Istanbul Technical University
Faculty of Computer and Informatics

BLG336E – Analysis of Algorithms II, Spring 2021
Homework 3
Local Sequence Alignment with Smith-Waterman

Handed out: May 04th 2021 **Due:** May 19th, 2021 23.59
Res. Asst.: Uğur Önal

Notes:

- Please submit your homework only through Ninova. Late submissions will not be accepted.
- Please do not forget to write your name and student ID on the top of each document you upload.
- You should write your code in C++ language and try to follow an object-oriented methodology with well-chosen variables, methods, and class names and comments where necessary.
- You need to submit your source code, output file and report in a ZIP file.
- Your code should compile on Linux using g++. You can test your code through ITU's Linux Server (you can access it through SSH).
- Please compile your sour code with these flags, do NOT remove or change the flags.

```
g++ -std=c++11 -Wall -Werror <yourStudentNumber>.cpp -o <yourStudentNumber>
```

- All warnings will be counted as errors. So fix the errors untill you get none.
- Remember to compile on SSH once you are done with implementation.
- Because your codes will be processed with an automated tool (Calico), make sure that your output format matches the given sample output.
- To test your program yourself with the given test cases, please check out the tutorial for Calico provided to you on Ninova.
- In your source code files, use new line character as: \n
  NOT as \r\n
- Carriage return '\r' is only used in test case files.
- You can use STL containers.
- You may discuss the problem addressed in the homework at an abstract level with your classmates, but you should not share or copy code from your classmates or any web sources. You should submit your individual homework. In case of detection of an inappropriate exchange of information, disciplinary actions will be taken.
- If you have any questions, please ask in **Ninova message board** or contact **Uğur Önal** via **onalug@itu.edu.tr**.

Overview

The Smith–Waterman algorithm is a dynamic programming algorithm used to find an optimum local alignment of given two sequences [1]. The main application area of the Smith–Waterman algorithm is biological domain, i.e., the alignment of molecular sequences with the motivation of better understanding the homology and the functionality of the biological sequences.

Another popular application area of the algorithm is in the natural language processing field while calculating the similarities between natural language texts [2]. For instance, some spell checker tools compare a misspelled word with a list of well-typed words, and computes a similarity value (also called as distance value) between the misspelled word and each of the well-typed word (correction candidate). Finally, spell checker tool suggests the well-typed word with the smallest distance (the most similar one) to the misspelled word as the most likely correction candidate.

In this homework, you will apply the Smith–Waterman algorithm to find the local alignments in natural language sequences provided to you.

A local sequence alignment is a scheme of writing the most common sequence between two different sequences. Similar sequences may have different length, though, which is generally explained through insertions or deletions in sequences. Thus, a letter or a stretch of letters may be paired up with dashes in the other sequence to signify such an insertion or deletion. Since an insertion in one sequence can always be seen as a deletion in the other one frequently uses the term "indel". For an example sequence alignment of letters see the given below.

```
INFORM→→
→→FORMAL
```

The most common sequence between these sequences is:

```
→→FORM→→
```

Implementation

In such a simple evolutionarily motivated scheme, an alignment mediates the definition of a distance for two sequences. One generally assigns a positive number to a match, some negative number to a mismatch and a larger negative number to an indel. You can use your own numbers to assign while keeping the order. If a value becomes negative, then it is changed into 0. By adding these values along an alignment, one obtains a score for this alignment. The most common sequences found by moving from highest value to zero while moving between matches. Luckily, using dynamic programming this minimization can be affected without explicitly enumerating all possible alignment of two sequences.
Dynamic programming algorithms are recursive algorithms modified to store intermediate results, which improves efficiency for certain problems. The Smith-Waterman algorithm uses a dynamic programming approach to find the optimal local alignment of two sequences — a and b. The alignment algorithm is based on finding the elements of a matrix S where the element Si,j is the optimal score for aligning the sequence (a1, a2, ..., ai) with (b1, b2, ..., bj). Two similar sequences receive a high score, two dissimilar sqeuences receive a low score.
The higher the score of a path through the matrix, the better the alignment. The matrix S is found by progressively finding the matrix elements, starting at S1,1 and proceeding in the directions of increasing i and j. Each element is set according to:

$$S_{i,j} = \max \left[\begin{array}{l} S_{i-1,j-1} + s_{i,j} \\ S_{i-1,j} + g \\ S_{i,j-1} + g \\ 0 \end{array} \right.$$

where $s_{i,j}$ is the similarity score of comparing element a_i to element b_j and g is the penalty for a single gap (see also the Figure 1). To understand how to fill the matrix to compute the dynamic programming algorithm you can see the step-by-step example given here.

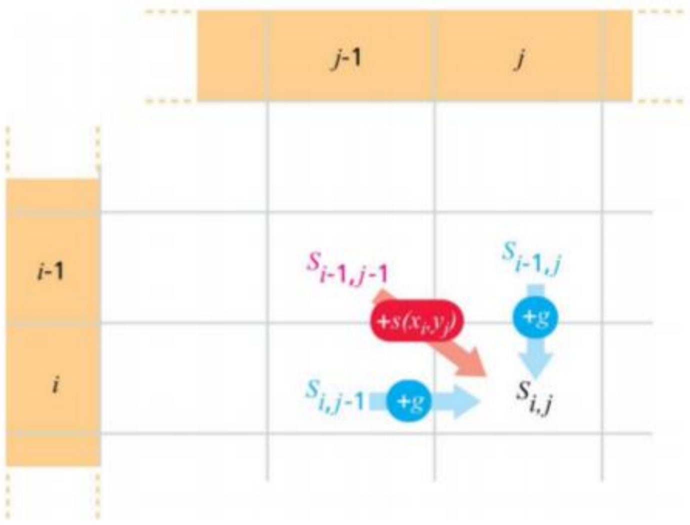

Figure 1: How to calculate $S_{i,j}$ using subsolutions

For this homework, you need to calculate local sequences for every string pairs in a group of strings given in a TXT file. After the calculation, you need to print the most common sequences between each pair in the format below:

1) Order a pair in alphabetical order between each other.
2) Order all pairs in alphabetical order from top to bottom.
3) If a pair have more than one most common sequences, order them in alphabetical order.

For example:

The given group of strings:

```
information
funimatio
malfunction
```

The output:

```
funimatio – information
Score: 5 Sequence(s): “matio”
funimatio – malfunction
Score: 3 Sequence(s): “fun” “tio”
information – malfunction
Score: 4 Sequence(s): “tion”
```

Report

Please prepare a file that reports and discuss the following items:

1) Report your problem formulation:
    a) Write your pseudo-code.
    b) Show the complexity of your algorithm on the pseudo-code.
2) Analyze and compare the algorithm results with assessing all possible alignments one by one in terms of:
    a) the calculations made,
    b) the maximum number of calculation results kept in the memory,
    c) the running time.

References

1. Smith, Temple F., and Michael S. Waterman. "Identification of common molecular subsequences." *Journal of molecular biology* 147.1 (1981): 195-197.
2. Katrenko, Sophia, and Pieter Adriaans. "A local alignment kernel in the context of NLP." *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. 2008.