Exercise 02
Due Data: 14-Sep-2022, 13:30 PM

SMITH-WATERMAN ALGORITHM (15%)

Bioinformatics is a field that uses techniques from computer science, statistics, and engineering to help study large amounts of biological data. Analysis of protein structure and sequence is very important in bioinformatics, especially for understanding how cells work, which makes it easier to make drugs for metabolic pathways. Protein sequence alignment is a method for figuring out the relationships between different protein structures by finding the similarities between their structures.

The Smith-Waterman algorithm uses local sequence alignment to find similar regions between two strings of nucleic acid or protein sequences. Rather than comparing the entire sequence, the Smith-Waterman algorithm compares segments of varying lengths and optimizes the similarity measure.

MOTIVATION

Genome projects on various organisms have generated massive amounts of sequence data for genes and proteins in recent years, necessitating computational analysis. Sequence alignment reveals the relationships between genes or proteins, allowing us to understand their homology and functionality better. The alignment of sequences can also reveal conserved domains and motifs.

ALGORITHM DESGIN

This algorithm aims to compare the structures of two amino acids and is proposed for use in local sequence alignment in bioinformatics. The first row and column are both initialized to a value of zero. Then, **Equation 1** is applied to each cell in the matrix. Set the Gap to -2. Score 1 when the two strings being compared are identical and score 0 otherwise. This procedure must be repeated until the entire matrix has been filled in. When this happens, the matrix's last cell incurs the dissimilarity cost. ***Figure 1*** shows the steps for algorithm written in pseudocode.

$$H_{i,j} = MAX \begin{cases} MAX(H_{i-1,j-1} + Score, 0) \\ MAX(H_{i-1,j} + Gap, Gap) \\ MAX(H_{i,j-1} + Gap, Gap) \end{cases}$$

Equation 1

Algorithm 1 Sequential Implementation of Smith-Waterman

```
1: procedure SW(Str1, Str2,N)
2:     Initial first row and first column from 1 to N
3:     for <Row = 0 to N -1> do
4:         for <Column = 0 to N -1> do
5:             if Str1[Row − 1] == Str2[Column − 1] then
6:                 Score = 0
7:             else
8:                 Score = 1
9:             end if
10:            H[Row][Column]= Calculate Distance using
    Equation (1)
11:        end for
12:    end for
13: end procedure
```

Figure 1: pseudocode of Smith-Waterman sequence alignment

Table 1: deoxyribonucleic acid structure

Nucleotide Name	**Code**
Adenine	A
Guanine	G
Cytosine	C
Thymine	T

Random DNA Sequence results

```
>random sequence 1 consisting of 10 bases.
GTCGATTTGA

>random sequence 2 consisting of 10 bases.
ACGAAAGAGG

>random sequence 3 consisting of 10 bases.
TTCTGTCTGA

>random sequence 4 consisting of 10 bases.
ACTGGCATTC

>random sequence 5 consisting of 10 bases.
CGTACCGTCG

>random sequence 6 consisting of 10 bases.
GGCCCAAGGA

>random sequence 7 consisting of 10 bases.
CAATAGTCCG

>random sequence 8 consisting of 10 bases.
TTACAGGTGT

>random sequence 9 consisting of 10 bases.
ACTTGTCGTC

>random sequence 10 consisting of 10 bases.
CTACGCCTAC
```

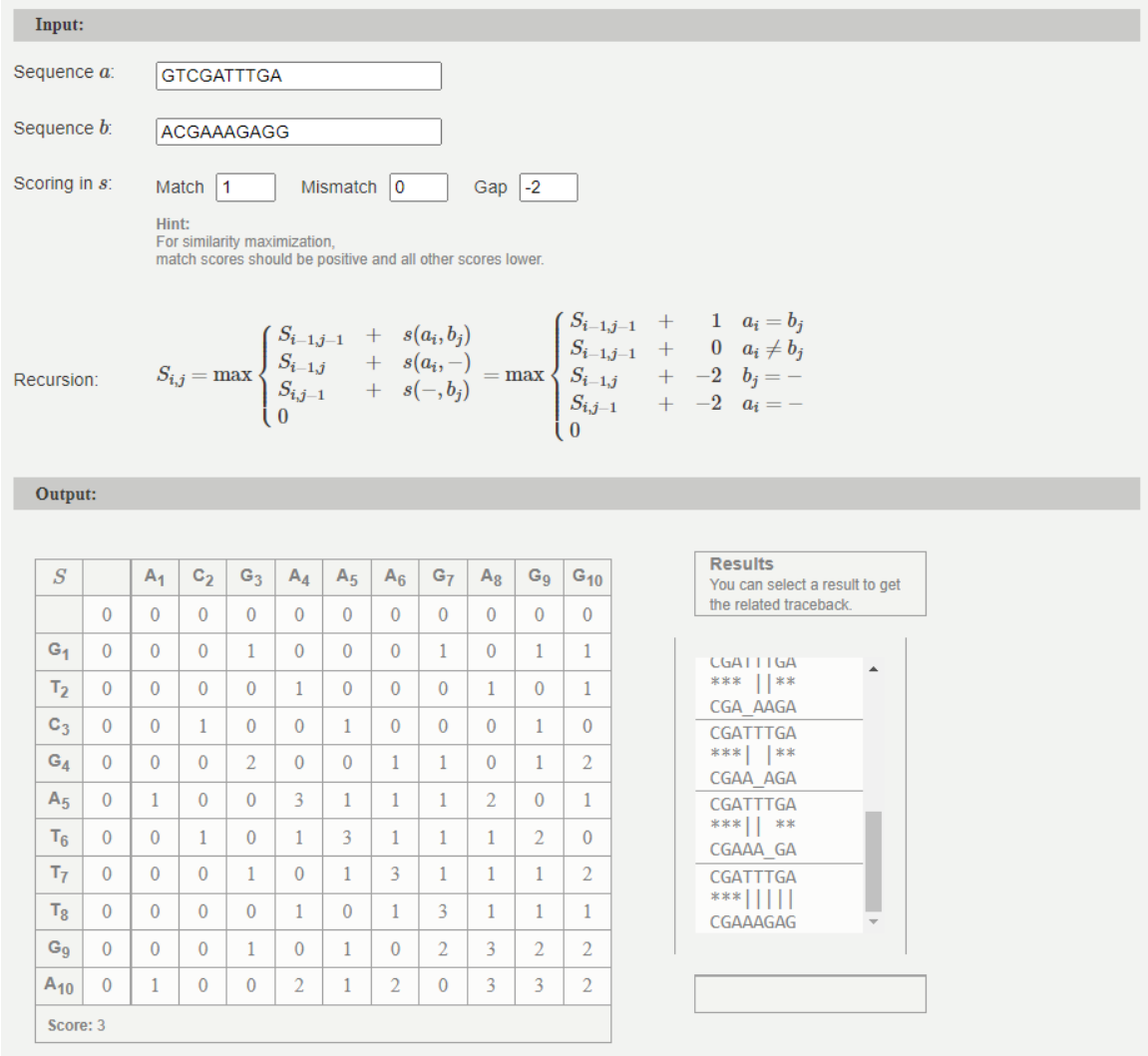
Figure 2 displays an example to calculate the similarity between `sequence 1` and `sequence 2` using online tool1. You have to implement Java code for this algorithm to measure the similarity between any DNA sequences. You can use the previous 10 sequences as test cases to your implementation.

Grading schema:

- Read from text file (5 Pts): **Note: you need to handle I/O exceptions**.
- Java implementation for Smith-Waterman algorithm (5 Pts).
- Presentation (5 Pts)

[1] https://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Smith-Waterman

Exercise 02
Due Data: 14-Sep-2022, 13:30 PM



S		A_1	C_2	G_3	A_4	A_5	A_6	G_7	A_8	G_9	G_{10}
	0	0	0	0	0	0	0	0	0	0	0
G_1	0	0	0	1	0	0	0	1	0	1	1
T_2	0	0	0	0	1	0	0	0	1	0	1
C_3	0	0	1	0	0	1	0	0	0	1	0
G_4	0	0	0	2	0	0	1	1	0	1	2
A_5	0	1	0	0	3	1	1	1	2	0	1
T_6	0	0	1	0	1	3	1	1	1	2	0
T_7	0	0	0	1	0	1	3	1	1	1	2
T_8	0	0	0	0	1	0	1	3	1	1	1
G_9	0	0	0	1	0	1	0	2	3	2	2
A_{10}	0	1	0	0	2	1	2	0	3	3	2

Figure 2: Shows an example for Smith-Waterman Algorithm.