dan algoritma Lempel-Ziv.

2. Algoritma Needleman-Wunsch memiliki keunggulan dalam *alignment* sekuens dengan panjang karakter kurang dari 1000, Needleman-Wunsch memiliki waktu proses dengan nilai minimal 1 miliseconds dan nilai maksimal 75 miliseconds. sementara algoritma Lempel-Ziv memiliki waktu kinerja yang lebih lama dengan waktu proses minimal 25 *miliseconds* dan maksimal 960 *miliseconds* dalam penelitian untuk karakter dengan panjang maksimal 1000, namun Lempel-Ziv memiliki keunggulan dalam alignment untuk sekuens dengan faktorisasi sempurna. Faktorisasi sempurna dapat dilihat dari frase yang terurut secara bertahap dengan pengulangan dari tiap frase yang sudah tercatat di atasnya dengan tambahan satu karakter selanjutnya dari tiap frase. Needleman-Wunch dapat digunakan untuk pensejajaran sekuens pada umumnya namun Lempel-Ziv memiliki keunggulan dalam proses pengolahan *dataset* dengan faktorisasi sempurna.
3. Algoritma Needleman-Wunsch dan Lempel-Ziv dapat memberikan hasil *alignment* dengan visualisasi *gap* yang dapat dilihat jelas oleh pengguna, sehingga hasil alignment tersebut dapat digunakan untuk penelitian dalam bidang biologi molekuler untuk mencari tanda mutasi (*mutation mark*) dalam suatu kasus pensejajaran DNA.
4. Lempel-Ziv *factorisation phrase* memiliki keunggulan hanya dalam kasus sekuens dengan *perfect factorisation phrase*. Namun proses *factorisation phrase* tidak mengubah nilai *gap* terhadap hasil *alignment* sehingga data hasil pengujian tetap valid.

Saran pengembangan aplikasi Bioinformatika yang dapat diberikan adalah sebagai berikut:

1. Memberikan batasan karakter maksimal untuk pengolahan sekuens yang tidak melebihi kapasitas memori komputer/laptop yang digunakan, karena untuk saat ini, proses *alignment* dengan karakter yang melebihi 8000 akan mengalami masalah kelebihan pemakaian memori.
2. Memberikan *suggestion* terhadap algoritma Lempel-Ziv yang memiliki keunggulan dalam pemrosesan data untuk sekuens yang memiliki faktorisasi sempurna. *Suggestion* dapat berupa disclaimer yang terdapat di aplikasi, sehingga pengguna mengerti bagaimana tiap algoritma menangani sekuens yang diberikan.

Memisahkan data sekuens untuk Lempel-Ziv yang sudah mengalami factorisation *phrase* sehingga perhitungan untuk proses Lempel-Ziv diharapkan memiliki waktu yang tidak jauh berbeda dengan Needleman-Wunsch.

DAFTAR PUSTAKA

[1] Z. R. Yang, Machine Learning Approaches to Bioinformatics, Singapore: World Scientific Publishing Co. Pte. Ltd., 2010.

[2] W. S. Susan Elrod, Schaum's Genetika, Jakarta: Erlangga, 2007.

[3] J. Simarmata, Rekayasa Perangkat Lunak, Yogyakarta: Penerbit ANDI, 2010.

[4] S. J. P. J. Z. Shanika Kuruppu, "Optimized Relative Lempel-Ziv Compression of Genomes," vol. XXXIV, pp. 1-8, 2011.

[5] V. Likic, "The Needleman-Wunsch Algorithm for Sequence Alignment," in *Bio21 Molecular Science and Biotechnology Institute*, Melbourne, 2008.

[6] R. S. Harris, Improved Pairwise Alignment Of Genomic DNA, Pennsylvania: Pennylvania State University, 2007.

[7] D. P. A. P. Ernawati, "Implementasi Algoritma Smith-Waterman Pada Local Alignment Dalam Pencarian Kesamaan Pensejajaran Barisan DNA (Studi Kasus : DNA Tumor Wilms)," *Jurnal Pseudocode*, pp. 170-177, 2014.

[8] S. J. P. Dominik Kempa, "Lempel-Ziv Factorization: Simple, Fast, Practical," *Algorithm Engineering and Experiments (ALENEX)*, pp. 103-112, 2013.

[9] J. C. G. C. N. L. J. L. J. F. D. B. S. C. L. Dandan Song, "Parameterized BLOSUM Matrices for Protein Alignment," *Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 3, pp. 686-694, 2015.

[10] A. Chan, "Stanford Project," 2004. [Online]. Available: biochem218.stanford.edu/Projects%202004.Chan.pdf.

[11] W. H. B. R. Alex Aravind, "Pairwise sequence alignment algorithms : a survey," *Association for Computing Machinery Journal*, 2016.

[12] D. Agashe, "Large-Effect Beneficial Synonymous Mutations Mediate Rapid and Parallel Adaptation in a Bacterium," *Molecular Biology and Evolution*, vol. 33, no. 6, pp. 1542-1553, 2016.