CS612 – Algorithms in Bioinformatics

Homework Assignment 1 – 02/16/2023

Aravind Haridas - 02071139

**Part 1 – Practice**

1. Sequence alignment hands-on exercise. We have given the following protein sequence:

>protein TCPFADPAALYSRQDTTSGQSPLAAYEVDDSTGYLTSDVGGPIQDQTSLKAGIRGPTLLEDFMFRQKIQHFDHERVPERAV

The sequence is also available at the course webpage: http://www.cs.umb.edu/ nurith/cs612/protein.fasta .

Solution:

1. In this protein sequence we need to perform the protein Blast from <http://blast.ncbi.nlm.nih.gov>.

Select “Protein Blast” to get to the **BlastP** website. Paste the above sequence to the top window. Use the default parameters – **nr** database (nonredundant protein sequence database). We will be getting the following.



1. In this we need to Repeat the search above with the **UniprotKB**/**SwissProt** database and attach it first 2 pages as part of your homework

We will be getting the following as the output.



1. Repeat search a. above with **PAM30** as a substitution matrix. This can be done in the **blastP** homepage by opening “algorithm parameters” at the bottom of the page.

We can have the following differences highlighted in the attachment.

We have below ranked for A and C and name of the protein sequences are as above.

Default matrix

Graphical user interface

Description automatically generated with medium confidence

Substitution matrix

Graphical user interface, application

Description automatically generated

The above are the ranks in a and c.

We have the below as the output file while extracting the structure.

2. DNA sequence alignment: The following sequence was constructed by NCBI scientist Mark Boguski for Michael Chrichton’s “The Lost World” of the Jurassic Park series:

The sequence is available at the course webpage:

http://www.cs.umb.edu/ nurith/cs612/dino.fasta

(a) What are the two main species used to construct the dinosaur DNA sequence?

(I) [Gallus gallus](https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Info&id=9031&lvl=3&lin=f&keep=1&srchmode=1&unlock) (chicken)

(ii)  [Xenopus laevis](https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Info&id=8355&lvl=3&lin=f&keep=1&srchmode=1&unlock) (African clawed frog)

(b) with Blastx

(i) [Gallus gallus](https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Info&id=9031&lvl=3&lin=f&keep=1&srchmode=1&unlock) (chicken)

(ii) Cygnus olor (mute swan )

(iii) [Parus major](https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Info&id=9157&lvl=3&lin=f&keep=1&srchmode=1&unlock) (Great Tit)

(iv)  Catharus ustulatus (Swainson's thrush)

b. Now we need to Repeat the search with blastx (DNA vs. protein sequence) using the default non-redundant protein sequence database. Look at the top sequence alignment we get the below ones.



**Part 2 – Theory**

1. Lesk book question 5.3: The edit distance between the strings agtcc and cgctca is 3, consistent with the following alignment:

Solution: To convert the string "agtcc" to "cgctca" using a minimum of 3 edit operations, we can use the following sequence of operations:

Replace "a" with "c" at position 1: "cgtcc"

agtcc

cgtcc

Insert "g" at position 2: "cggtcc"

agtcc

cggtcc

Replace "c" with "a" at position 4: "cggtca"

agtcc

cggtca

This sequence of operations results in the desired string "cgctca" and requires a minimum of 3 edit operations: one replacement, one insertion, and one replacement.



1. Dynamic programming:
2. Use the Needleman Wunsch global alignment Dynamic programming formula in slide set no. 2 to find the sequence alignment score of the two DNA sequences ATCGAACTGCC and TACGCACTCCA.



1. Smith waterman local alignment



1. Semi global alignment



3.a) Given the Longest Common Subsequence (LCS) of two sequences X and Y, you can obtain the shortest common super sequence (SCS) of the two sequences by following these steps:

Initialize two pointers, one pointing to the start of the LCS and the other pointing to the start of the sequences X and Y, respectively.

Iterate through the LCS, and for each element in the LCS, do the following:

Append all the characters in X and Y that come before the current element of the LCS to the SCS.

Append the current element of the LCS to the SCS.

Move the X and Y pointers to the characters immediately following the current element in the LCS.

After iterating through the entire LCS, append all the remaining characters in X and Y to the SCS.

The resulting sequence is the shortest common super sequence of X and Y.

For example, suppose we have two sequences X = "ABCBDAB" and Y = "BDCABA", and their LCS is "BCBA". To obtain the SCS of X and Y, we can follow these steps:

Initialize two pointers, one pointing to the start of the LCS "BCBA" and the other pointing to the start of X ("A") and Y ("B"), respectively.

Iterate through the LCS "BCBA", and for each element in the LCS, do the following:

Append all the characters in X and Y that come before the current element of the LCS to the SCS. In this case, we append "A" to the SCS.

Append the current element of the LCS to the SCS. In this case, we append "B", "C", "B", and "A" to the SCS.

Move the X and Y pointers to the characters immediately following the current element in the LCS. In this case, we move the X and Y pointers to "D" and "D", respectively.

After iterating through the entire LCS, append all the remaining characters in X and Y to the SCS. In this case, we append "D", "A", and "B" to the SCS.

The resulting sequence is "ABCBADAB".

b. We are given two sequences that are TACGGGTAT and GGACGTACG.

To find the shortest common sequence of two strings, we can use a technique called the

"Longest common substring" algorithm, which finds the longest substring that is common to both strings. However, in this case, there is no common substring that is shared between the two sequences.

TACGGGTAT

GGACGTACG

Therefore, the shortest common sequence of "TACGGGTAT" and "GGACGTACG" is simply an empty sequence or string "" or we can have the whole sequences

**“TGGACGGGTATCG”**

Text, letter

Description automatically generated

4.Substuition Matrix:

We have given the two sequences.

THISSEQ

THATSEQ

From the given BLOSUM-62 matrix matrix, we can obtain these values and

(a)



(b) From the given PAM -250 matrix, we can obtain these values and



5. Multiple Sequence Alignment for extended dynamic programming.

-> A multiple sequence alignment is basically an alignment of more than two sequences.

-> A pair wise alignment tells you about the similarities of two sequences but a multiple

Sequence tells you about the similarity among the multiple sequences.

-> Local alignment is to identify small highly similar regions of two proteins or two genes

With minimum gaps.

-> Global alignment aims end to end alignment of two sequences, reflecting overall

Sequence variations. Below is the brief description of them.

Pair wise alignment

Local alignment Global alignment

Tools: EMBOSS Water Tools: EMBOSS Needle

Multiple sequence alignment

Local alignment Global alignment

Tools: MUSCLE, CLUSTALW, Tools: BLOCK MAKER

MAFFT, T-COFFEE

-> To extend the dynamic programming formula for multiple sequence alignment to three dimensions, we will introduce a new variable and a new dimension to the dynamic programming matrix.

-> Suppose we have three sequences A, B, and C of lengths p, q, and r, respectively. We will define a 3D matrix S of size (p +1) x (q +1) x (r +1),

where S [i][j][k] represents the score of the optimal alignment of the i first characters of sequence A, the j first characters of sequence B, and the k first characters of sequence C.

The dynamic programming formula for three dimensions is as follows:

S[i][j][k] = max(S[i-1][j-1][k-1] + Z(A[i], B[j], C[k]), # match/mismatch

S[i-1][j][k] + Z(A[i], GAP, GAP), # gap in sequence B and C

S[i][j-1][k] + Z(GAP, B[j], GAP), # gap in sequence A and C

S[i][j][k-1] + Z(GAP, GAP, C[k])) # gap in sequence A and B

where Z(A[i], B[j], C[k]) is the score of aligning the characters A[i], B[j], and C[k].

If the characters are the same, we assign a positive score to the match.

If they are different, we assign a negative score to the mismatch.

We also assign a negative score to the gaps.

The runtime of the dynamic programming algorithm for three-dimensional multiple sequence alignment is **O(pqr),** which is cubic in the length of the sequences.

So, the run time in this case **is O(p\*q\*r)**

And we can have **7 cases to compare.**

The number of cases we need to compare is the number of paths from the vertex (0,0,0) to the vertex (p,q,r) in a cubic grid of size (p+1) x (q+1) x (r+1).

Each path corresponds to a possible alignment of the three sequences, and there are (p+q+r) choose ‘ r ’ paths.

This is because we need to choose r steps out of a total of (p+q+r) steps, and the order in which we choose them does not matter.

Therefore, the number of cases we need to compare is (p+q+r) choose r.

*THANKING YOU*

*YOURS SINCERALY.*