

# Sequence Bioinformatics

WS 2022/23

Tutor: Anupam Gautam

1	2	$\Sigma$

Marina Dittschar & Clarissa

Auckenthaler

## Blatt 1+2

(Abgabe am 02.11.2022)

### Assignment 01

Info: We used version 17.0.1 of java. To run our programs direct to the src folder in the terminal.

#### Task 1.2: FastA input and output (3 points)

FastA is only called by the other classes.

#### Task 1.3: FastA echo (1 point)

Enter the following code in the command line to run the file:

```
java EchoFastA_Auckenthaler_Dittschar.java 'dna.fasta' 'Auckenthaler_Dittschar_Echoout.fasta'
```

You can find the written output file in our submission: **Auckenthaler\_Dittschar\_Echoout.fasta**.

#### Task 1.4: DNA translation (4 points)

Enter the following code in the command line to run the file:

```
java Translate_Auckenthaler_Dittschar.java 'dna.fasta' 'Auckenthaler_Dittschar_Transout.fasta'
```

You can find the written output file in our submission: **Auckenthaler\_Dittschar\_Transout.fasta**

#### Task 1.5: Edit distance (2 points)

Enter the following code in the command line to run the file:

```
java EditDistance_Auckenthaler_Dittschar.java 'dna.fasta'
```

### Assignment 02

#### Task 2.1: Needleman-Wunsch basic implementation (3 points)

Enter the following code in the command line to run the file:

```
java GlobalAligner_Auckenthaler_Dittschar.java 'short.fasta' 'quadraticSpace'
```

```
java GlobalAligner_Auckenthaler_Dittschar.java 'medium.fasta' 'quadraticSpace'
```

```
java GlobalAligner_Auckenthaler_Dittschar.java 'long.fasta' 'quadraticSpace'
```

### Task 2.2: Needleman-Wunsch with linear space (4 points)

Our task 2.2 only computes the best score in linear space using the two-column trick. Computing  $c(m, n)$  (the row in which a traceback crosses the middle column, only for the full problem size) using the two-column trick.

Enter the following code in the command line to run the file:

```
java GlobalAligner__Auckenthaler__Dittschar.java 'short.fasta' 'linearSpace'
```

The optimal score for the short sequence is: 24 (same as for the Needleman-Wunsch quadratic space).

The middle point of the full problem is at column 26 and cell 23.

```
java GlobalAligner__Auckenthaler__Dittschar.java 'medium.fasta' 'linearSpace'
```

The optimal score for the medium sequence is: 79.

The middle point of the full problem is at column 74 and cell 92.

```
java GlobalAligner__Auckenthaler__Dittschar.java 'long.fasta' 'linearSpace'
```

The optimal score for the long sequence is: 635

The middle point of the full problem is at column 452 and cell 454.

### Task 2.3: Needleman-Wunsch, no table (2 points)

Enter the following code in the command line to run the file:

```
java GlobalAligner__Auckenthaler__Dittschar.java 'short.fasta' 'noDP'
```

We used different values of  $i, j = \{10, 15\}$ . The shortest sequence of 'short.fast' has a length of 53, because of the long run time we are not able to run that algorithm for the whole input file.

**Results for  $i=10, j=10$ :**

Optimal score Needleman-Wunsch recursively  $F(i,j)$ : 1

Total Runtime Needleman-Wunsch recursively: 77 ms

**Results for  $i=15, j=15$ :**

Optimal score Needleman-Wunsch recursively  $F(i,j)$ : 4

Total Runtime Needleman-Wunsch recursively: 121980 ms

Since for  $i=20, j=20$  the runtime was too long, we did not get any result. Total Runtime Needleman-Wunsch recursively: Indeterminate, very long time.

### Task 2.4: Comparison (1 point)

Needleman-Wunsch basic implementation (quadratic space)

- Runtime:  $O(nm)$
- Our Runtime: short= 14 ms | medium= 17 ms | long= 81 ms

- Space requirement =  $O(nm)$

Needleman-Wunsch with linear space

- Runtime:  $\sum_0^n \frac{1}{2^i} < 2$  times  $O(nm)$
- Our runtime: (short= 25 ms | medium= 93 ms )
- Space requirement:  $O(\min\{m,n\})$  (Source: [http://www.inf.fu-berlin.de/lehre/WS05/aldabi/downloads/pairAlign\\_part2.pdf](http://www.inf.fu-berlin.de/lehre/WS05/aldabi/downloads/pairAlign_part2.pdf))

Needleman-Wunsch, no table

- Runtime: Exponential
- Our Runtime: (Sequence of length 10 = 77 ms | Sequence of length 15 = 121980 ms)
- Space requirement: only two integers at any given time

We can see that, while the standard Needleman-Wunsch-Algorithm takes quadratic space and time, there are algorithms which only take linear space (with the two-column trick). There is a fundamental trade-off between space and time however, since the linear space algorithm takes twice as long as the quadratic algorithm. If we want to reduce the necessary memory even further (with a recursive implementation) the runtime requirements become exponential. Additionally, one fundamental disadvantage of the recursive algorithm is that it does not provide a traceback.