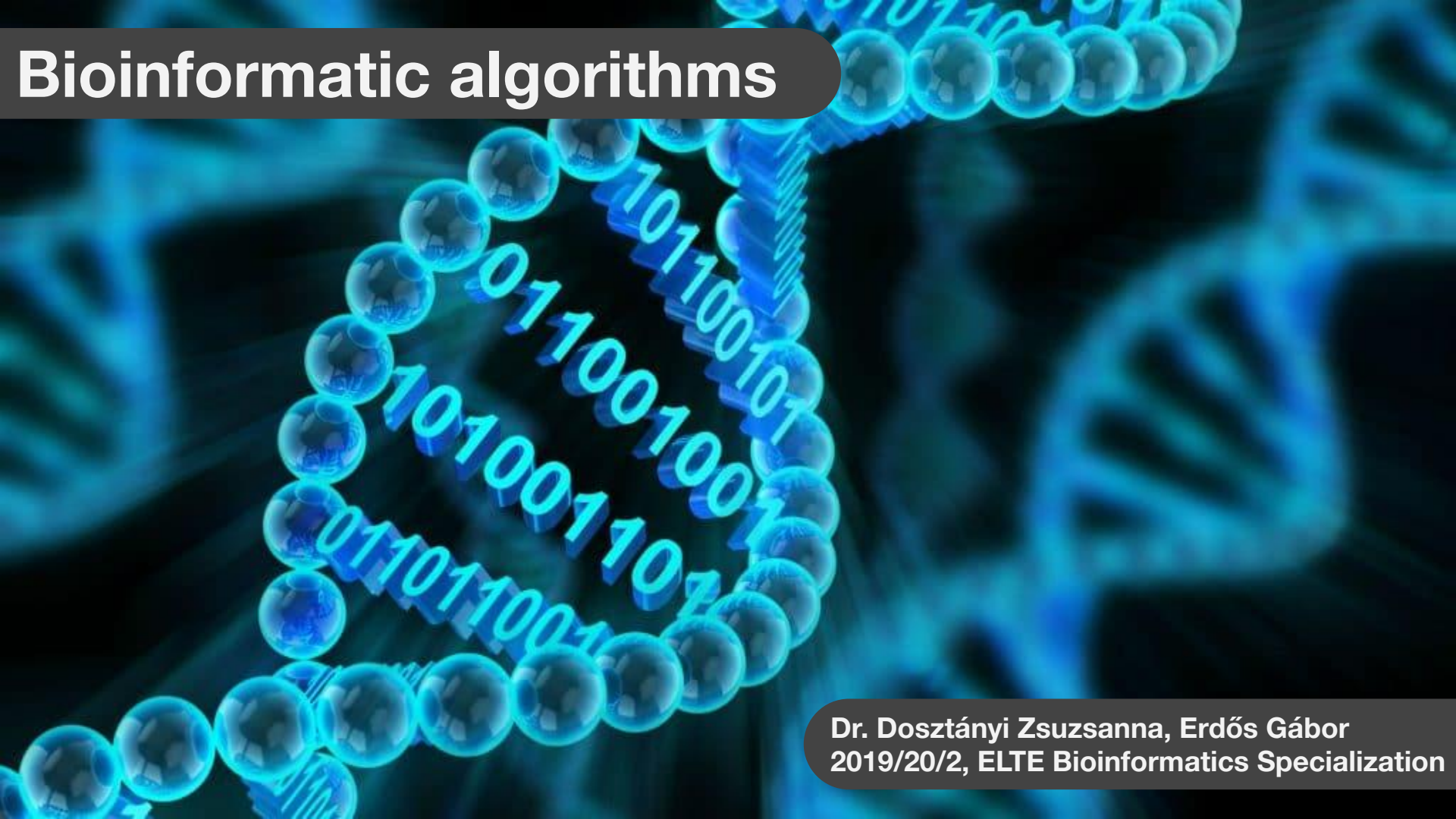
Bioinformatic algorithms



Dr. Dosztányi Zsuzsanna, Erdős Gábor
2019/20/2, ELTE Bioinformatics Specialization

Scoring alignments

```
YAFDLGYTCMFPVLLGGGELHIVQKETYTAPDEIAHYIKEHGITYIKLTPSLFHTIVNTASFAFDANFESLRLIVLGGEKIIPIDVIAFRKMYGHTE-FINHYGPTEATIGA
-AFDVSAGDFARALLTGGQLIVCPNEVKMDPASLYAIIKKYDITIFEATPALVIPLMEYI-YEQKLDISQLQILIVGSDSCSMEDFKTLVSRFGSTIRIVNSYGVTEACIDS
```

19

Simply maximizing matches produces unrealistic alignments

```
YAFDL--G-YTCMFP--VLL-GGGELHIV---Q-K-E--T-YTAPDEIAHYIK--EHGITYI---KLTPSL-FHT
-AFDVSAGD----FARA-LLTGG-QL-IVCPNEVKMDPASLY-A---I---IKKYD--IT-IFEA--TPALV---

IVNTASFAFDANFE-----S-LR-LIVLGG-----EKIIPIDVIAFRK-M---YGHTEFI---NHYGPTEATIGA
IPLMEYIY-----EQKLDISQLQILIV-GSDSCSME-----D---F-KTLVSRFGST--IRIVNSYGVTEACIDS
```

46

To generalize the alignment scoring model, we still award +1 for **matches**, but we also penalize **mismatches** by some positive constant μ (the **mismatch penalty**) and **indels** by some positive constant σ (the **indel penalty**). As a result, the score of an alignment is equal to the following expression:

$$\#\text{ matches} - \mu \cdot \#\text{ mismatches} - \sigma \cdot \#\text{ indels}$$

For example, with the parameters μ = 1 and σ = 2, the following alignment will be assigned a score of -4.

A	T	–	G	T	T	A	T	A
A	T	C	G	T	–	C	–	C
+1	+1	-2	+1	+1	-2	-1	-2	-1

Scoring matrices

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	–
A	2	-2	0	0	-3	1	-1	-1	-1	-2	-1	0	1	0	-2	1	1	0	-6	-3	-8
C	-2	12	-5	-5	-4	-3	-3	-2	-5	-6	-5	-4	-3	-5	-4	0	-2	-2	-8	0	-8
D	0	-5	4	3	-6	1	1	-2	0	-4	-3	2	-1	2	-1	0	0	-2	-7	-4	-8
E	0	-5	3	4	-5	0	1	-2	0	-3	-2	1	-1	2	-1	0	0	-2	-7	-4	-8
F	-3	-4	-6	-5	9	-5	-2	1	-5	2	0	-3	-5	-5	-4	-3	-3	-1	0	7	-8
G	1	-3	1	0	-5	5	-2	-3	-2	-4	-3	0	0	-1	-3	1	0	-1	-7	-5	-8
H	-1	-3	1	1	-2	-2	6	-2	0	-2	-2	2	0	3	2	-1	-1	-2	-3	0	-8
I	-1	-2	-2	-2	1	-3	-2	5	-2	2	2	-2	-2	-2	-2	-1	0	4	-5	-1	-8
K	-1	-5	0	0	-5	-2	0	-2	5	-3	0	1	-1	1	3	0	0	-2	-3	-4	-8
L	-2	-6	-4	-3	2	-4	-2	2	-3	6	4	-3	-3	-2	-3	-3	-2	2	-2	-1	-8
M	-1	-5	-3	-2	0	-3	-2	2	0	4	6	-2	-2	-1	0	-2	-1	2	-4	-2	-8
N	0	-4	2	1	-3	0	2	-2	1	-3	-2	2	0	1	0	1	0	-2	-4	-2	-8
P	1	-3	-1	-1	-5	0	0	-2	-1	-3	-2	0	6	0	0	1	0	-1	-6	-5	-8
Q	0	-5	2	2	-5	-1	3	-2	1	-2	-1	1	0	4	1	-1	-1	-2	-5	-4	-8
R	-2	-4	-1	-1	-4	-3	2	-2	3	-3	0	0	0	1	6	0	-1	-2	2	-4	-8
S	1	0	0	0	-3	1	-1	-1	0	-3	-2	1	1	-1	0	2	1	-1	-2	-3	-8
T	1	-2	0	0	-3	0	-1	0	0	-2	-1	0	0	-1	-1	1	3	0	-5	-3	-8
V	0	-2	-2	-2	-1	-1	-2	4	-2	2	2	-2	-1	-2	-2	-1	0	4	-6	-2	-8
W	-6	-8	-7	-7	0	-7	-3	-5	-3	-2	-4	-4	-6	-5	2	-2	-5	-6	17	0	-8
Y	-3	0	-4	-4	7	-5	0	-1	-4	-1	-2	-2	-5	-4	-4	-3	-3	-2	0	10	-8
–	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	

Figure: The PAM_{250} scoring matrix for protein alignment. Here, the indel penalty has been set to 8.

Global Alignment problem

Global Alignment Problem: *Find a highest-scoring alignment of two strings as defined by a scoring matrix.*

- **Input:** Two strings and a scoring matrix *Score*.
- **Output:** An alignment of the strings whose alignment score (as defined by *Score*) is maximized over all alignments of the strings.

To solve the Global Alignment Problem, we still must find a longest path in the alignment graph after updating the edge weights to reflect the values in the scoring matrix. Recalling that **deletions** correspond to vertical edges (↓), **insertions** correspond to horizontal edges (→), and **matches**/**mismatches** correspond to diagonal edges (↘/↘), we obtain the following recurrence for $s_{i,j}$, the length of a longest path from (0, 0) to (i, j):

$$s_{i,j} = \max \begin{cases} s_{i-1,j} & + Score(v_i, -) \\ s_{i,j-1} & + Score(-, w_j) \\ s_{i-1,j-1} & + Score(v_i, w_j) \end{cases}$$

Global Alignment

Code Challenge: Solve the Global Alignment Problem.

- **Input:** Two protein strings written in the single-letter amino acid alphabet.
- **Output:** The maximum alignment score of these strings followed by an alignment achieving this maximum score. scoring matrix for matches and mismatches as well as the indel penalty σ = 5.

Sample Input:

```
PLEASANTLY
MEANLY
```

Sample Output:

```
8
PLEASANTLY
-MEA--N-LY
```

Use the Blosum matrix

Limitations of global alignment

Global alignment seeks similarities between two strings across their entire length; However, sometimes we need to find a conserved segment within much longer genes and ignore the flanking areas, which exhibit little similarity. So are looking for smaller, *local* regions of similarity and do not need to align the entire strings.

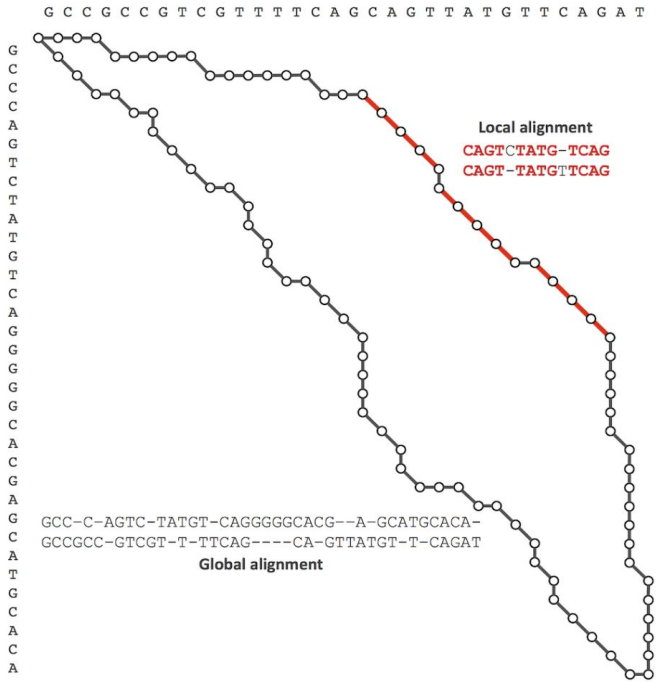
For example, the global alignment between the two sequences below has 22 **matches**, 18 indels, and 2 mismatches, resulting in the score 22 − 18 - 2 = 2 (if σ = μ = 1).

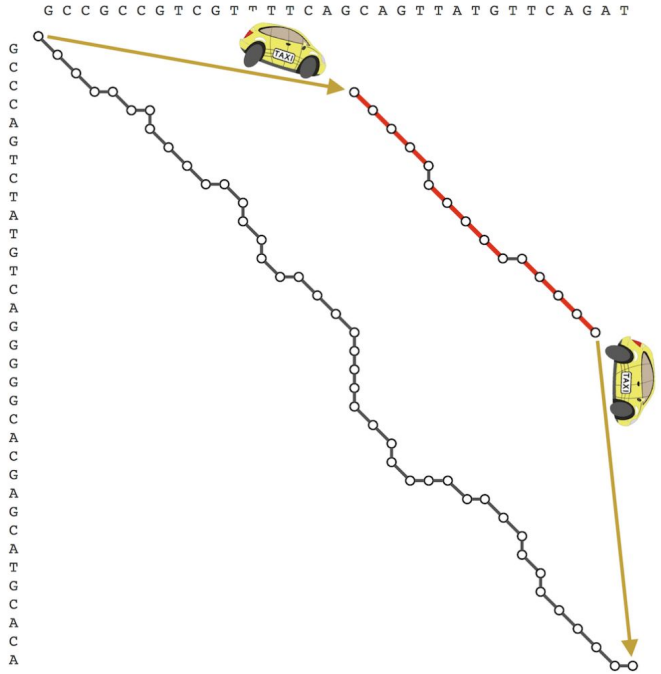
```
GCC-C-AGTC-TATGT-CAGGGGGCACG--A-GCATGCACA-
GCCGCC-GTCGT-T-TTCAG----CA-GTTATGT-T-CAGAT
```

However, these sequences can be aligned differently (with 17 matches and 32 indels) based on a highly conserved interval represented by the substrings CAGTCTATGTCAG and CAGTTATGTTCAG:
This alignment has fewer matches and a lower score of 17 − 32 = −15, even though the conserved region of the alignment contributes a score of 12 - 2 = 10, which is hardly an accident.

```
---G----C-----C--CAGTCTATG-TCAGGGGGCACGAGCATGCACA
GCCGCCGTCGTTTTCAGCAGT-TATGTTCAG-----A------T-----
```

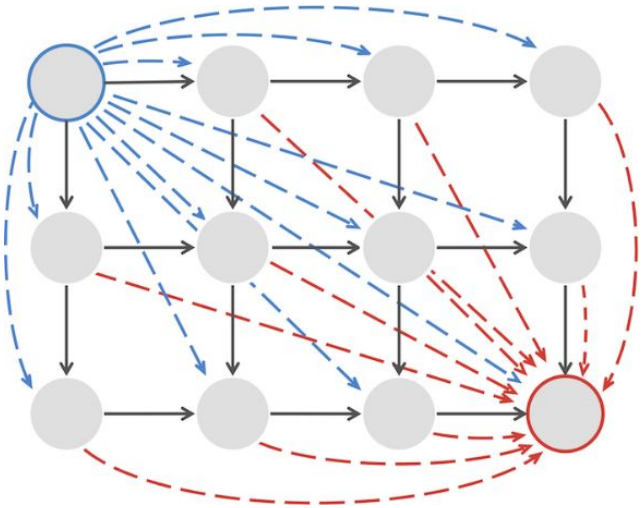
Global vs local alignment





Global and local alignments of two DNA strings that share a highly conserved interval. The relevant alignment that captures this interval (upper path) loses to an irrelevant alignment (lower path), since the former incurs heavy indel penalties.

Local alignment algorithm



To compute the values $s_{i,j}$, we can add zero-weight edges from (0, 0) to every node has made the source node (0, 0) a predecessor of every node (*i*, *j*). Therefore, there are now four edges entering (*i*, *j*), which adds only one new term to the longest path recurrence relation:

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + Score(v_i, -) \\ s_{i,j-1} + Score(-, w_j) \\ s_{i-1,j-1} + Score(v_i, w_j) \end{cases}$$

The recurrence above incorporates free rides from *source* = (0, 0), but it does not incorporate free rides into *sink* = (*n*, *m*). Since *sink* has every other node as a predecessor, $s_{n,m}$ will be equal to the largest value of $s_{i,j}$ over the entire alignment graph,

$$s_{n,m} = \max_{0 \le i \le n,\, 0 \le j \le n} s_{i,j}$$

Local alignment problem

Code Challenge: Solve the Local Alignment Problem.

- **Input:** Two protein strings written in the single-letter amino acid alphabet.
- **Output:** The maximum score of a local alignment of the strings, followed by a local alignment of these strings achieving the maximum score. Use the PAM250 scoring matrix for matches and mismatches as well as the indel penalty σ = 5.

Sample Input:

```
MEANLY
PENALTY
```

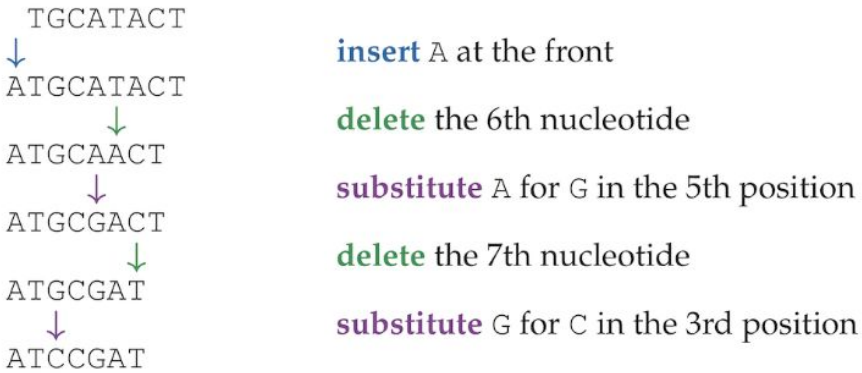
Sample Output:

```
15
EANL-Y
ENALTY
```

Problem 1: Edit distance

In 1966, Vladimir Levenshtein introduced the notion of the **edit distance** between two strings as the minimum number of **edit operations** needed to transform one string into another. Here, an edit operation is the insertion, deletion, or substitution of a single symbol.

What is the edit distance between TGCATACT into ATCCGAT ?



Edit distance

Edit Distance Problem: *Find the edit distance between two strings.*

- **Input:** Two strings.
- **Output:** The edit distance between these strings.

Code Challenge: Solve the Edit Distance Problem.

Sample Input:

```
PLEASANTLY
MEANLY
```

Sample Output:

```
5
```

Problem 2: Fitting alignment

Say that we wish to compare the approximately 20,000 amino acid-long NRP synthetase from *Bacillus brevis* with the approximately 600 amino acid-long A-domain from *Streptomyces roseosporus*, the bacterium that produces the powerful antibiotic Daptomycin. We hope to find a region within the longer protein sequence *v* that has high similarity with all of the shorter sequence *w*. Global alignment will not work because it tries to align all of *v* to all of *w*; local alignment will not work because it tries to align substrings of both *v* and *w*. Thus, we have a distinct alignment application called the **Fitting Alignment Problem**.

"Fitting" *w* to *v* requires finding a substring *v'* of *v* that maximizes the global alignment score between *v'* and *w* among all substrings of *v*. For example, the best global, local, and fitting alignments of *v* = CGTAGGCTTAAGGTTA and *w* = ATAGATA are shown in the figure below (with mismatch and indel penalties equal to 1).

Fitting Alignment Problem: *Construct a highest-scoring fitting alignment between two strings.*

- **Input:** Strings *v* and *w* as well as a matrix *Score*.
- **Output:** A highest-scoring fitting alignment of *v* and *w* as defined by the scoring matrix *Score*.

Note in the figure that the optimal local alignment (with score 3) is not a valid fitting alignment. On the other hand, the score of the optimal global alignment (6 - 9 - 1 = -4) is smaller than that of the best fitting alignment (5 - 2 - 2 = +1).

Fitting alignment

Code Challenge: Solve the Fitting Alignment Problem.

- **Input:** Two nucleotide strings *v* and *w*, where *v* has length at most 1000 and *w* has length at most 100.
- **Output:** A highest-scoring fitting alignment between *v* and *w*. Use the simple scoring method in which matches count +1 and both the mismatch and indel penalties are 1.

Sample Input:

```
GTAGGCTTAAGGTTA
TAGATA
```

Sample Output:

```
2
TAGGCTTA
TAGA--TA
```

Problem 3: Overlap alignment

In the chapter on genome assembly, we discussed how to use overlapping reads to assemble a genome, a problem that was complicated by errors in reads. Aligning the ends of the hypothetical reads shown below offers a way to find overlaps between error-prone reads.

```
ATGCATGCCGG
   T-CC-GAAAC
```

An **overlap alignment** of strings $v = v_1 \ldots v_n$ and $w = w_1 \ldots w_m$ is a global alignment of a suffix of v with a prefix of w. An optimal overlap alignment of strings v and w maximizes the global alignment score between an i-suffix of v and a j-prefix of w (i.e., between $v_i \ldots v_n$ and $w_1 \ldots w_j$) among all i and j.

Overlap alignment

Code Challenge: Solve the Overlap Alignment Problem.

- **Input:** Two strings *v* and *w*, each of length at most 1000.
- **Output:** The score of an optimal overlap alignment of *v* and *w*, followed by an alignment of a suffix *v'* of *v* and a prefix *w'* of *w* achieving this maximum score. Use an alignment score in which matches count +1 and both the mismatch and indel penalties are 2.

Sample Input:

```
PAWHEAE
HEAGAWGHEE
```

Sample Output:

```
1
HEAE
HEAG
```

Multiple sequence alignment

1. Extend alignment algorithm to multiple sequences.
   a. 2 sequences : 2D matrix
   b. 3 sequences : 3D matrix
2. Greedy approach

```
AT-GTTaTA
AgCGaTC-A
ATCGT-CTc
```

induces three pairwise alignments:

```
AT-GTTaTA
AgCGaTC-A
```

```
AT-GTTaTA
ATCGT-CTc
```

```
AgCGaTC-A
ATCGT-CTc
```

But can we work in the opposite direction, combining optimal pairwise alignments into a multiple alignment?

3. Profile alignment