Programming Lab 2

Summer 2018\2019

Programming Project Group 05: Computational Methods Of Sequence Alignment

Rana Aldisi 3113587

Mohamed Mahgoub 3118671

Rahma Maghraby 3119594

Table of Content

Introduction

Sequence alignment is an approach in biological studies that compares sequences of DNA, RNA or protein to identify regions of similarity between them. This method is usually used to discover structural, functional and evolutionary information; since similar sequences may have similar secondary and tertiary structure, similar function and a common ancestry. A sequence alignment can be made between a known sequence, reference sequence, and unknown sequence, query sequence, or between two unknown sequences [1].

Types of Sequence Alignment

There are two main types of sequence alignment, **Global Alignment** and **Local Alignment**. Global Alignment attempts to match the residues of two sequences across their entire length. The main goal of this type is to identify conserved regions between the sequences and it is usually used to compare two genes from different species with same function or to compare two proteins with similar function. Local Alignment attempts to match subsections of the sequences, eliminating regions that are highly variant. The main goal of local alignment is to see whether a substring in one sequence aligns well with a substring in the other, thus, it is used to searching for local similarities in large sequences (e.g., newly sequenced genomes) and look for conserved domains or motifs in two proteins. There are many other types of alignments that were derived from those two main types such as affine gap alignment, overlap alignment, fitting alignment and semi-global alignment [1].

Scoring Matrices

When aligning a letter in the sequence with a letter in the other sequence, a score is given in case it is a match (same letter) or mismatch. This score can be fixed or predetermined depending on the pair aligned. Since sequence alignment is usually done

on protein sequences, a 20 x 20 scoring matrices are usually created to statically give a score to each possible pair of amino acid residues. Many scoring matrices have been created for sequences alignment for different purposes. The two most popular matrices used are BLOSUM and PAM [2].
BLOSUM (**Blo**cks **Su**bstitution **M**atrix) matrices are commonly used to score evolutionary divergence between protein sequences. It is good for detecting highly conserved subsections of alignments. The numbers next to the matrix name denote smaller evolutionary distance (i.e higher sequence similarity)) [2].
PAM (**P**ercent **A**ccepted **M**utation) matrices are usually used score alignments between protein sequences that are closely related. It's good at detecting mutations in alignments which is highly significant for alignments like global alignment. The numbers next to the matrix name (e.g PAM30) denote the evolutionary distances; the higher the number, the greater the evolutionary distance [2].

Background

The alignment between two short sequences can be easily found manually. However, as the sequences get longer, it gets much more difficult for someone to calculate the optimal alignment, which is why many algorithms and computational methods have been made to solve such problems. In this section, we will discuss few algorithms that have been used for this project.

Needleman-Wunsch algorithm

Needleman-Wunsch algorithm was developed in 1970 by Saul Needleman and Christian Wunsch and was one of the first biological sequence alignment applications that uses dynamic programming [3]. The algorithm computes the optimal global alignment between two sequences S[1..n] and T[1..m] where V(n,m) is the score of the optimal alignment.

To compute this optimal alignment score the matrix V(1..n, 1..m) is first initialized as follows:

$V_{0,0} = 0$

$V_{0,j} = j * gap\ (to\ denote\ insertion)$

$V_{i,0} = i * gap\ (to\ denote\ deletion)$

Then the matrix is filled row by row using a recursive equation:

$V_{i,j} = max(V_{i-1,j-1} + S(A_i, B_j),\ V_{i-1,j} + gap,\ V_{i,j-1} + gap)$

In which $S(A_i, B_j)$ denotes a score of the match or mismatch of letters in the sequences and the gap is the penalty given in case of a gap aligned with a letter in the sequence.

The score of the optimal alignment would be the last column in the last row. To be able to recover the optimal alignment, a direction can be indicated during the recursive procedure, in which $V_{i-1,j-1}$ equals to diagonal direction, $V_{i-1,j} + gap$ denotes a vertical direction and $V_{i,j-1} + gap$ equals to horizontal direction [4].

The complexity of this algorithm is O(nm) for two sequences S(1..n) and T(1..m), which means that it can only be efficiently used on relatively short sequences.

Smith-Waterman algorithm

Smith-Waterman algorithm is a variant of Needleman-Wunsch algorithm that was proposed in 1981 by Temple Smith and Michael Waterman to perform local sequence alignment [5]. In this algorithm, we consider two sequences S[1..n] and T[1..m] in which the optimal local alignment V(i,j) is the maximum global alignment of the substrings of S and substrings of T, where $1 \le i \le n$ and $1 \le j \le m$. To compute this optimal alignment we initialize the matrix and follows:

$V_{0,j} = 0$

$V_{i,0} = 0$

Then we fill the matrix recursively row by row using the following equation:

$V_{i,j} = max(0, V_{i-1,j-1} + S(A_i, B_j),\ V_{i-1,j} + gap,\ V_{i,j-1} + gap)$

The equation is the same as the Needleman-Wunsch equation except for the 0 which means that it will align empty strings. The optimal local alignment score will be the maximum score in the V matrix. The aligned sequence can then be retrieved by backtracking from the optimal alignment score [4].
Similar to Needleman-Wunsch algorithm, the complexity is O(nm) for two sequences S(1..n) and T(1..m).

Project overview

The main objective of this project is to create an alignment between two sequences. The sequences can be aligned using global, local, affine gap, overlap or fitting alignment and the results contain the optimal alignment score as well as the alignment of the sequences. The program can be accessed through a command-line interface, a web interface or through a REST API.

Project Design

The project is divided into four sections. Fasta-tools and sequencelib are the main libraries containing all the methods needed to read and make the alignment. Cl-interface and web-interface contain the methods needed for the user to input parameters and present the results, and they depend on sequencelib library to get the alignment results.

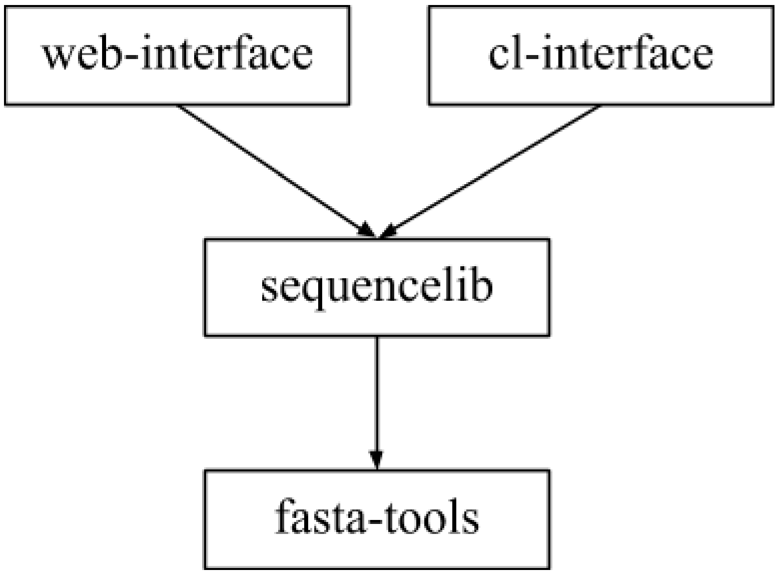


Figure 1: Outline of the project's flow.

fasta-tools library

This library was created to handle and parse fasta format files to extract the sequences in them. Fasta format files that represent DNA or protein sequences and they usually starts with a '>' symbol followed by a description then the sequence. This library contains two classes Sequence and Sequences and it is a general library that can be used for purposes other than sequence alignment.

- Sequence
    - The Sequence object gets the header of the sequence, the sequence and its length.
- Sequences
    - The Sequences class sets a list of Sequence objects by reading a file, creating a Sequence object the saving it in Sequences Object. It only accepts a fasta file format.

sequencelib library

This library implements all the alignment types used in both User Interface (UI) applications. It depends on the fasta-tools library to handle the input files and get the sequences from the user. Each alignment has its own class which inherits some methods from their parent alignment class. The library also contains a ScoringMatrices class which gets the different commonly known scoring, or substitution matrices used in sequence alignment.

- Alignment
    - A parent (super) class that gets the sequences from the fasta-tools library, gets the aligned sequences after an alignment, gets the optimal score for the alignment, the matrices used and performs common methods for its children classes. It has two methods:
        - getMax, which retrieves the indices of the maximum score in a matrix
        - traceback, which backtracks from a matrix depending on the direction of the optimal alignment score. This follows a simple logic where a matrix is made that gives directions (up, left and match) a numerical value (2, 1, 0), and depending on the maximum score in the current position the value is saved in the traceback matrix.
- GlobalAlignment (Child)
    - This class implements the Needleman-Wunsch algorithm.
- LocalAlignment (Child)
    - This class implements the Smith-Waterman algorithm.

- OverlapAlignment (Child)
  - This class implements a modified version of Needleman-Wunsch algorithm in which the suffix of one sequence is aligned with the prefix of the other.
  - Initialization is the same as Smith-Waterman.
- FittingAlignment (Child)
  - This class also implements a modified version of Needleman-Wunsch algorithm. In this case, a very long sequence is aligned to a much shorter one, so only a substring of the long sequence is aligned with the second sequence.
  - Initialization is the same as Smith-Waterman.
- SemiglobalAlignment (Child)
  - This class is almost identical to the Overlap alignment, in which it ignores the gaps at the beginnings and ends of the sequences.
  - Initialization is the same as Smith-Waterman.
- AffineGapAlignment (Child)
  - This class is similar to global alignment; however, instead of having a fixed gap penalty, affine gap alignment gives a high penalty to the opening of a gap, but the extension of the gap doesn't cost as much.
  - Initialization is similar to global alignment.
  - Three matrices are created to store the scores. One for the score in case there is an extending gap in the first sequence, one in case there is an extending gap in the second sequence and one in case it's a match/mismatch situation.
- ScoringMatrices
  - This class implements the most popular scoring matrices for sequence alignment and can get the matrix requested by the user to use in the

alignment. It contains the following matrices: PAM(30, 70, 120 and 250) and BLOSUM(50, 62 and 80).

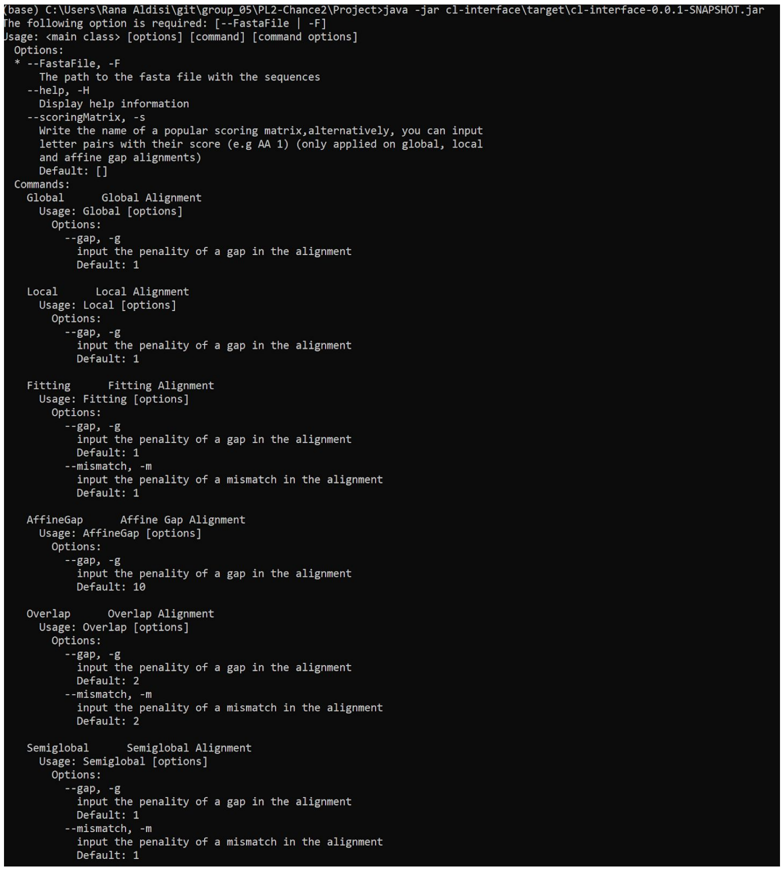
- ColoredScores
    - This class is mainly used in the web interface. It contains two values, a score and a boolean. The idea is to create a matrix of this object, in which the alignment classes use to save the score in a specific position during alignment calculation, then during the traceback, the objects in the positions of the direction of the optimal alignment will get a 'true' boolean. This is used later in the web interface to color the path of the alignment.

cl-interface

Command line interface (CLI) is a user interface that interacts with the program using input text. The CLI for this project was implemented using Jcommander [6], which is a simple java framework used to parse command line parameters. Using Jcommander, one can build a set of commands that have a set of parameters, which can be marked as required or optional, and can have a default value or none. And then using the commands a user can input the parameters for the specific command to run the program.

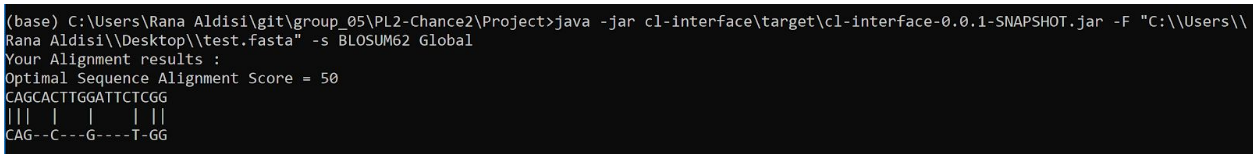
Implementation

In this CLI, the main command requires that the user inputs a Fasta file {--FastaFile, -F}, optionally the user can also choose a scoring matrix for the alignment or create their own scoring matrix {--scoringMatrix, -s}. Then the alignment type is input as a second command. Each alignment has its own command and its own parameters and depending on the type of alignment chosen, the user can input their own parameters or keep it as default. After inputting the correct set of commands, the CLI uses sequecenlib library to get the optimal alignment score and print it into the command line along with the aligned sequences. Figure 2 shows the help information for the command line and Figure 3 shows an example of the output.



```
(base) C:\Users\Rana Aldisi\git\group_05\PL2-Chance2\Project>java -jar cl-interface\target\cl-interface-0.0.1-SNAPSHOT.jar
The following option is required: [--FastaFile | -F]
Usage: <main class> [options] [command] [command options]
  Options:
  * --FastaFile, -F
      The path to the fasta file with the sequences
    --help, -H
      Display help information
    --scoringMatrix, -s
      Write the name of a popular scoring matrix,alternatively, you can input
      letter pairs with their score (e.g AA 1) (only applied on global, local
      and affine gap alignments)
      Default: []
  Commands:
    Global      Global Alignment
      Usage: Global [options]
        Options:
          --gap, -g
            input the penality of a gap in the alignment
            Default: 1

    Local      Local Alignment
      Usage: Local [options]
        Options:
          --gap, -g
            input the penality of a gap in the alignment
            Default: 1

    Fitting      Fitting Alignment
      Usage: Fitting [options]
        Options:
          --gap, -g
            input the penality of a gap in the alignment
            Default: 1
          --mismatch, -m
            input the penality of a mismatch in the alignment
            Default: 1

    AffineGap      Affine Gap Alignment
      Usage: AffineGap [options]
        Options:
          --gap, -g
            input the penality of a gap in the alignment
            Default: 10

    Overlap      Overlap Alignment
      Usage: Overlap [options]
        Options:
          --gap, -g
            input the penality of a gap in the alignment
            Default: 2
          --mismatch, -m
            input the penality of a mismatch in the alignment
            Default: 2

    Semiglobal      Semiglobal Alignment
      Usage: Semiglobal [options]
        Options:
          --gap, -g
            input the penality of a gap in the alignment
            Default: 1
          --mismatch, -m
            input the penality of a mismatch in the alignment
            Default: 1
```

Figure 2: the commands needed for the command line interface to run an alignment.



```
(base) C:\Users\Rana Aldisi\git\group_05\PL2-Chance2\Project>java -jar cl-interface\target\cl-interface-0.0.1-SNAPSHOT.jar -F "C:\\Users\\
Rana Aldisi\\Desktop\\test.fasta" -s BLOSUM62 Global
Your Alignment results :
Optimal Sequence Alignment Score = 50
CAGCACTTGGATTCTCGG
|||  |   |    | ||
CAG--C---G----T-GG
```

Figure 3: an example of a successful command line interface run.

web-interface

Web interface uses a web server to interact with the user via web browser. Spring [7] is a Java framework that is used for building web applications and services. Spring boot, which is a part of Spring framework, is a good framework for beginners in web services and design since it is easy to configure and run, and it provides a lot of tutorials that are simple and easy to follow. To design a web page, HTML (Hyper Text Markup Language) needs to be used. This language has elements that are represented by tags, and these elements code how to display the content of the web page. HTML uses a CSS (Cascading Style Sheets) syntax to describe the layout of the web page and how the elements need to be displayed on it. To be able to use HTML\CSS in the web application, Spring integrates Thymeleaf[8], which is a Java template engine that can process HTML, XML, JavaScript, CSS and plain text. Bulma[9] is an open source CSS Framework that helps in creating more elegant stylesheets for a web page in a simple manner. Bulma is a great resource for designing web pages, especially for beginners, since it has a lot of documentation and tutorials.

Implementation

For the basic web application, Spring boot was used to create a controller that gets the alignment parameters from the user along with an uploaded file. The parameters are saved into a class (Align) to be retrieved for the alignment process and the file is checked for its validity, if the file is not valid the user will be transported to an Error page, if not, the file information is copied to a temporary file to be used then deleted after the alignment process is done. Once the parameters are all set the type of the alignment selected is identified and the sequencelib library is used to run the alignment.

When the program is run, the user can go to the home page through (http://localhost:8080) which contains a welcome phrase for the user, as well as hyperlinks that can lead the user to an information page (http://localhost:8080/info) or the sequence alignment page (http://localhost:8080/align). The sequence alignment page contains a form, creating using Thymeleaf, for selecting and filling parameters for the alignment, once the form is filled and submitted, the web services gives the results in a new page, which contains the alignment score, the aligned sequences and the calculated score matrix.

Bulma was used to create a simple yet elegant design for the web pages, in the info page, expandable cards were used to hide the information until expanded by user. A navigation bar was also added to all pages, except the home page, to make it easier to navigate through the different pages. All pages have a header for clarification and horizontal scrolling was added when needed, figure 4 shows the different designs of pages of the web application.

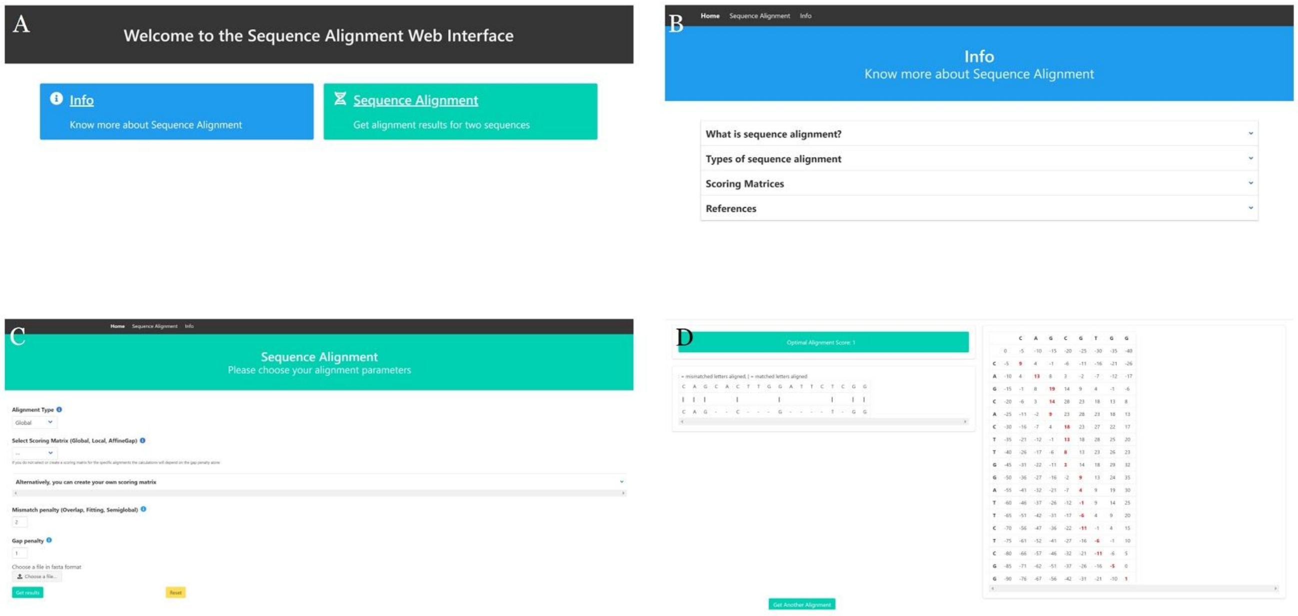
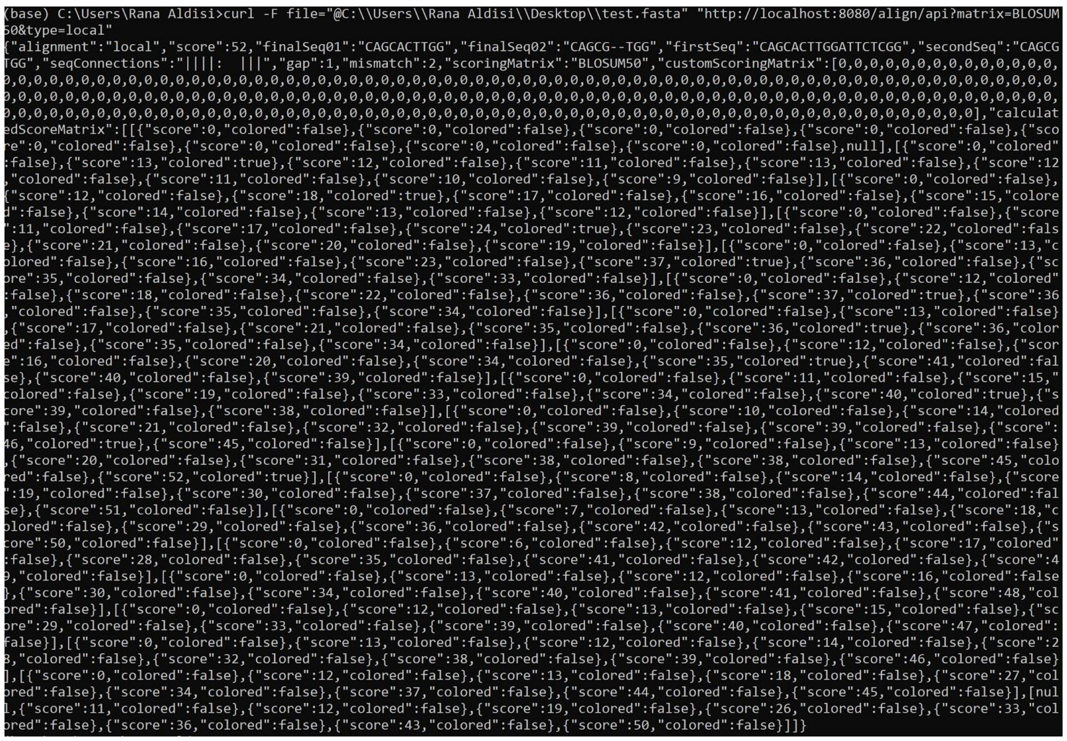

A
Welcome to the Sequence Alignment Web Interface
Info
Know more about Sequence Alignment
Sequence Alignment
Get alignment results for two sequences
B
Home Sequence Alignment Info
Info
Know more about Sequence Alignment
What is sequence alignment?
Types of sequence alignment
Scoring Matrices
References
C
Home Sequence Alignment Info
Sequence Alignment
Please choose your alignment parameters
Alignment Type
Global
Select Scoring Matrix (Global, Local, AffineGap)
Alternatively, you can create your own scoring matrix
Mismatch penalty (Overlap, Fitting, Semiglobal)
Gap penalty
Choose a file in fasta format
Choose a file...
Get results
Reset
D
Optimal Alignment Score: 1
Get Another Alignment

Figure 4: The design of the web pages. A) The home page. B) The information page. C) The sequence alignment form page. D) The alignment results page.

RESTful API

As the name suggests, RESTful API uses the software architectural style REST (Representational State Transfer), which allows the user to access the resources in the web server using HTTP requests (like GET or POST) and receive the results as a response [10].

For this project, A REST controller was created which will take an uploaded file {-f file=@path} from the user and return the results in json format. The user can further choose alignment parameters by adding them to the http request. Then the controller uses the same methods used by the web interface above to get the alignment and saves the results in the Align class, which is then submitted as a response body to the user. Figure 5 shows an example of a request and the resulted output.



```
(base) C:\Users\Rana Aldisi>curl -F file="@C:\\Users\\Rana Aldisi\\Desktop\\test.fasta" "http://localhost:8080/align/api?matrix=BLOSUM
50&type=local"
{"alignment":"local","score":52,"finalSeq01":"CAGCACTTGG","finalSeq02":"CAGCG--TGG","firstSeq":"CAGCACTTGGATTCTCGG","secondSeq":"CAGCG
TGG","seqConnections":"||||:  |||","gap":1,"mismatch":2,"scoringMatrix":"BLOSUM50","customScoringMatrix":[0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"calculat
edScoreMatrix":[[{"score":0,"colored":false},{"score":0,"colored":false},{"score":0,"colored":false},{"score":0,"colored":false},{"sco
re":0,"colored":false},{"score":0,"colored":false},{"score":0,"colored":false},{"score":0,"colored":false},null],[{"score":0,"colored"
:false},{"score":13,"colored":true},{"score":12,"colored":false},{"score":11,"colored":false},{"score":13,"colored":false},{"score":12
,"colored":false},{"score":11,"colored":false},{"score":10,"colored":false},{"score":9,"colored":false}],[{"score":0,"colored":false},
{"score":12,"colored":false},{"score":18,"colored":true},{"score":17,"colored":false},{"score":16,"colored":false},{"score":15,"colore
d":false},{"score":14,"colored":false},{"score":13,"colored":false},{"score":12,"colored":false}],[{"score":0,"colored":false},{"score
":11,"colored":false},{"score":17,"colored":false},{"score":24,"colored":true},{"score":23,"colored":false},{"score":22,"colored":fals
e},{"score":21,"colored":false},{"score":20,"colored":false},{"score":19,"colored":false}],[{"score":0,"colored":false},{"score":13,"c
olored":false},{"score":16,"colored":false},{"score":23,"colored":false},{"score":37,"colored":true},{"score":36,"colored":false},{"sc
ore":35,"colored":false},{"score":34,"colored":false},{"score":33,"colored":false}],[{"score":0,"colored":false},{"score":12,"colored"
:false},{"score":18,"colored":false},{"score":22,"colored":false},{"score":36,"colored":false},{"score":37,"colored":true},{"score":36
,"colored":false},{"score":35,"colored":false},{"score":34,"colored":false}],[{"score":0,"colored":false},{"score":13,"colored":false}
,{"score":17,"colored":false},{"score":21,"colored":false},{"score":35,"colored":false},{"score":36,"colored":true},{"score":36,"color
ed":false},{"score":35,"colored":false},{"score":34,"colored":false}],[{"score":0,"colored":false},{"score":12,"colored":false},{"scor
e":16,"colored":false},{"score":20,"colored":false},{"score":34,"colored":false},{"score":35,"colored":true},{"score":41,"colored":fal
se},{"score":40,"colored":false},{"score":39,"colored":false}],[{"score":0,"colored":false},{"score":11,"colored":false},{"score":15,"
colored":false},{"score":19,"colored":false},{"score":33,"colored":false},{"score":34,"colored":false},{"score":40,"colored":true},{"s
core":39,"colored":false},{"score":38,"colored":false}],[{"score":0,"colored":false},{"score":10,"colored":false},{"score":14,"colored
":false},{"score":21,"colored":false},{"score":32,"colored":false},{"score":39,"colored":false},{"score":39,"colored":false},{"score":
46,"colored":true},{"score":45,"colored":false}],[{"score":0,"colored":false},{"score":9,"colored":false},{"score":13,"colored":false}
,{"score":20,"colored":false},{"score":31,"colored":false},{"score":38,"colored":false},{"score":38,"colored":false},{"score":45,"colo
red":false},{"score":52,"colored":true}],[{"score":0,"colored":false},{"score":8,"colored":false},{"score":14,"colored":false},{"score
":19,"colored":false},{"score":30,"colored":false},{"score":37,"colored":false},{"score":38,"colored":false},{"score":44,"colored":fal
se},{"score":51,"colored":false}],[{"score":0,"colored":false},{"score":7,"colored":false},{"score":13,"colored":false},{"score":18,"c
olored":false},{"score":29,"colored":false},{"score":36,"colored":false},{"score":42,"colored":false},{"score":43,"colored":false},{"s
core":50,"colored":false}],[{"score":0,"colored":false},{"score":6,"colored":false},{"score":12,"colored":false},{"score":17,"colored"
:false},{"score":28,"colored":false},{"score":35,"colored":false},{"score":41,"colored":false},{"score":42,"colored":false},{"score":4
9,"colored":false}],[{"score":0,"colored":false},{"score":13,"colored":false},{"score":12,"colored":false},{"score":16,"colored":false
},{"score":30,"colored":false},{"score":34,"colored":false},{"score":40,"colored":false},{"score":41,"colored":false},{"score":48,"col
ored":false}],[{"score":0,"colored":false},{"score":12,"colored":false},{"score":13,"colored":false},{"score":15,"colored":false},{"sc
ore":29,"colored":false},{"score":33,"colored":false},{"score":39,"colored":false},{"score":40,"colored":false},{"score":47,"colored":
false}],[{"score":0,"colored":false},{"score":13,"colored":false},{"score":12,"colored":false},{"score":14,"colored":false},{"score":2
8,"colored":false},{"score":32,"colored":false},{"score":38,"colored":false},{"score":39,"colored":false},{"score":46,"colored":false}
],[{"score":0,"colored":false},{"score":12,"colored":false},{"score":13,"colored":false},{"score":18,"colored":false},{"score":27,"col
ored":false},{"score":34,"colored":false},{"score":37,"colored":false},{"score":44,"colored":false},{"score":45,"colored":false}],[nul
l,{"score":11,"colored":false},{"score":12,"colored":false},{"score":19,"colored":false},{"score":26,"colored":false},{"score":33,"col
ored":false},{"score":36,"colored":false},{"score":43,"colored":false},{"score":50,"colored":false}]]}
```

Figure 5: an example of a request on the REST API and the resulted output.

Conclusion and Future Work

This project was created to make a simple pairwise sequence alignment program using different sequence alignment methods. It has a command line interface, a web interface and a REST API for user application. However, this program cannot take really long sequences, so improving the computational complexity of the algorithms or using a java library might be a better option for improving the program. Also, the web interface could use some beautification, like showing and removing specific options depending on the user's alignment choice, as well as improving the input of the custom matrix for a better visualization.

References

1. Mount, D. W. *Bioinformatics: sequence and genome analysis*. (Cold Spring Harbor Laboratory Press, 2001).
2. Mount, D. W. Comparison of the PAM and BLOSUM Amino Acid Substitution Matrices. *CSH Protoc* **2008**, pdb.ip59 (2008).
3. Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **48**, 443–453 (1970).
4. Sung, W.-K. *Algorithms in Bioinformatics: A Practical Introduction*. (CRC Press, 2009).
5. Smith, T. F. & Waterman, M. S. Identification of common molecular subsequences. *Journal of Molecular Biology* **147**, 195–197 (1981).
6. JCommander. Available at: http://jcommander.org/. (Accessed: 11th July 2019)
7. spring.io. Available at: https://spring.io/. (Accessed: 11th July 2019)
8. Thymeleaf. Available at: https://www.thymeleaf.org/. (Accessed: 11th July 2019)
9. Bulma: Free, open source, & modern CSS framework based on Flexbox. Available at: https://bulma.io/. (Accessed: 11th July 2019)
10. What is RESTful API? - Definition from WhatIs.com. *SearchMicroservices* Available at: https://searchmicroservices.techtarget.com/definition/RESTful-API. (Accessed: 11th July 2019)