



# Lecture 2 - Biological Sequence Alignment



## Inexact String Matching

# Lecture Topics

1. **Exact Pattern Matching.**
2. **Inexact string matching and the edit distance.**
3. **Dynamic Programming.**
4. **Biological Database search.**
5. **FASTA, BLAST.**
6. **Substitution Matrices – PAM, BLOSSUM (exercise).**
7. **Multiple sequence alignment.**

# Recommended Reading :

1. Dan Gusfield, Algorithms on Strings, Trees and Sequences, (1997) – Part III.
2. M. Waterman, Introduction to Computational Biology.
3. Setubal and Meidanis, Introduction to Computational Biology.

# Motivation :

- Biomolecular sequence (DNA, RNA, protein) similarity implies structural and functional similarity.
- “Successful” biological structures are duplicated in the evolutionary process modulo mutations.
- Abundance of biological sequence data and the need to query large databases.

# The Exact Pattern Matching Problem :

- Given a **pattern P** of length **m** and a (longer) **string T** of length **n**, find all the occurrences of **P** in **T**.
- Example :  $P = \text{aba}$ ,  $T = \text{bbaababaskababa}$ ;  $m = 3$ ,  $n = 15$ .  
 $P$  occurs in  $T$  at locations 4 , 6, 11, 13.  
Note, that the matches may overlap.

# Complexity of the exact pattern matching :

- Naive algorithm  $O(m*n)$ .
- Sophisticated algorithms (Boyer - Moore, Knuth-Pratt-Morris) -  $O(n+m)$  (see Gusfield's book for a thorough presentation).

# Evolution of Biological sequences

- Due to mutations in the DNA sequences : **substitutions, insertions, deletions, reversals.**
- Assuming the reading frame is not altered, the DNA mutations imply similar mutations in the protein sequences.

# The Dynamic Programming (DP) Method

- The need to align inexact sequence data appeared in different fields, where the data can be presented as a discrete ordered sequence of numerals (e.g. speech signals as function of time).
- Key to the DP method – the optimum at each “time frame” can be computed using a “*short memory*”.

# The Edit Distance

- Edit operations : **substitution, insertion, deletion.**
- Each operation has a **weight**, which might be location dependent.
- In biological tasks the *penalty* for a long gap should be smaller than the sum of single gap *penalties* (*introduces computational difficulty*).

# The Edit Distance

**Definition** : The **edit distance** is the minimal no. of edit operations required to transform the first sequence into the second (matches are not counted).

# Global Sequence Alignment

Def.:  $S_1, S_2$  – strings.

$D(i,j)$  – edit distance btwn  $S_1[1..i]$  and  $S_2[1..j]$ .

If  $S_1$  is of length  $n$  and  $S_2$  of length  $m$ , then the edit distance btwn  $S_1$  and  $S_2$  is  **$D(n,m)$** .

**$D(n,m)$**  is computed recursively by the Dynamic Programming (DP) method.

# The DP matrix

Base conditions :

$$D(i,0) = i ; D(0,j) = j .$$

Note : equal penalty (+1) for insertions/deletions  
(and substitutions).

Recurrence relation :

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i-1,j-1) + t(i,j) \\ D(i,j-1) + 1 \end{cases}$$

$t(i,j) = 0$  (match) or  $1$  (mismatch=substitution)

# DP matrix – bottom-up computation

| $D(i, j)$ |   | w | r | i | t | e | r | s |   |
|-----------|---|---|---|---|---|---|---|---|---|
|           |   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|           | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| v         | 1 | 1 |   |   |   |   |   |   |   |
| i         | 2 | 2 |   |   |   |   |   |   |   |
| n         | 3 | 3 |   |   |   |   |   |   |   |
| r         | 4 | 4 |   |   |   |   |   |   |   |
| a         | 5 | 5 |   |   |   |   |   |   |   |
| e         | 6 | 6 |   |   |   |   |   |   |   |
| r         | 7 | 7 |   |   |   |   |   |   |   |

Figure 11.1: Table to be used to compute the edit distance between vlttrne and wrttrs. The values in row 0 and column zero are already included. They are given directly by the base conditions.

from Gusfield

| $D(i, j)$ |   | w | r | i | t | e | r | s |   |
|-----------|---|---|---|---|---|---|---|---|---|
|           |   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|           | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| v         | 1 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| i         | 2 | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 |
| n         | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 6 |
| r         | 4 | 4 | 4 | 4 | 4 | * |   |   |   |
| a         | 5 | 5 |   |   |   |   |   |   |   |
| e         | 6 | 6 |   |   |   |   |   |   |   |
| r         | 7 | 7 |   |   |   |   |   |   |   |

Figure 11.2: Edit distances are filled in one row at a time, and in each row they are filled in from left to right. The example shows the edit distances  $D(i, j)$  to column 3 of row 4. The next value to be computed is here an asterisk appears. The value for cell (4, 4) is 3, since  $S_1(4) = S_2(4) = r$  and  $D(3, 3) = 3$ .

# DP solution - traceback

from Gusfield

| $D(i, j)$ |     | w     | r     | i     | t   | e   | r   | s   |
|-----------|-----|-------|-------|-------|-----|-----|-----|-----|
|           | 0   | 1     | 2     | 3     | 4   | 5   | 6   | 7   |
| 0         | 0   | ← 1   | ← 2   | ← 3   | ← 4 | ← 5 | ← 6 | ← 7 |
| v 1       | ↑ 1 | ↖ 1   | ↖ 2   | ↖ 3   | ↖ 4 | ↖ 5 | ↖ 6 | ↖ 7 |
| i 2       | ↑ 2 | ↖ ↑ 2 | ↖ 2   | ↖ 2   | ← 3 | ← 4 | ← 5 | ← 6 |
| n 3       | ↑ 3 | ↖ ↑ 3 | ↖ ↑ 3 | ↖ ↑ 3 | ↖ 3 | ↖ 4 | ↖ 5 | ↖ 6 |
| j 4       | ↑ 4 | ↖ ↑ 4 | ↖ ↑ 4 | ↖ ↑ 4 | ↖ 3 | ↖ 4 | ↖ 5 | ↖ 6 |
| n 5       | ↑ 5 | ↖ ↑ 5 | ↖ ↑ 5 | ↖ ↑ 5 | ↑ 4 | ↖ 4 | ↖ 5 | ↖ 6 |
| e 6       | ↑ 6 | ↖ ↑ 6 | ↖ ↑ 6 | ↖ ↑ 6 | ↑ 5 | ↖ 4 | ↖ 5 | ↖ 6 |
| r 7       | ↑ 7 | ↖ ↑ 7 | ↖ 6   | ↖ ↑ 7 | ↑ 6 | ↑ 5 | ↖ 4 | ← 5 |

Figure 11.3: The complete dynamic programming table with pointers included. The arrow ← in cell  $(i, j)$  points to cell  $(i, j-1)$ , the arrow ↑ points to cell  $(i-1, j)$ , and the arrow ↖ points to cell  $(i-1, j-1)$ .

it is possible to either go up or to go diagonally. The three optimal alignments are:

```

w r i t - e r s
v i n t n e r -

w r i - t - e r s
v - i n t n e r -

```

and

```

w r i - t - e r s
- v i n t n e r -

```

Complexity : matrix bldg. -  **$O(m*n)$**  ; traceback –  $O(m+n)$

# Local Sequence Alignment/Similarity

**Def.**:  $S_1, S_2$  – strings. Find substrings  $\alpha, \beta$  of  $S_1, S_2$ , whose similarity is maximum over all pairs of substrings from  $S_1, S_2$ .

## Motivation :

Detection of remote similarity between evolutionary related sequences.

Motif detection.

# Computing Local Alignment /Similarity (see Gusfield, chapter 11)

(Restriction on global alignment scoring: global alignment of two empty strings =0.)

Def: *local suffix alignment* problem.

# Sequence Database Searches

- FASTA - fast all
- BLAST - basic local alignment search tool.
- Heuristic algorithms which speed up the search significantly.

# FASTA

- Finds favorable sub-diagonals in the dynamic programming matrix by hashing k-tuples (k=6 for DNA, k=2 for proteins).
- Sub-diagonals represent alignments with no insertions/deletions and possible substitutions.
- Good subalignments are combined, allowing some insertions/deletions.

# Multiple Sequence Alignment

- Lesk : “ One or two homologous sequences whisper ... a full multiple alignment shouts out loud.”
- Allows inference on conserved sequence features in biologically related sequences.
- Facilitates deduction of evolutionary information.

# Computational Method:

- Score : sum of pairs - mathematical convenience.
- Algorithm : Multi-dimensional dynamic programming.

# Repeated motif methods

- Find a substring that is common to many strings in the ensemble (**motif**).
- Align the sequences according to the motif.
- Continue recursively on both sides of the motif.