Exercise sheet 4: Sequence Alignment - affine gap costs

Exercise 1 - Waterman-Smith-Beyer traceback

Within the algorithm of Waterman, Smith and Beyer arbitrarily large gaps are considered. Thus, also the traceback has to investigate all gap sizes. This can be done following two strategies:

- check in increasing gap length (start with smallest gap)
- check in decreasing gap length (start with largest gap)

1a)

Given a gap function, are the strategies (measured in absolute runtime or number operations) expected to be equally performant or is one of them better than the other?

Hide

Possible Answers

- ☐ both are equally performant.
- ☐ checking in increasing direction performs better
- ☐ checking in decreasing direction performs better

Solution

- ☒ both are equally performant.
- ☐ checking in increasing direction performs better
- ☐ checking in decreasing direction performs better

(You still have to check all possible gap sizes)

1b)

Does the order of checking insertion and deletions change the runtime?

Hide

Possible Answers

- ☐ yes, checking insertions first will perform better
- ☐ yes, checking deletions first will perform better
- ☐ no, the order doesn't affect performance
- ☐ checking in an alternating way might improve performance

Solution

- ☐ yes, checking insertions first will perform better
- ☐ yes, checking deletions first will perform better
- ☒ no, the order doesn't affect performance
- ☐ checking in an alternating way might improve performance

(You still have to check all possible gaps and sizes)

Exercise 2 - Waterman-Smith-Beyer traceback

Given the Waterman-Smith-Beyer algorithm with the following scoring function:

$$s(x,y) = \begin{cases} -1 & if\ x = y \\ +1 & else \end{cases} \tag{1}$$

$$g(k) = 1 + k \tag{2}$$

When aligning sequences of very different lengths the penalizing of leading and trailing gaps, i.e unaligned sequence ends, dominates the alignment score.

We distinguish between leading and trailing gaps (example):

```
A-CC-CC-G
-TCCGCCT-
```

In this case the leading gaps are:

```
A-
-T
```

The trailing gaps are:

```
-G
T-
```

The algorithm by Waterman, Smith and Beyer can be adapted to treat end gaps with a score of 0. The adapted recursion formula is defined here, where n and m are the lengths of the respective sequences:

Let A and B be sequences of length n and m respectively

$$a_i \in A,\ with\ 1 \leq i \leq nb_j \in B,\ with\ 1 \leq j \leq m$$

Warning

The previously uploaded recursion schema was wrong. We will either try to fix this exercise or remove it completely. Sorry for any inconvenience.

(Also let us know if you can provide a correct recursion schema)

Exercise 3 - Gotoh Algorithm

Consider the following sequences $S1$, $S2$ and the similarity scoring via $s(x, y)$ and $g(k)$.

$$S1 = TCCGA \tag{3}$$

$$S2 = TACGCAGA \tag{4}$$

$$s(x,y) = \begin{cases} +1 & if\ x = y \\ 0 & else \end{cases} \tag{5}$$

$$g(k) = -4 - 1 * k \tag{6}$$

3a)

Which optimization scheme (minimization/maximization) needs to be applied to generate an useful sequence alignment.

Hide

Solution Maximization since we are scoring a similarity and not a distance.

3b)

Fill the dynamic programming matrices $D_{i,j}$, $Q_{i,j}$ and $P_{i,j}$ using the Gotoh algorithm!

(Remember: $D_{i,j}$ is the match/mismatch matrix. $Q_{i,j}$ corresponds to gaps in $S1$ whilst $P_{i,j}$ corresponds to gaps in $S2$)

Hide

Hint1: Formulae

$$g(k) = \alpha + k * \beta \tag{7}$$

$$\tag{8}$$

$$D_{i,j} = max \begin{cases} D_{i-1,j-1} + s(a_i, b_i) \\ P_{i,j} \\ Q_{i,j} \end{cases} \tag{9}$$

$$\tag{10}$$

$$Q_{i,j} = max \begin{cases} D_{i,j-1} + g(1) \\ Q_{i,j-1} + \beta \end{cases} \tag{11}$$

$$\tag{12}$$

$$P_{i,j} = max \begin{cases} D_{i-1,j} + g(1) \\ P_{i-1,j} + \beta \end{cases} \tag{13}$$

Hint2: Add the missing values

Solution

3c)

Calculate all optimal alignments and the corresponding score!

Hide

Solution Score: -3

```
T---CCGA    TCC---GA    TCCG---A
|   |:||    |:|   ||    |:||   |
TACGCAGA    TACGCAGA    TACGCAGA
```

3d)

Calculate the alignments using the Waterman-Smith-Beyer algorithm instead.

Hide

Solution The same alignments as in 3C, since the same scoring function is optimized. Calculation is just less efficient.

Exercise 4 - Programming assignment

Programming assignments are available via Github Classroom and contain automatic tests.

We recommend doing these assignments since they will help you to further understand this topic.

Access the Github Classroom link: Programming Assignment: Sheet 04.
