GIT Department of Computer Engineering

CSE 654 / 484 Fall 2022

Homework 1 # Report

Okan Torun

1801042662

Subject of assignment:

100 different text documents were created with the contents taken from 20 different meb books. Later on, common sentences were inserted into some documents. By comparing 2 different documents from these documents, common lines should be found.

What is Smith-Waterman Algorithm ?

The Smith–Waterman algorithm performs local sequence alignment; that is, for determining similar regions between two strings of nucleic acid sequences or protein sequences. Instead of looking at the entire sequence, the Smith–Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure.

How Do I Solve the Problem ?

- First of all, I extracted the dynamic programming matrix result using the Smith-Waterman Algorithm for the rows I examined.

- When generating these results, I used 2 scores in each match(substitution) operation, -1 score in the mismatch operation, and -1 score in the delete(gap) and insert(gap) operations.

- Then I filled the matrix I created according to the dimensions of the two strings I have with the scores I determined and the operations I made.

- After creating the matrix, I created two different alignments by comparing the similarities of the two lines with the **Traceback** function.

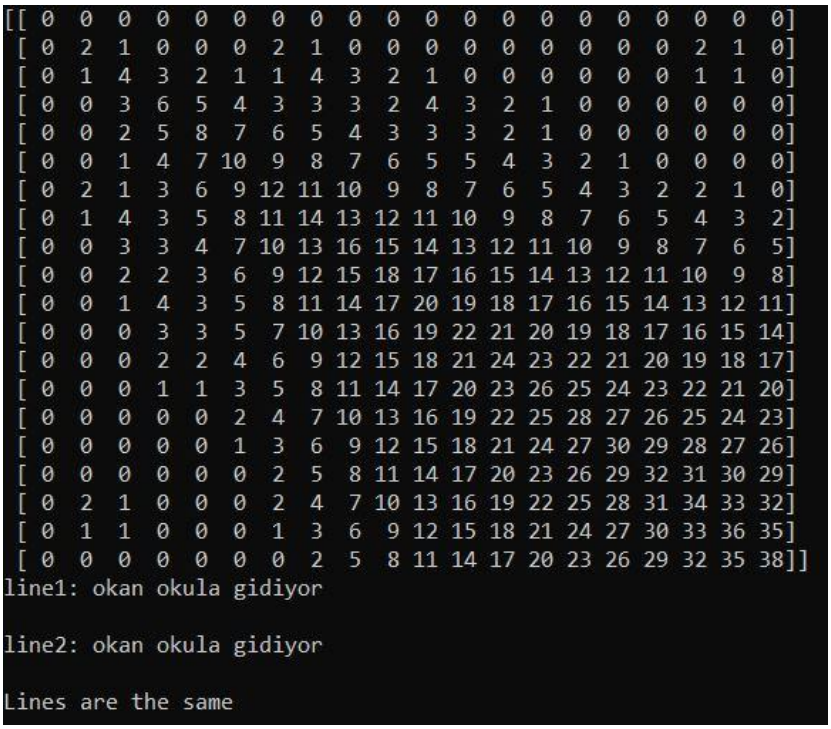
- The main purpose of the **Smith-Waterman Algorithm** is to find **the largest score** in the matrix and follow the largest scores to the starting point. In this way, the **best local alignment** is found.

- While returning from this score, I create the alignment 1 and alignment 2 strings. If these strings are equal, it means I found the common lines.

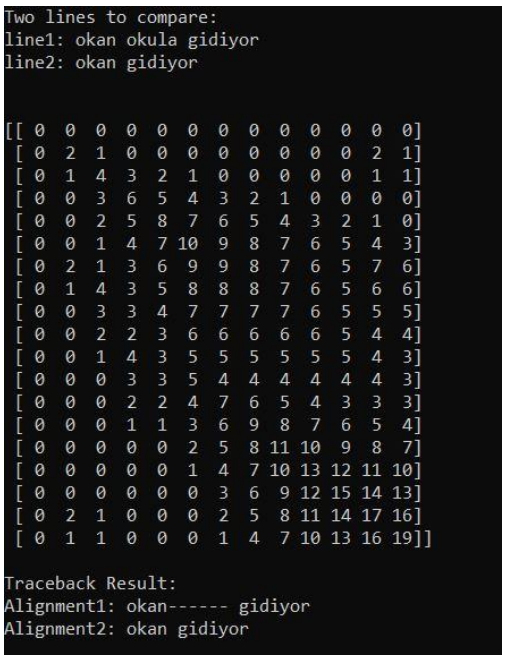
- The end of the traceback process ends when all the paths it can take are 0.

Test Cases :

- The result of the common lines I entered in the documents I created



- Printed result if rows are not common



```
[[ 0  0  0  0  0  0  0  0  0  0]
 [ 0  2  1  0  0  0  0  0  0  0]
 [ 0  1  4  3  2  1  0  0  0  0]
 [ 0  0  3  6  5  4  3  2  1  0]
 [ 0  0  2  5  8  7  6  5  4  3]
 [ 0  0  1  4  7 10  9  8  7  6]
 [ 0  0  0  3  6  9 12 11 10  9]
 [ 0  0  0  2  5  8 11 14 13 12]
 [ 0  0  0  1  4  7 10 13 16 15]
 [ 0  0  0  0  3  6  9 12 15 18]]
line1: Glikojen

line2: Glikojen

Lines are the same
```

```
Two lines to compare:
line1: ATCAT
line2: ATTATC


[[0 0 0 0 0 0 0]
 [0 2 1 0 2 1 0]
 [0 1 4 3 2 4 3]
 [0 0 3 3 2 3 6]
 [0 2 2 2 5 4 5]
 [0 1 4 4 4 7 6]]

Traceback Result:
Alignment1: ATCAT
Alignment2: AT-AT
```

Note : Since there are many lines of 1-2 characters without meaning in the documents and to get more meaningful results, if the line length is greater than 5, I put it in the evaluation.