FACULDADE ENGENHARIA

Departamento de Informática

Bioinformática
(15861, 13422)

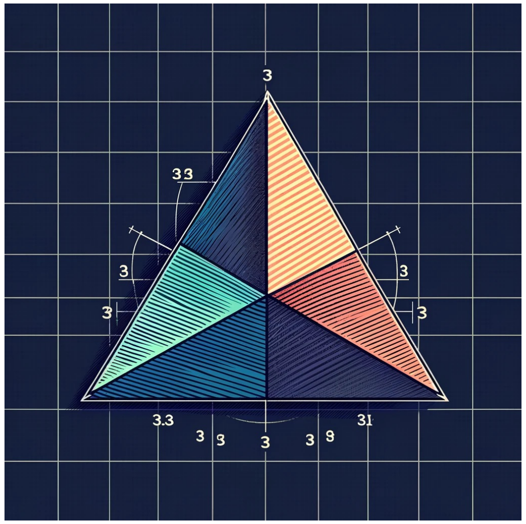
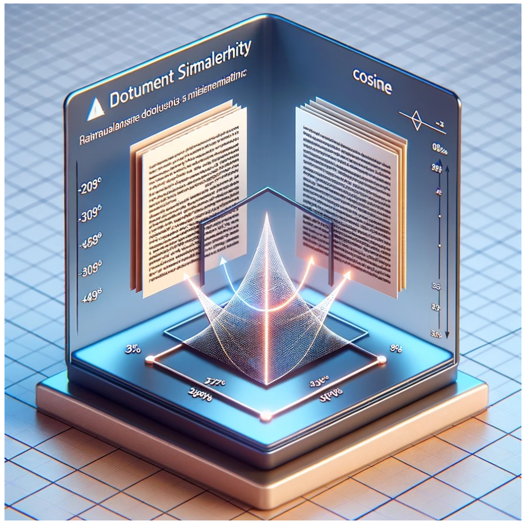
Sequence Alignment

FACULDADE ENGENHARIA
Departamento de Informática

Sequence Alignment

- Objectives for today's lesson:
  - Understand the concept of sequence alignment.
  - Explore multiple algorithms and scenarios.
  - Hands-on exercises and in-class activities.

Similarity?

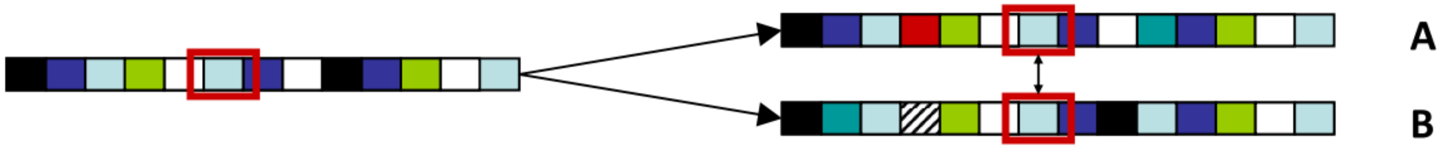



FACULDADE ENGENHARIA
Departamento de Informática

Sequence Alignment in Biology

- **Biological Question**

The purpose of a sequence alignment is to line up all residues in the inputted sequence(s) for maximal level of similarity, on the sense of their functional or evolutionary relationship.



FACULDADE ENGENHARIA

Departamento de Informática

Why Scoring?

- **Scoring matrices** in bioinformatics are tools used to quantify the similarity between biological sequences, which can be DNA, RNA, or proteins.
- To understand scoring matrices, let's first dive into what they are and then use an analogy to make the concept more accessible.

FACULDADE ENGENHARIA
Departamento de Informática

Why Scoring?

- **Scoring matrices** are tables used in sequence alignment that assign scores to each possible pair of residues (nucleotides or amino acids).
- These scores are based on the likelihood of one residue being replaced by another over evolutionary time. A high score is given to pairs that are likely to be conserved (remain the same), while substitutions that are less likely or more detrimental get lower scores.

Analogy

- Think of a scoring matrix as a menu in a restaurant where each dish is priced based on its ingredients and the effort to prepare it. In sequence alignment, the "dishes" are combinations of nucleotides or amino acids. A "luxury dish" (high score) in the menu is a pair of residues that commonly go together and have been preserved through evolution due to their functional importance, like a well-loved classic recipe. A "discounted dish" (low score) represents a rare and possibly non-functional or less optimal pairing, like an experimental dish that is not commonly ordered.

- The "prices" (scores) help in deciding the "order" (best sequence alignment) by choosing combinations that give the highest total value (optimal alignment score), indicating the best evolutionary and functional relationship between the sequences being compared.

FACULDADE ENGENHARIA

Departamento de Informática

From LCS to Alignment: Change the Scoring

- **Recall:** The Longest Common Subsequence (LCS) problem allows only insertions and deletions (no mismatches).

- In the LCS Problem, we scored 1 for matches and 0 for *indels*, so our alignment score was simply equal to the total number of matches.

- Let’s consider penalizing mismatches and *indels* instead.

FACULDADE ENGENHARIA

Departamento de Informática

From LCS to Alignment: Change the Scoring

- Simplest *scoring schema*: For some positive numbers μ and σ:
  - **Match Premium:** $+1$
  - **Mismatch Penalty:** $-\mu$
  - **Indel Penalty:** $-\sigma$

- Under these assumptions, the alignment score becomes as follows:

$$\text{Score} = \#matches - \mu(\#mismatches) - \sigma(\#indels)$$

- Our specific choice of μ and σ depends on how we wish to penalize mismatches and *indels*.

The Global Alignment Problem

- <u>Input</u> : Strings **v** and **w** and a scoring schema

- <u>Output</u> : An alignment with maximum score

- We can use dynamic programming to solve the Global Alignment Problem:

$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + 1 & \text{if } v_i = w_j \\ s_{i-1,j-1} - \mu & \text{if } v_i \neq w_j \\ s_{i-1,j} - \sigma & \\ s_{i,j-1} - \sigma & \end{cases}$$

μ : mismatch penalty
σ : indel penalty

FACULDADE ENGENHARIA

Departamento de Informática

Scoring Matrices

- To further generalize the scoring of alignments, consider a (4+1) x (4+1) **scoring matrix** δ.
  - The purpose of the scoring matrix is to score one nucleotide against another, e.g. A matched to G may be "worse" than C matched to T.
  - The addition of 1 is to include the score for comparison of a gap character "-".
- This will simplify the algorithm to the dynamic formula at right:

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

FACULDADE ENGENHARIA
Departamento de Informática

Scoring Matrices: Example

- Say we want to align AGTCA and CGTTGG with the following scoring matrix:

	A	G	T	C	—
A	1	-0.8	-0.2	-2.3	-0.6
G	-0.8	1	-1.1	-0.7	-1.5
T	-0.2	-1.1	1	-0.5	-0.9
C	-2.3	-0.7	-0.5	1	-1
—	-0.6	-1.5	-0.9	-1	n/a

Sample Alignment:

```
A - GTC - A
-  CGTT GG
```

Score: –0.6 – 1 + 1 + 1 – 0.5 – 1.5 – 0.8 = –2.4

Departamento de Informática

Exercise: Aligning Protein Sequences

- Consider the following matrix:

	A	C	G	T	—
A	4	-1	-2	-1	-5
C	-1	4	-1	-2	-5
G	-2	-1	4	-1	-5
T	-1	-2	-1	4	-5
—	-5	-5	-5	-5	n/a

- And the given sequences:

  Sequence X: ACTG

  Sequence Y: GACT

Departamento de Informática

Exercise: Aligning Protein Sequences

- Consider the following matrix:

	A	C	G	T	—
A	4	-1	-2	-1	-5
C	-1	4	-1	-2	-5
G	-2	-1	4	-1	-5
T	-1	-2	-1	4	-5
—	-5	-5	-5	-5	n/a

1. List possible alignments (include up to one gap in each sequence).
2. Calculate the score for each alignment using the hypothetical scoring matrix.
3. Find the alignment with the highest score.

Departamento de
Informática

Solution

Possible alignments:

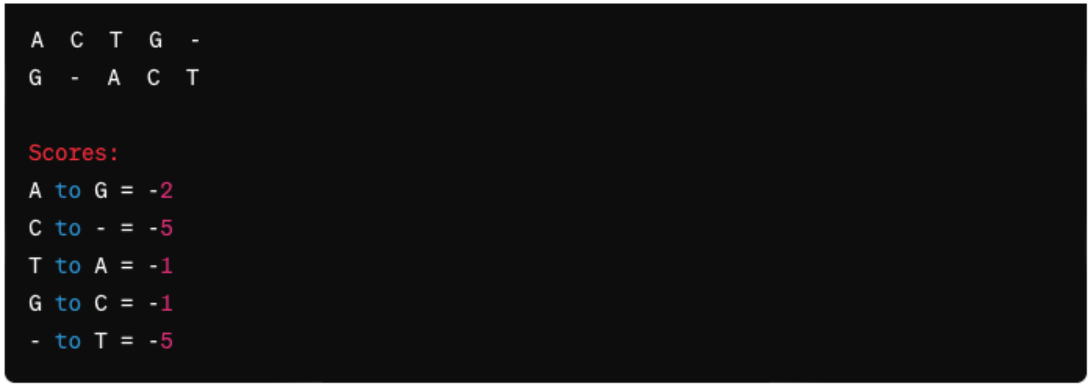
1. ACTG-

   G-ACT

2. A-CTG

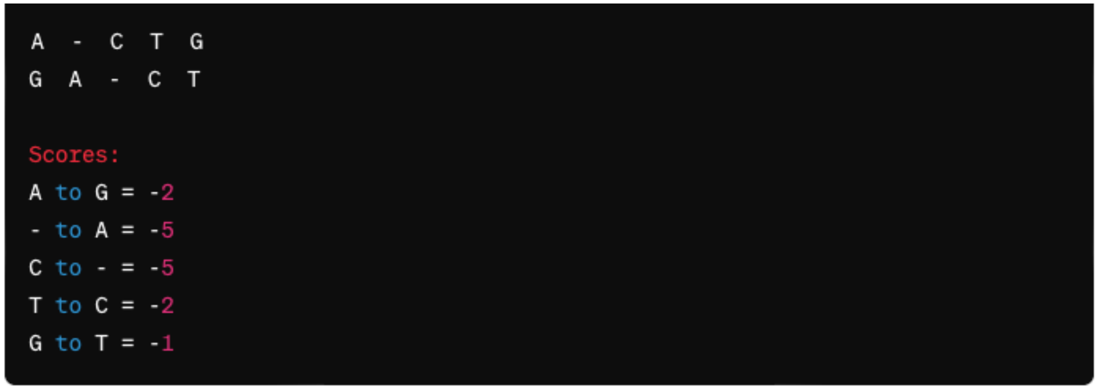
   GA-CT

3. There is a better option?



```
A  C  T  G  -
G  -  A  C  T

Scores:
A to G = -2
C to - = -5
T to A = -1
G to C = -1
- to T = -5
```

Total score for Alignment 1: -2 - 5 - 1 - 1 - 5 = -14



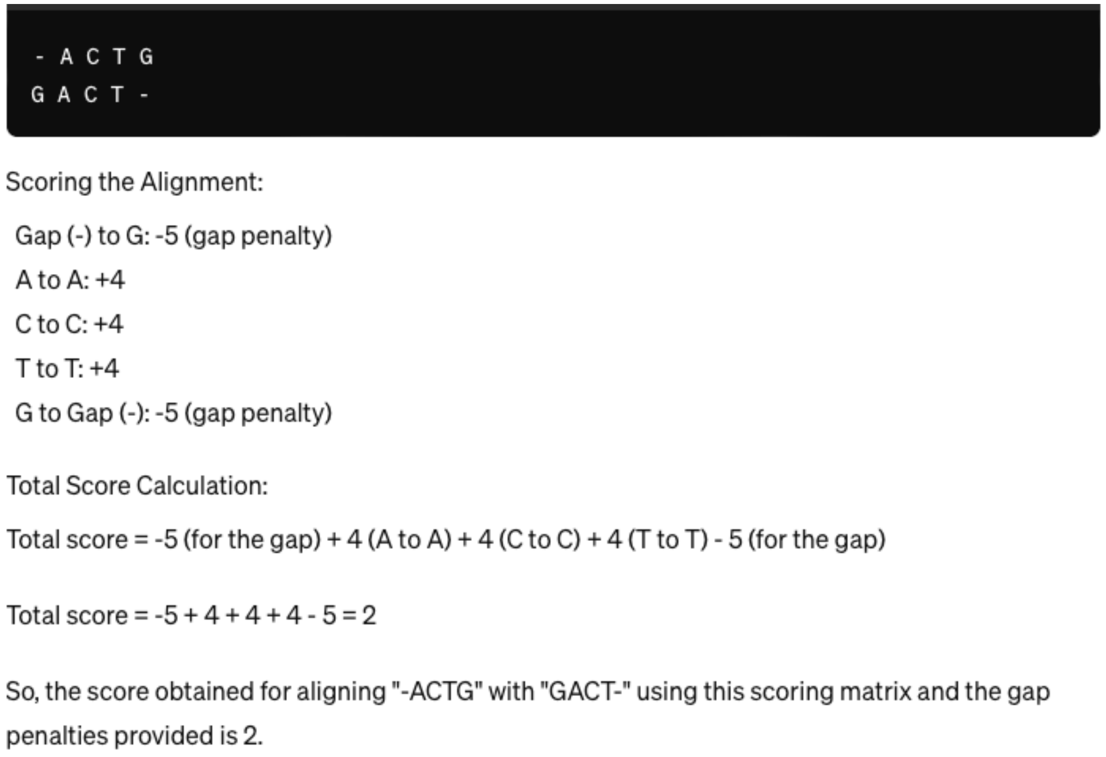
```
A  -  C  T  G
G  A  -  C  T

Scores:
A to G = -2
- to A = -5
C to - = -5
T to C = -2
G to T = -1
```

Total score for Alignment 2: -2 - 5 - 5 - 2 - 1 = -15

Solution

Yes, we can:

3. –ACTG
   GACT-



```
- A C T G
G A C T -
```

Scoring the Alignment:

Gap (-) to G: -5 (gap penalty)

A to A: +4

C to C: +4

T to T: +4

G to Gap (-): -5 (gap penalty)

Total Score Calculation:

Total score = -5 (for the gap) + 4 (A to A) + 4 (C to C) + 4 (T to T) - 5 (for the gap)

Total score = -5 + 4 + 4 + 4 - 5 = 2

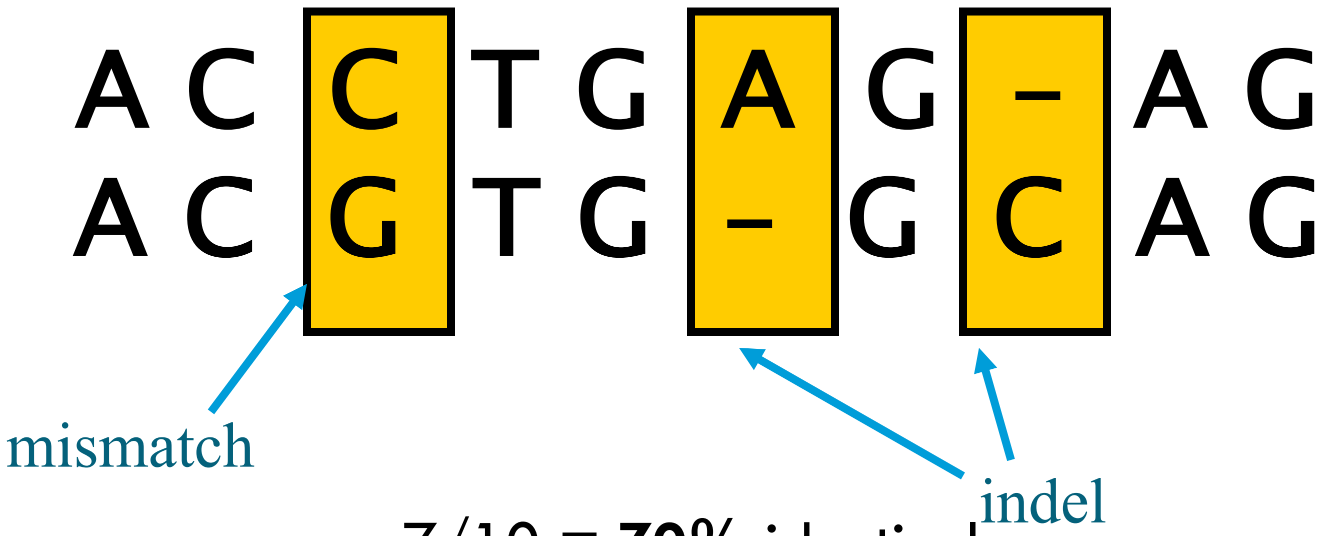
So, the score obtained for aligning "-ACTG" with "GACT-" using this scoring matrix and the gap penalties provided is 2.

Departamento de Informática

Percent Sequence Identity

- **Percent Sequence Identity:** The extent to which two nucleotide or amino acid sequences are invariant.

- **Example:**



7/10 = **70%** identical

How Do We Make a Scoring Matrix?

- Scoring matrices are created based on biological evidence.
- Alignments can be thought of as two sequences that differ due to mutations.
- Some of these mutations have little effect on the protein’s function, therefore some penalties, $\delta(v_i , w_j)$, will be less harsh than others.
- This explains why we would want to have a scoring matrix to begin with.

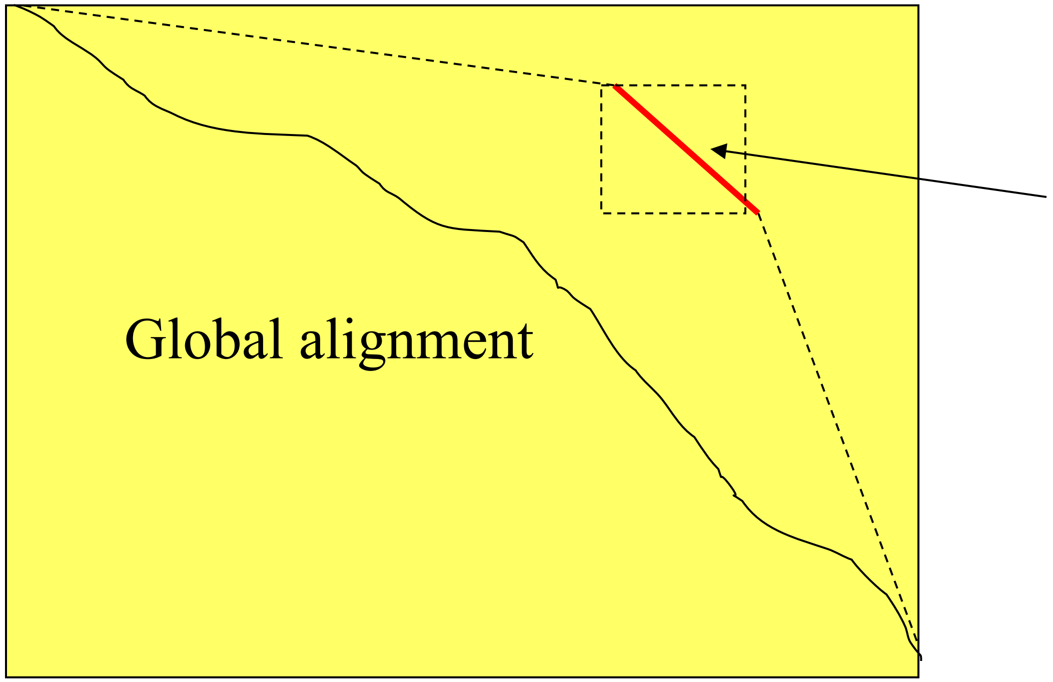
FACULDADE ENGENHARIA
Departamento de Informática

Local Alignment: Why?

- Two genes in different species may be similar over short conserved regions and dissimilar over remaining regions.

- The main purpose is to search for an alignment which has a positive score *locally*, meaning that an alignment on substrings of the given sequences has a positive score.

Local Alignment: Illustration



Compute a "mini" Global Alignment to get Local Alignment

Local vs. Global Alignment: Example

- Global Alignment:

```
--T—-CC-C-AGT—-TATGT-CAGGGGACACG—A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG—T-CAGAT--C
```

- Local Alignment—better alignment to find conserved segment:

```
                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

The Local Alignment Problem

- Goal: Find the best local alignment between two strings.

- Input : Strings ***v*** and ***w*** as well as a scoring matrix δ

- Output : Alignment of substrings of ***v*** and ***w*** whose alignment score is maximum among all possible alignments of all possible substrings of ***v*** and ***w***.

FACULDADE ENGENHARIA
Departamento de Informática

Local Alignment: How to Solve?

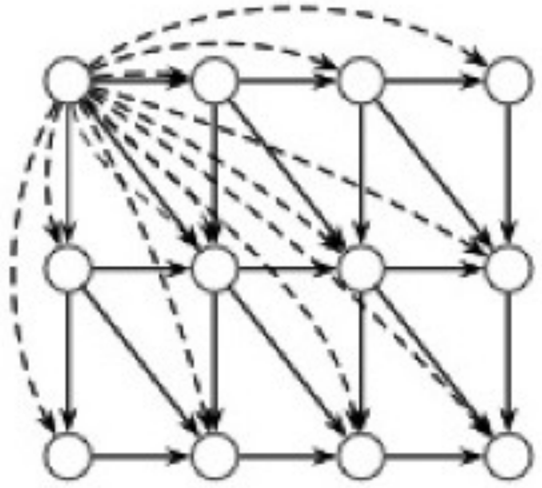
- We have seen that the Global Alignment Problem tries to find the longest path between vertices (0,0) and (*n*,*m*) in the edit graph.

- The **Local Alignment Problem** tries to find the longest path among paths between *arbitrary vertices* (*i*,*j*) and (*i'*,*j'*) in the edit graph.

- **Key Point:** In the edit graph with negatively-scored edges, Local Alignment may score higher than Global Alignment.

Local Alignment Solution: Free Rides

- The solution actually comes from *adding* vertices to the edit graph.
- The dashed edges represent the "free rides" from (0, 0) to every another node.
  - Each "free ride" is assigned an edge weight of 0.
  - If we start at (0, 0) instead of (*i*, *j*) and maximize the longest path to (*i*', *j*'), we will obtain the local alignment.



Departamento de Informática

Smith-Waterman Local Alignment Algorithm

- The largest value of $s_{i,j}$ over the whole edit graph is the score of the best local alignment.
- The recurrence:

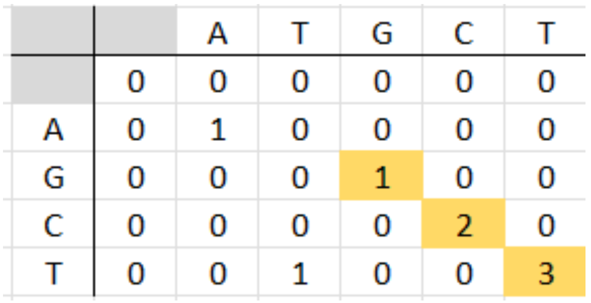
$$s_{i,j} = \max\begin{cases} 0 \\ s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, w_j) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

- Notice that the 0 is the only difference between the global alignment recurrence…hence our new algorithm is O(n^2)!

Smith-Waterman Local Alignment Algorithm

Exercise#1

Find the local alignment for the following sequences: **ATGCT**, and **AGCT** (using: Gap:-2; Match:1, Mismatch: -1).



		A	T	G	C	T
	0	0	0	0	0	0
A	0	1	0	0	0	0
G	0	0	0	1	0	0
C	0	0	0	0	2	0
T	0	0	1	0	0	3

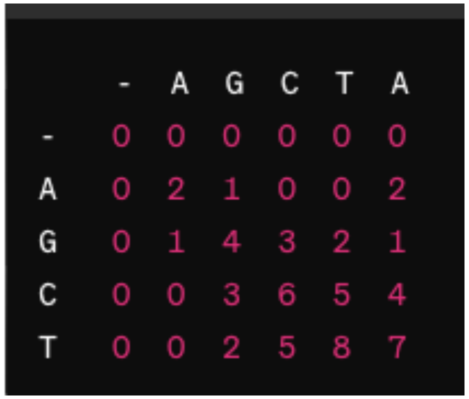
Solution: GCT

Smith-Waterman Local Alignment Algorithm

Exercise#2

- Use the Smith-Waterman algorithm to find the optimal local alignment between two DNA sequences.
  - Sequence 1: AGCT
  - Sequence 2: AGCTA

  - Scoring Scheme:
    - Match: 2
    - Mismatch: -1
    - Gap: -1



	-	A	G	C	T	A
-	0	0	0	0	0	0
A	0	2	1	0	0	2
G	0	1	4	3	2	1
C	0	0	3	6	5	4
T	0	0	2	5	8	7

FACULDADE ENGENHARIA

Departamento de Informática

Hamming Distance

- The Hamming Distance $d_H(v, w)$ between two DNA sequences v and w of the same length is equal to the number of places in which the two sequences differ.

- Example: Given as follows, $d_H(v, w) = 8$:

```
v:  ATATATAT
w:  TATATATA
```

- However, note that these sequences are still very similar.
  - Hamming Distance is therefore not an ideal similarity score, because it ignores insertions and deletions.

Edit Distance

- Levenshtein (1966) introduced the edit distance between two strings as the minimum number of elementary operations (insertions, deletions, and substitutions) needed to transform one string into the other

d(v,w) = MIN number of elementary operations

to transform $v \rightarrow w$

FACULDADE ENGENHARIA

Departamento de Informática

Edit Distance: Example 1

- So in our previous example, we can shift w one nucleotide to the right, and see that w is obtained from v by one insertion and one deletion:

v: ATATATAT-

w: -TATATATA

- Hence the edit distance, d(v, w) = 2.

- Note: In order to provide this distance, we had to "fiddle" with the sequences. Hamming distance was easier to find.

Departamento de Informática

Edit Distance: Example 2

- We can transform TGCATAT → ATCCGAT in 5 steps:

TGCATAT

Departamento de Informática

Edit Distance: Example 2

- We can transform TGCATAT → ATCCGAT in 5 steps:

TGCATAT (delete last T)

FACULDADE ENGENHARIA

Departamento de Informática

Edit Distance: Example 2

- We can transform TGCATAT → ATCCGAT in 5 steps:

TGCATAT	(delete last T)
TGCATA	(delete last A)

FACULDADE ENGENHARIA

Departamento de Informática

Edit Distance: Example 2

- We can transform TGCATAT → ATCCGAT in 5 steps:

TGCATAT	(delete last T)
TGCATA	(delete last A)
ATGCAT	(insert A at front)

FACULDADE ENGENHARIA
Departamento de Informática

Edit Distance: Example 2

- We can transform TGCATAT → ATCCGAT in 5 steps:

TGCATAT	(delete last T)
TGCATA	(delete last A)
ATGCAT	(insert A at front)
ATCCAT	(substitute C for G)

FACULDADE ENGENHARIA
Departamento de Informática

Edit Distance: Example 2

- We can transform TGCATAT → ATCCGAT in 5 steps:

TGCATAT	(delete last T)
TGCATA	(delete last A)
ATGCAT	(insert A at front)
ATCCAT	(substitute C for G)
ATCCGAT	(insert G before last A)

Edit Distance: Example 2

- We can transform TGCATAT → ATCCGAT in 5 steps:

TGCATAT	(delete last T)
TGCATA	(delete last A)
ATGCAT	(insert A at front)
ATCCAT	(substitute C for G)
ATCCGAT	(insert G before last A)

- **Note:** This *only* allows us to conclude that the edit distance is *at most 5*.

Edit Distance: Example 2

- Now we transform TGCATAT → ATCCGAT in 4 steps:

TGCATAT

Departamento de Informática

Edit Distance: Example 2

- Now we transform TGCATAT → ATCCGAT in 4 steps:

ATGCATAT (insert A at front)

Edit Distance: Example 2

- Now we transform TGCATAT → ATCCGAT in 4 steps:

ATGCATAT	(insert A at front)
ATGCATAT	(delete second T)

Departamento de Informática

Edit Distance: Example 2

- Now we transform TGCATAT → ATCCGAT in 4 steps:

ATGCATAT	(insert A at front)
ATGCATAT	(delete second T)
ATGCGAT	(substitute G for A)

Edit Distance: Example 2

- Now we transform TGCATAT → ATCCGAT in 4 steps:

ATGCATAT	(insert A at front)
ATGCATAT	(delete second T)
ATGCGAT	(substitute G for A)
ATCCGAT	(substitute C for G)

FACULDADE ENGENHARIA
Departamento de Informática

Edit Distance: Example 2

- Now we transform TGCATAT → ATCCGAT in 4 steps:

ATGCATAT	(insert A at front)
ATGCATAT	(delete second T)
ATGCGAT	(substitute G for A)
ATCCGAT	(substitute C for G)

- Can we do even better? 3 steps? 2 steps? How can we know?

Departamento de Informática

Key Result

- **Theorem:** Given two sequences **v** and **w** of length m and n, the edit distance d(**v**,**w**) is given by:

  $$d(\mathbf{v},\mathbf{w}) = m + n - s(\mathbf{v},\mathbf{w}),$$

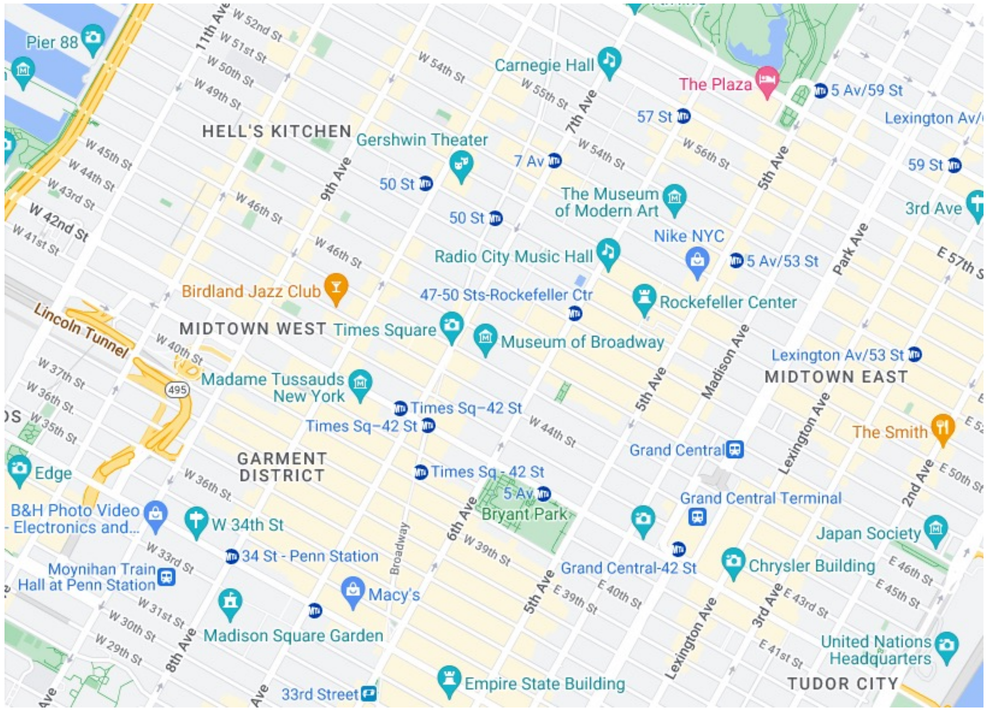
where s(**v**,**w**) is the length of the longest common subsequence of **v** and **w**.

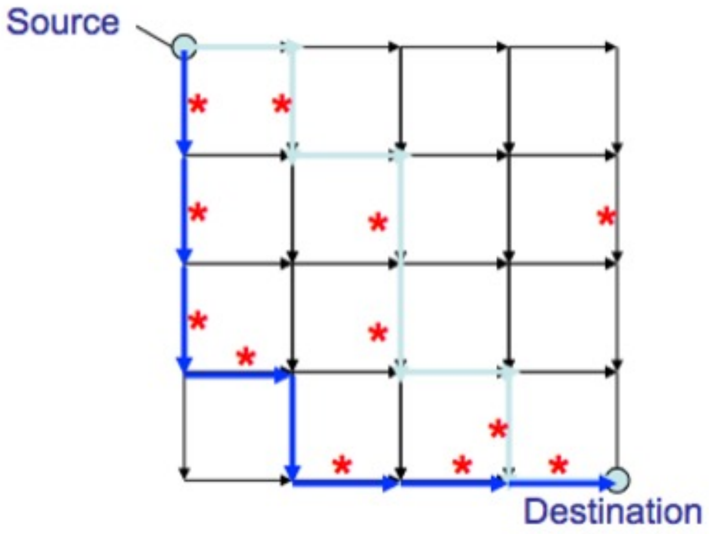
- Solving the the Longest Common Subsequence (LCS) problem for **v** and **w** is equivalent to finding the edit distance between them.

Manhattan Tourist Problem

- The **Manhattan Tourist Problem** is a classic problem in combinatorial optimization that is analogous to finding the longest path in a weighted grid, often used as a teaching example for understanding dynamic programming.
- The problem is inspired by the grid-like layout of many cities, such as Manhattan, where a tourist wishes to travel from the top-left corner to the bottom-right corner of the grid, moving only down or right, and wants to see as much as possible along the way. The goal is to find the path that covers the most "sights," where each "sight" corresponds to a score or weight on the grid.

Manhattan Tourist Problem



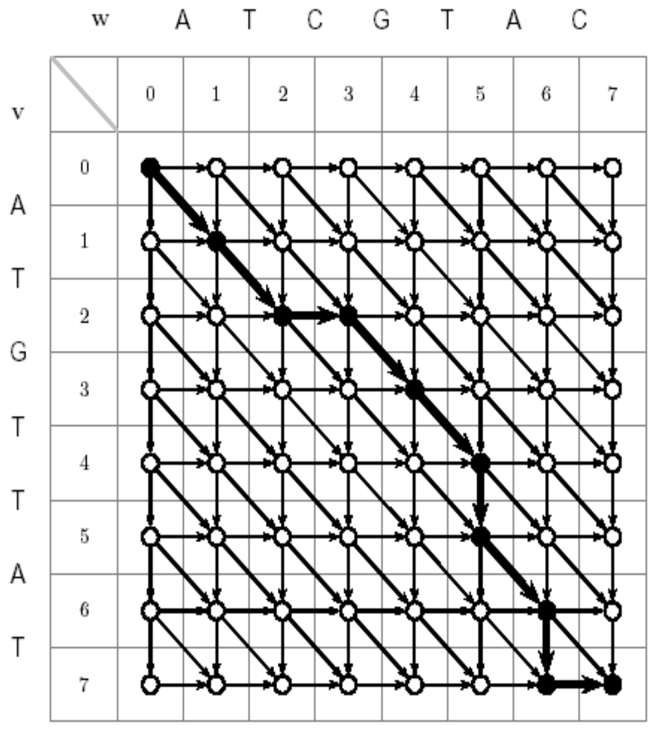


Departamento de Informática

Edit Graph

- Every alignment corresponds to a path from source to sink.

		0	1	2	2	3	4	5	6	7	7
v	=		A	T	-	G	T	T	A	T	-
			|	|		|	|		|		
w	=		A	T	C	G	T	-	A	-	C
		0	1	2	3	4	5	5	6	6	7



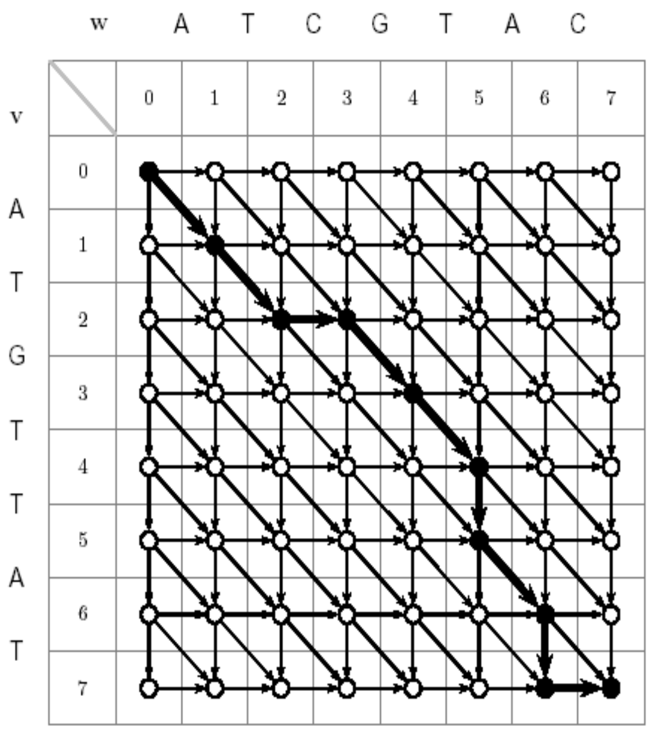
↘	↘	→	↘	↘	↓	↘	↓	→
A	T	-	G	T	T	A	T	-
A	T	C	G	T	-	A	-	C

Edit Graph

- Every alignment corresponds to a path from source to sink.

- Horizontal and vertical edges correspond to *indels* (deletions and insertions).

```
     0  1  2  2  3  4  5  6  7  7
v =     A  T  -  G  T  T  A  T  -
        |  |     |  |     |
w =     A  T  C  G  T  -  A  -  C
     0  1  2  3  4  5  5  6  6  7
```



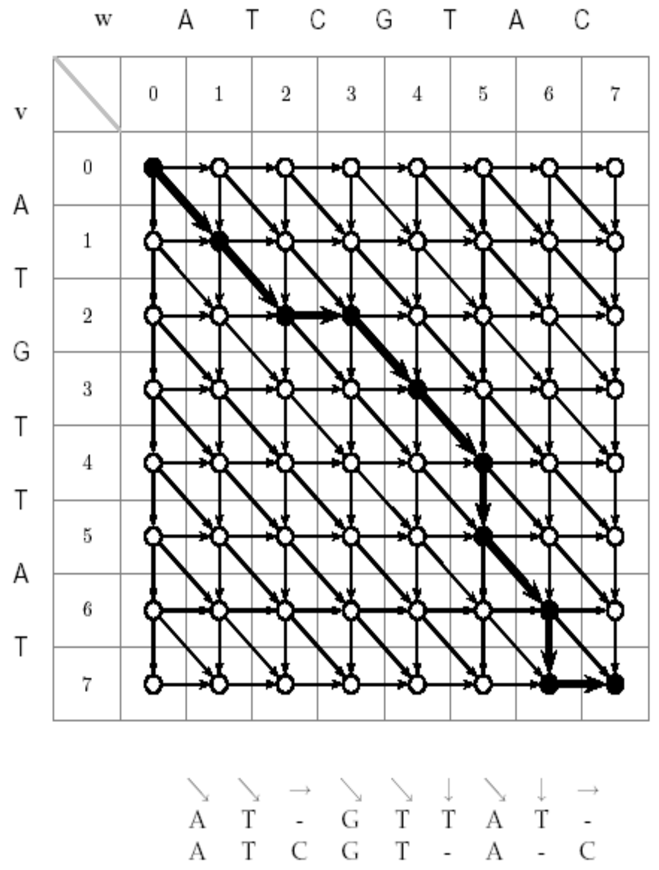
```
↘  ↘  →  ↘  ↘  ↓  ↘  ↓  →
A  T  -  G  T  T  A  T  -
A  T  C  G  T  -  A  -  C
```

Departamento de Informática

Edit Graph

- Every alignment corresponds to a path from source to sink.
- Horizontal and vertical edges correspond to *indels* (deletions and insertions).
- Diagonal edges correspond to matches.

```
         0  1  2  2  3  4  5  6  7  7
v  =        A  T  -  G  T  T  A  T  -
            |  |     |  |     |
w  =        A  T  C  G  T  -  A  -  C
         0  1  2  3  4  5  5  6  6  7
```



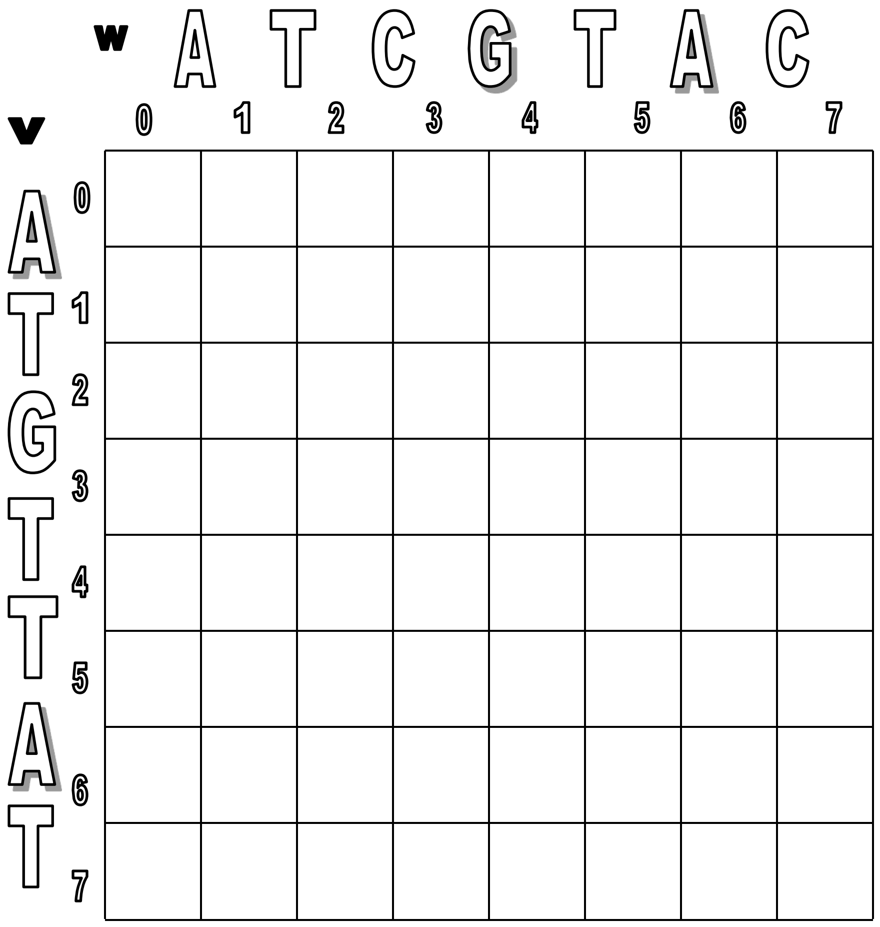
Departamento de Informática

Alignment as a Path in the Edit Graph: Example

- Suppose that our sequences are ATCGTAC, ATGTTAT.
- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:



FACULDADE ENGENHARIA

Departamento de Informática

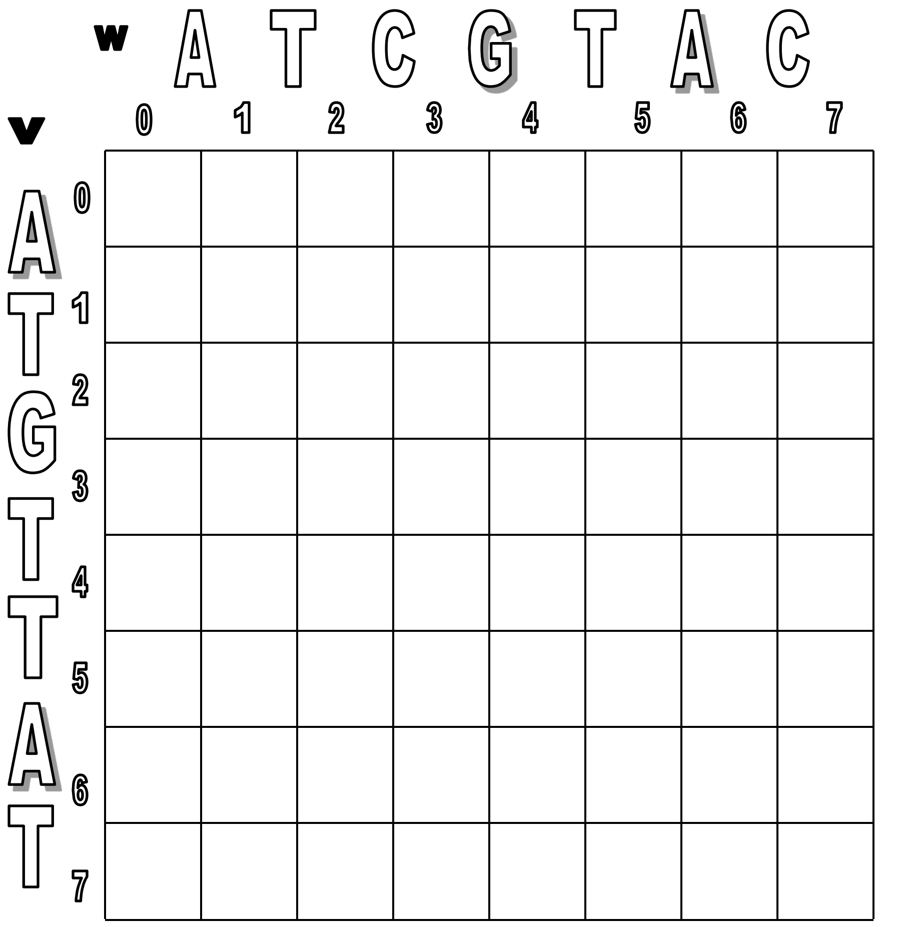
Alignment as a Path in the Edit Graph: Example

- Suppose that our sequences are ATCGTAC, ATGTTAT.
- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)



Departamento de Informática

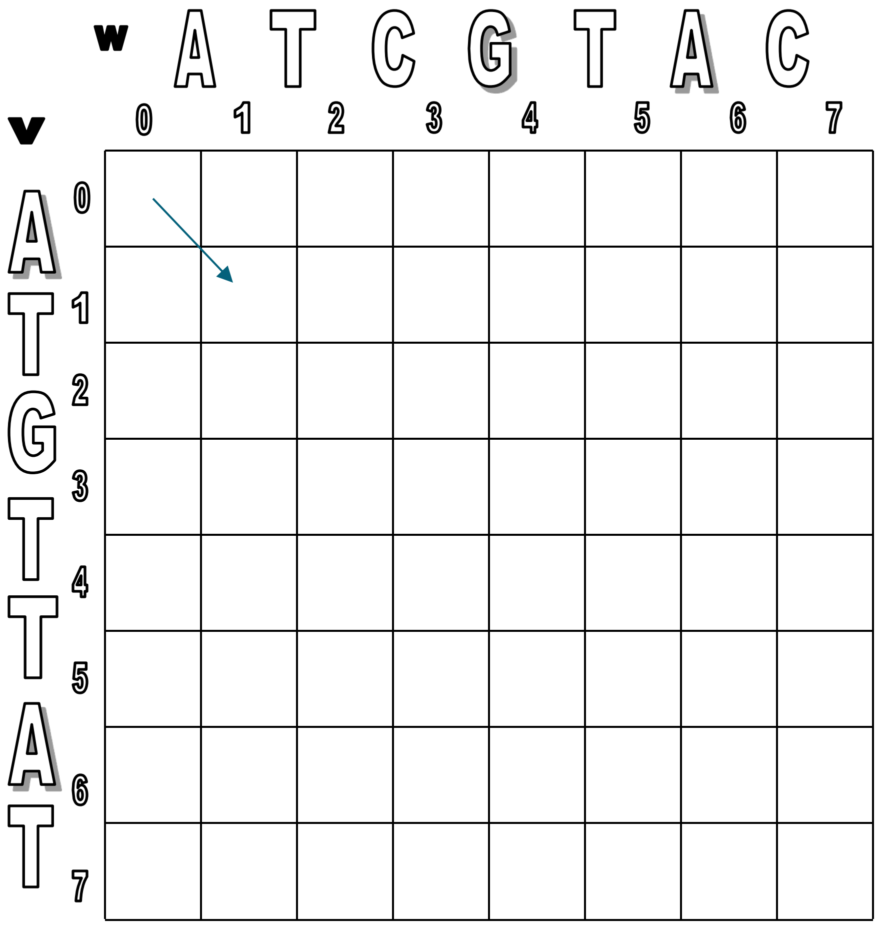
Alignment as a Path in the Edit Graph: Example

- Suppose that our sequences are ATCGTAC, ATGTTAT.
- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)→(1,1)



Departamento de Informática

Alignment as a Path in the Edit Graph: Example

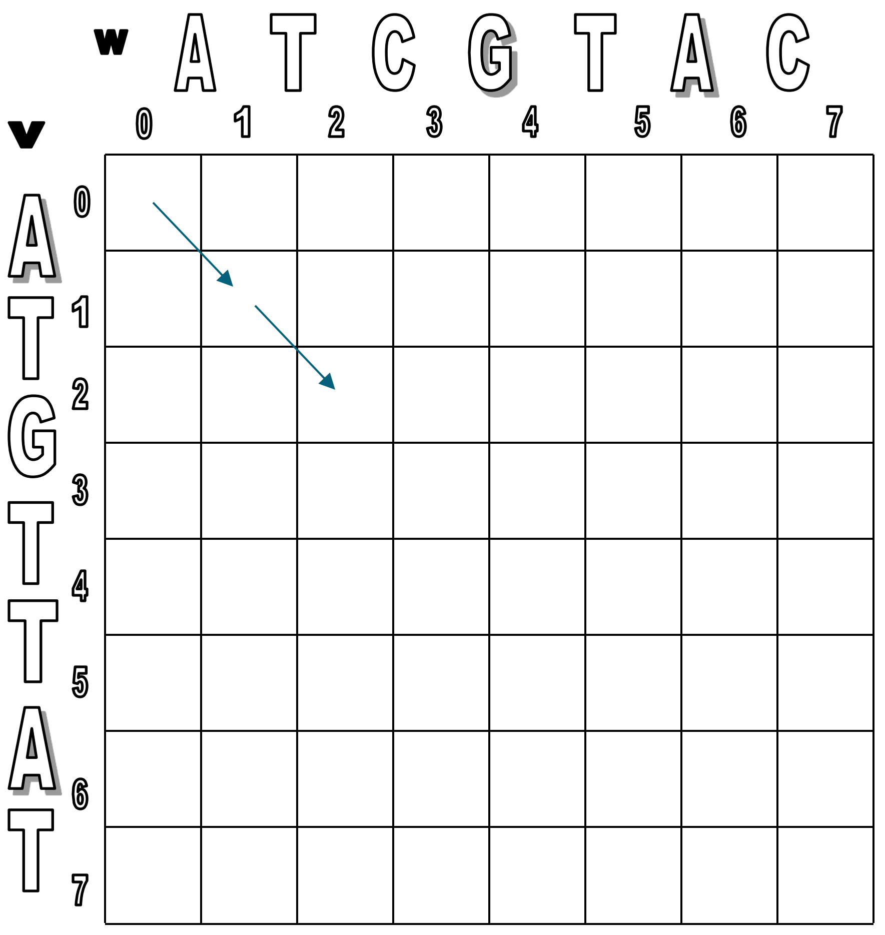
- Suppose that our sequences are ATCGTAC, ATGTTAT.

- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)→(1,1)→(2,2)



Alignment as a Path in the Edit Graph: Example

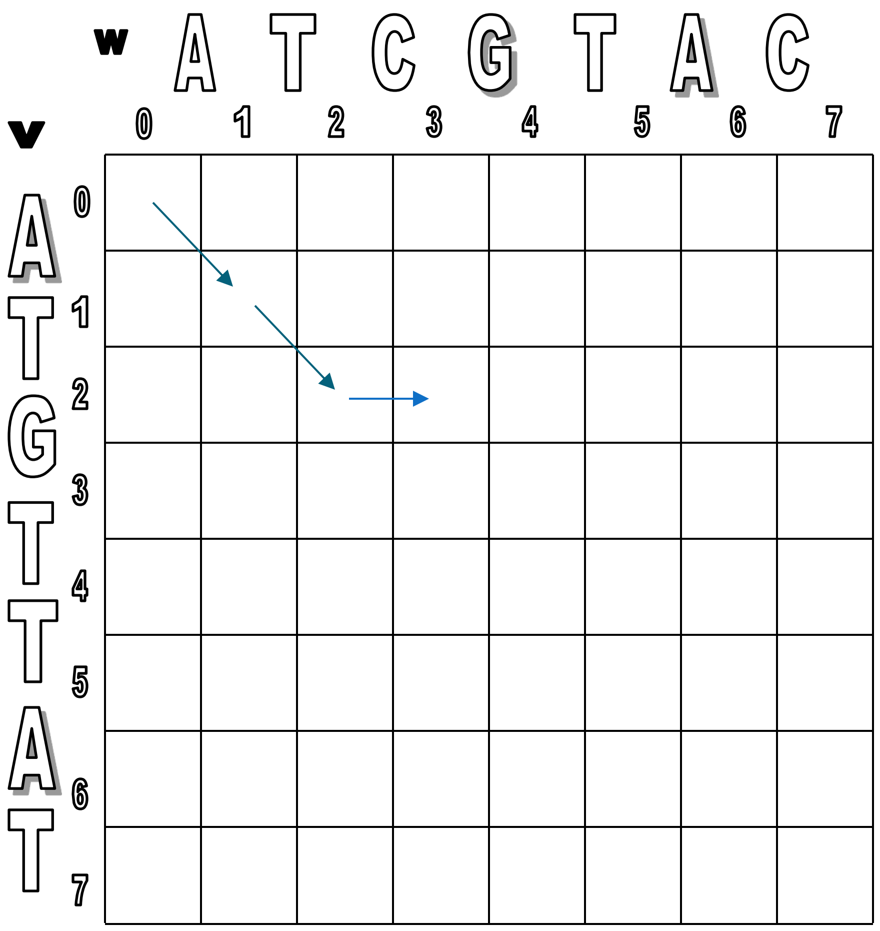
- Suppose that our sequences are ATCGTAC, ATGTTAT.

- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)→(1,1)→(2,2)→ (2,3)



Departamento de Informática

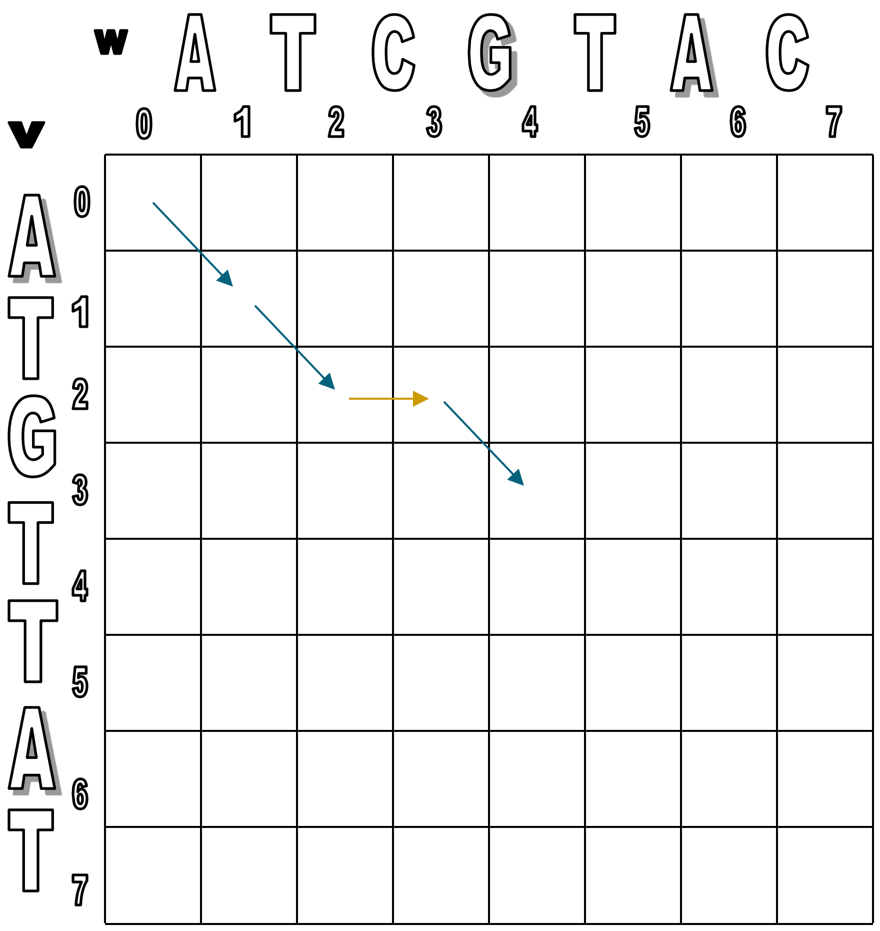
Alignment as a Path in the Edit Graph: Example

- Suppose that our sequences are ATCGTAC, ATGTTAT.
- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)→(1,1)→(2,2)→(2,3)→(3,4)



Departamento de Informática

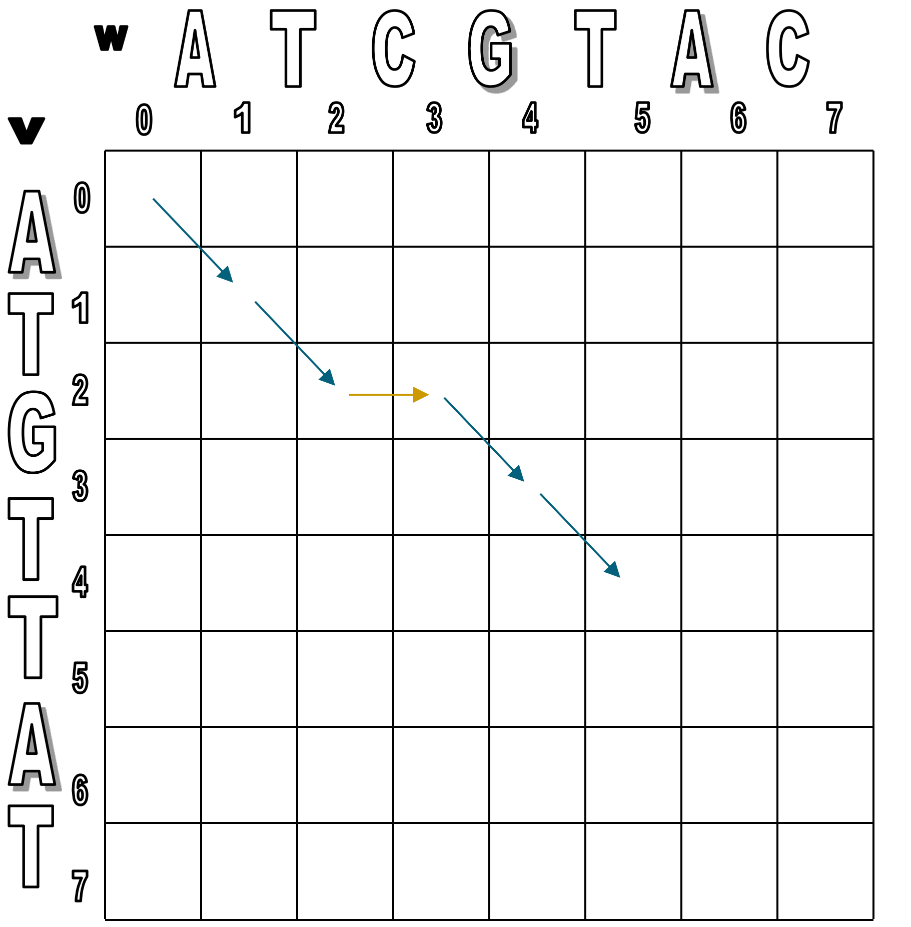
Alignment as a Path in the Edit Graph: Example

- Suppose that our sequences are ATCGTAC, ATGTTAT.
- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)→(1,1)→(2,2)→ (2,3)→(3,4)→etc.



FACULDADE ENGENHARIA

Departamento de Informática

Alignment as a Path in the Edit Graph: Example

- Suppose that our sequences are ATCGTAC, ATGTTAT.

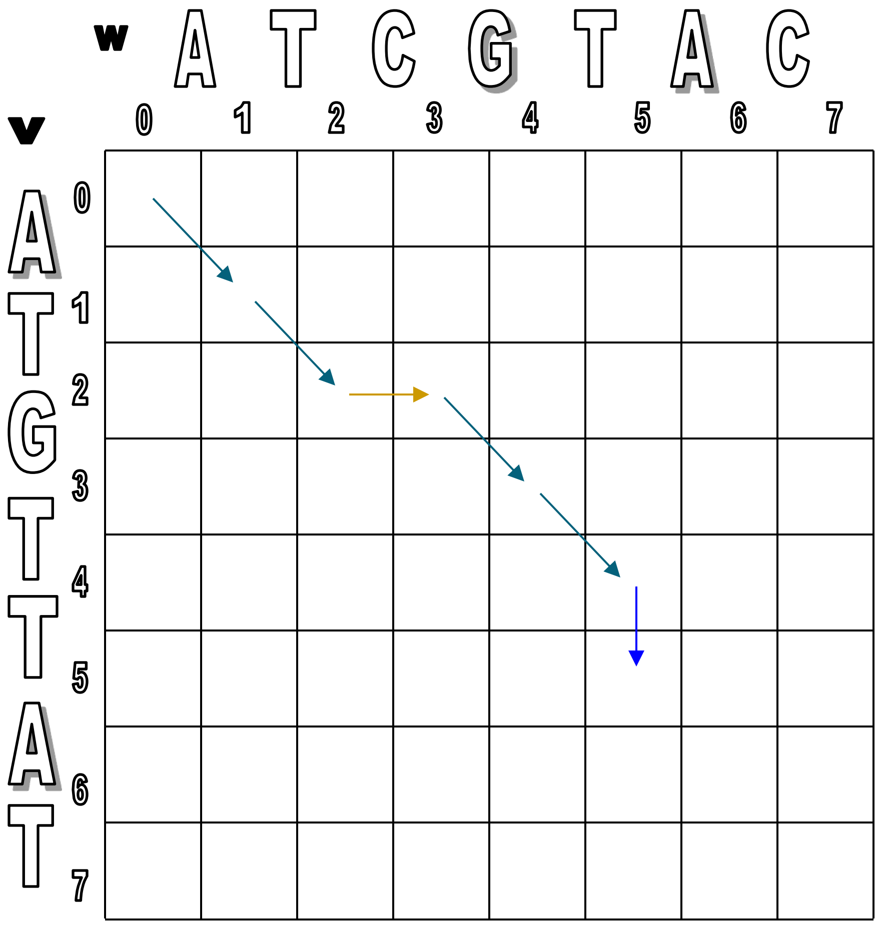
- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)→(1,1)→(2,2)→

(2,3)→(3,4)→etc.



Departamento de Informática

Alignment as a Path in the Edit Graph: Example

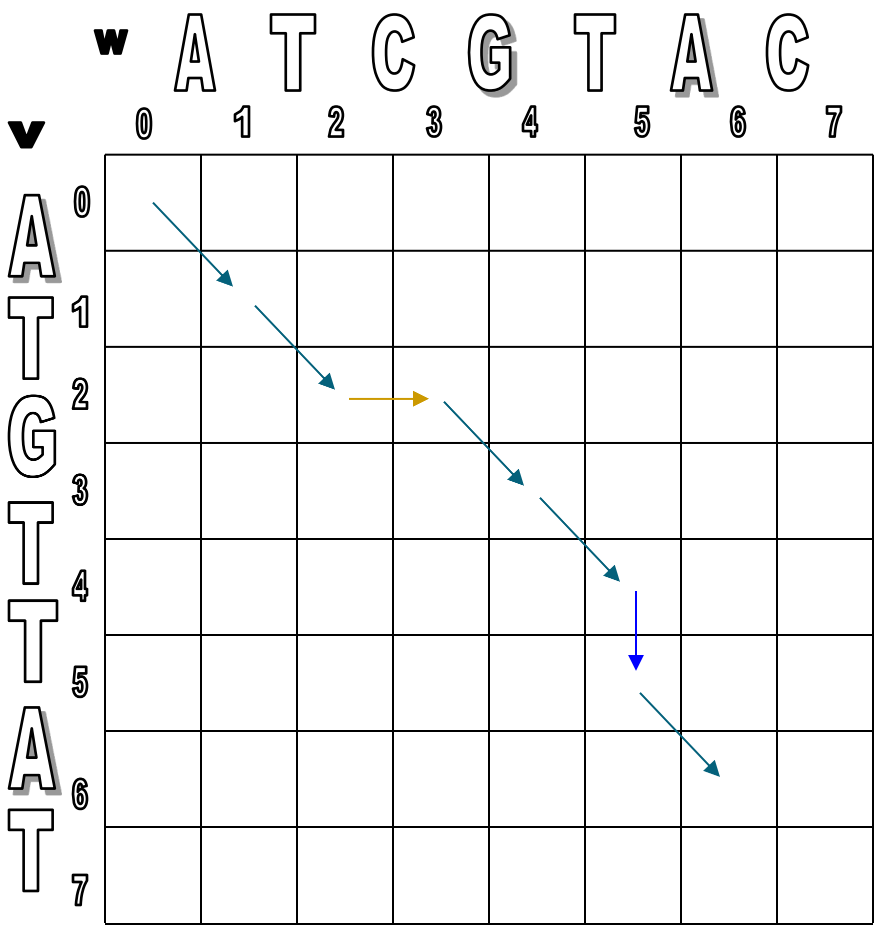
- Suppose that our sequences are ATCGTAC, ATGTTAT.

- One possible alignment is:

```
0 1 2 2 3 4 5 6 7 7
  A T _ G T T A T _
  A T C G T _ A _ C
0 1 2 3 4 5 5 6 6 7
```

- Edit graph path:

(0,0)→(1,1)→(2,2)→ (2,3)→(3,4)→etc.



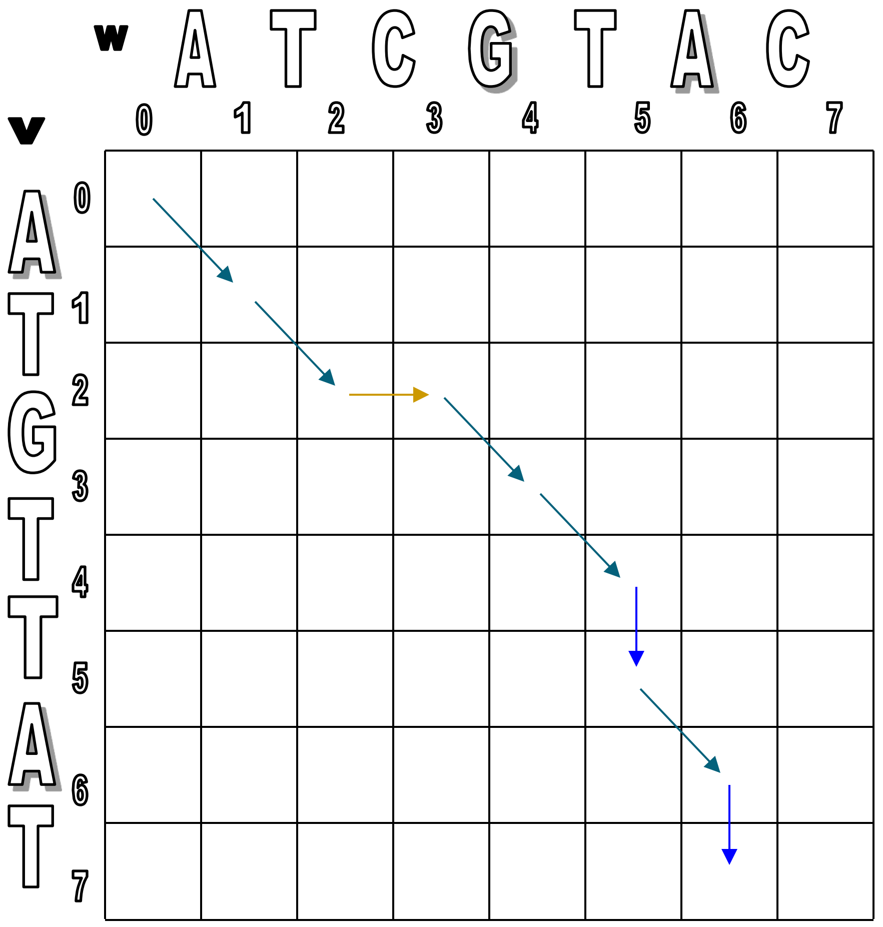
Alignment as a Path in the Edit Graph: Example

- Suppose that our sequences are ATCGTAC, ATGTTAT.
- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)→(1,1)→(2,2)→(2,3)→(3,4)→etc.



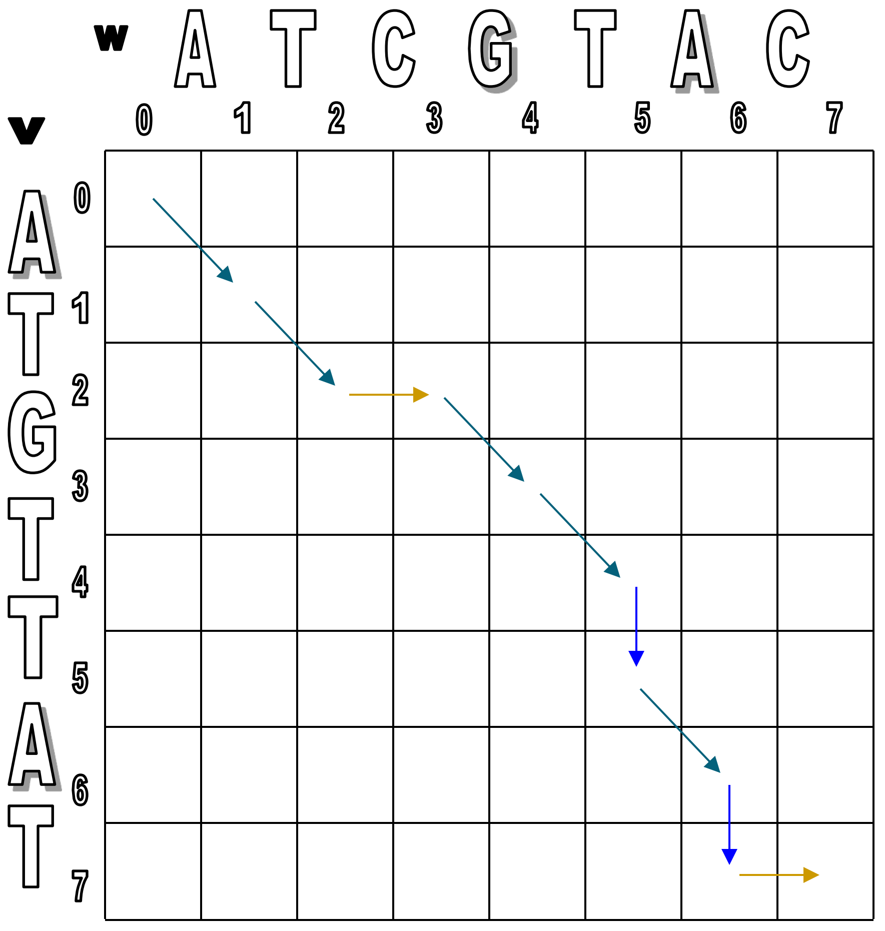
Alignment as a Path in the Edit Graph: Example

- Suppose that our sequences are ATCGTAC, ATGTTAT.
- One possible alignment is:

0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- Edit graph path:

(0,0)→(1,1)→(2,2)→(2,3)→(3,4)→etc.



FACULDADE ENGENHARIA

Departamento de Informática

Alignment with Dynamic Programming

- Initialize 0th row and 0th column to be all zeroes.

v \ w			A	T	C	G	T	A	C
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
A	1	0							
T	2	0							
G	3	0							
T	4	0							
T	5	0							
A	6	0							
T	7	0							

Departamento de Informática

Alignment with Dynamic Programming

- Initialize 0th row and 0th column to be all zeroes.

- Use the following recursive formula to calculate $s_{i,j}$ for each i, j:

$$S_{i,j} = \max \begin{cases} S_{i-1,\,j-1} + 1 & \nwarrow \\ S_{i-1,\,j} & \uparrow \\ S_{i,\,j-1} & \leftarrow \end{cases}$$

		w	A	T	C	G	T	A	C
v		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
A	1	0	↖ 1	← 1	← 1	← 1	← 1	↖ 1	← 1
T	2	0	↑ 1						
G	3	0	↑ 1						
T	4	0	↑ 1						
T	5	0	↑ 1						
A	6	0	↖ 1						
T	7	0	↑ 1						

Departamento de Informática

Dynamic Programming Example

- Note that the placement of the arrows shows where a given score originated from:
  - $\uparrow$ if from the top
  - $\leftarrow$ if from the left
  - $\nwarrow$ if $v_i = w_j$

FACULDADE ENGENHARIA
Departamento de Informática

Dynamic Programming Example

- Continuing with the dynamic programming algorithm fills the table.

	w		A	T	C	G	T	A	C
v		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
A	1	0	↖ 1	← 1	← 1	← 1	← 1	↖ 1	← 1
T	2	0	↑ 1						
G	3	0	↑ 1						
T	4	0	↑ 1						
T	5	0	↑ 1						
A	6	0	↖ 1						
T	7	0	↑ 1						

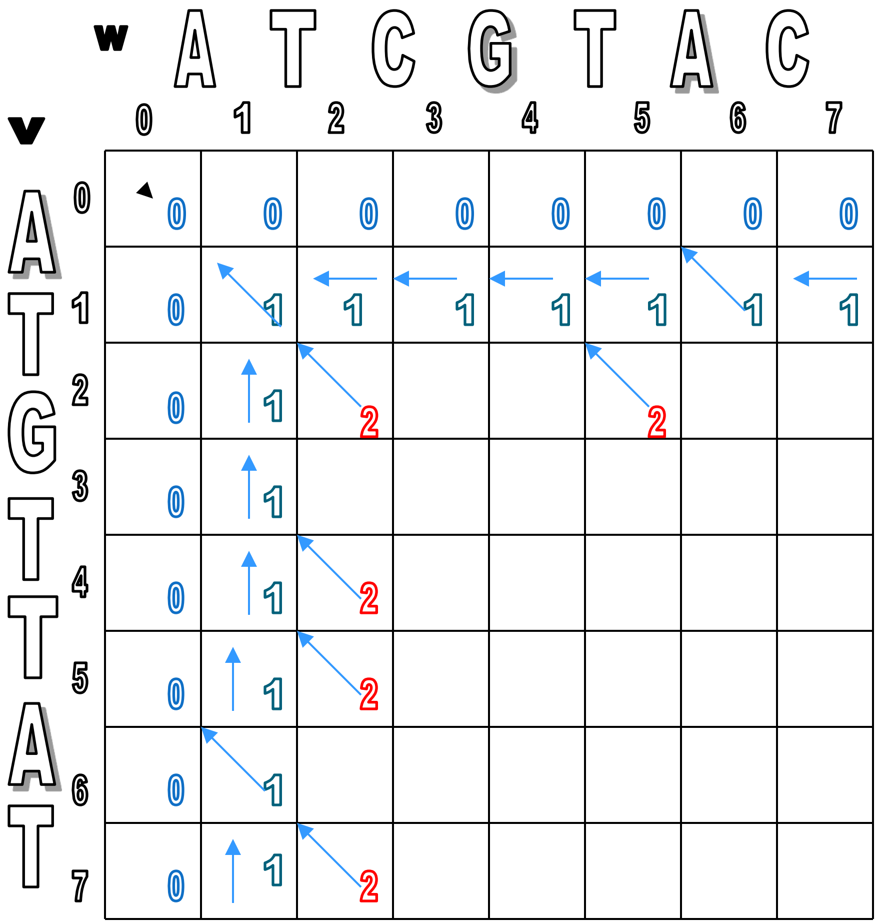
FACULDADE ENGENHARIA

Departamento de Informática

Dynamic Programming Example

- Continuing with the dynamic programming algorithm fills the table.

- We first look for matches, highlighted in red, and then we fill in the rest of that row and column.



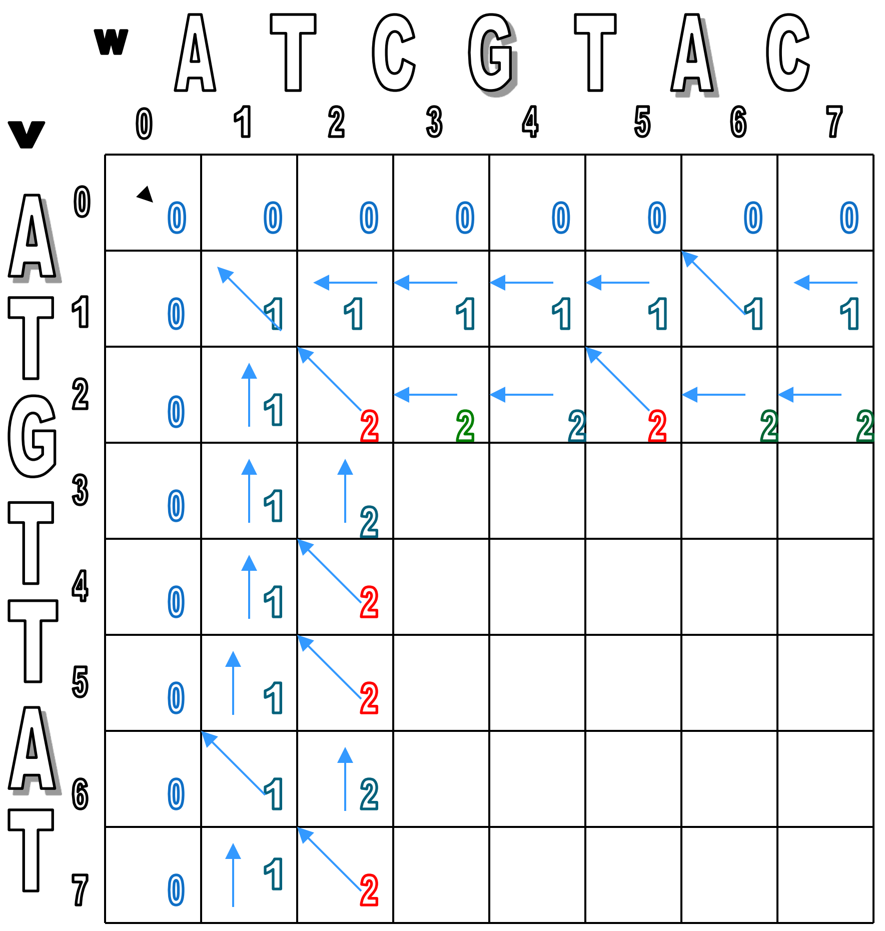
v \ w	0	1 A	2 T	3 C	4 G	5 T	6 A	7 C
0	0	0	0	0	0	0	0	0
1 A	0	↖ 1	← 1	← 1	← 1	← 1	↖ 1	← 1
2 T	0	↑ 1	↖ 2			↖ 2		
3 G	0	↑ 1						
4 T	0	↑ 1	↖ 2					
5 T	0	↑ 1	↖ 2					
6 A	0	↖ 1						
7 T	0	↑ 1	↖ 2					

Dynamic Programming Example

Departamento de Informática

- Continuing with the dynamic programming algorithm fills the table.

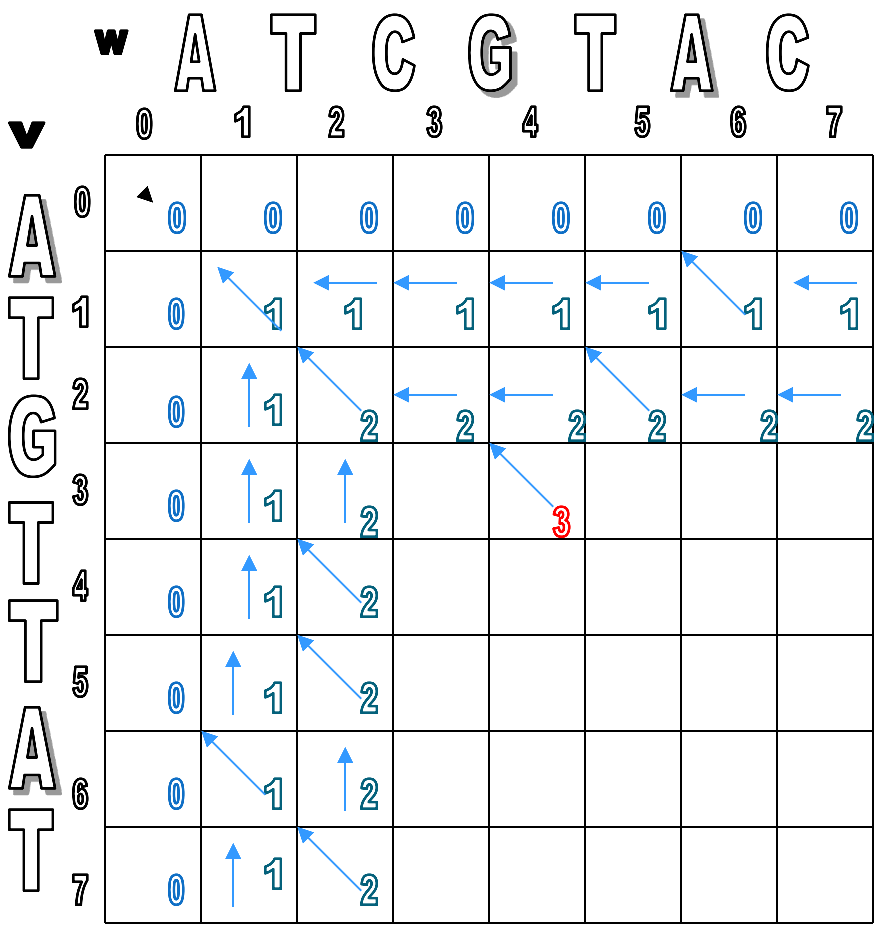
- We first look for matches, highlighted in red, and then we fill in the rest of that row and column.



v \ w	0	1 A	2 T	3 C	4 G	5 T	6 A	7 C
0	0	0	0	0	0	0	0	0
1 A	0	↖1	←1	←1	←1	←1	↖1	←1
2 T	0	↑1	↖2	←2	←2	↖2	←2	←2
3 G	0	↑1	↑2					
4 T	0	↑1	↖2					
5 T	0	↑1	↖2					
6 A	0	↖1	↑2					
7 T	0	↑1	↖2					

Dynamic Programming Example

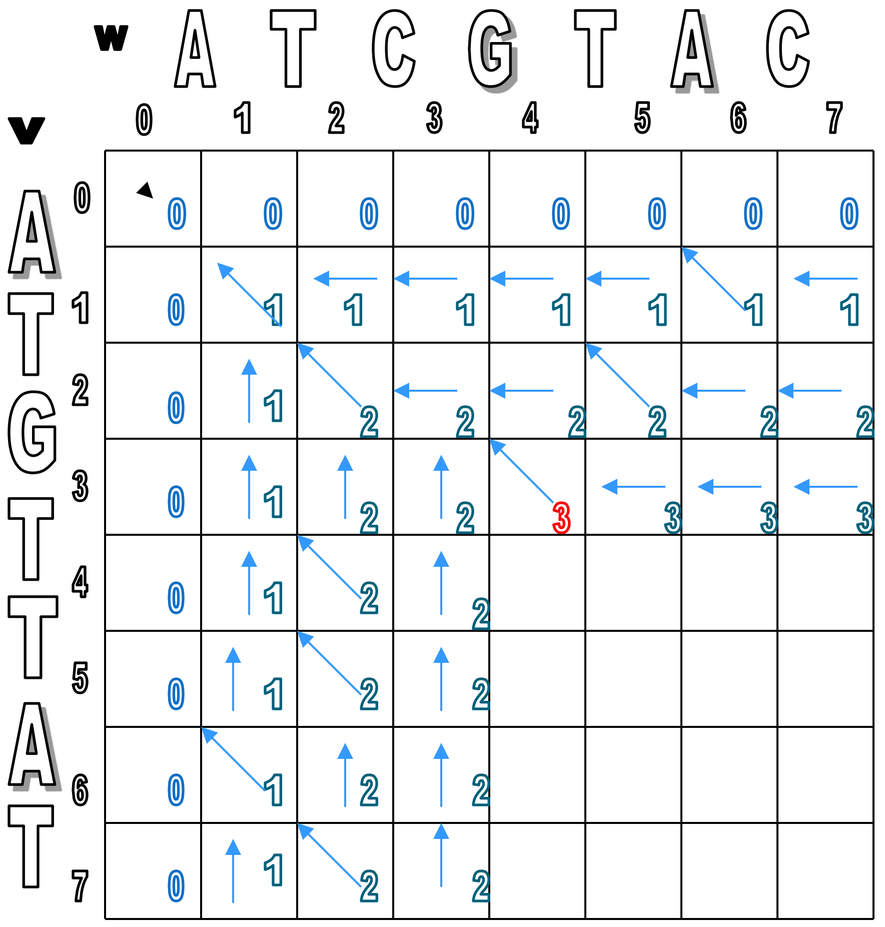
- Continuing with the dynamic programming algorithm fills the table.
- We first look for matches, highlighted in red, and then we fill in the rest of that row and column.
- As we can see, we do not simply add 1 each time.



v \ w	0	1 A	2 T	3 C	4 G	5 T	6 A	7 C
0	↖ 0	0	0	0	0	0	0	0
1 A	0	↖ 1	← 1	← 1	← 1	← 1	↖ 1	← 1
2 T	0	↑ 1	↖ 2	← 2	← 2	↖ 2	← 2	← 2
3 G	0	↑ 1	↑ 2		↖ 3			
4 T	0	↑ 1	↖ 2					
5 T	0	↑ 1	↖ 2					
6 A	0	↖ 1	↑ 2					
7 T	0	↑ 1	↖ 2					

Dynamic Programming Example

- Continuing with the dynamic programming algorithm fills the table.
- We first look for matches, highlighted in red, and then we fill in the rest of that row and column.
- As we can see, we do not simply add 1 each time.



	w		A	T	C	G	T	A	C
v		0	1	2	3	4	5	6	7
A	0	0	0	0	0	0	0	0	0
T	1	0	1	1	1	1	1	1	1
G	2	0	1	2	2	2	2	2	2
T	3	0	1	2	2	3	3	3	3
T	4	0	1	2	2				
A	5	0	1	2	2				
T	6	0	1	2	2				
	7	0	1	2	2				

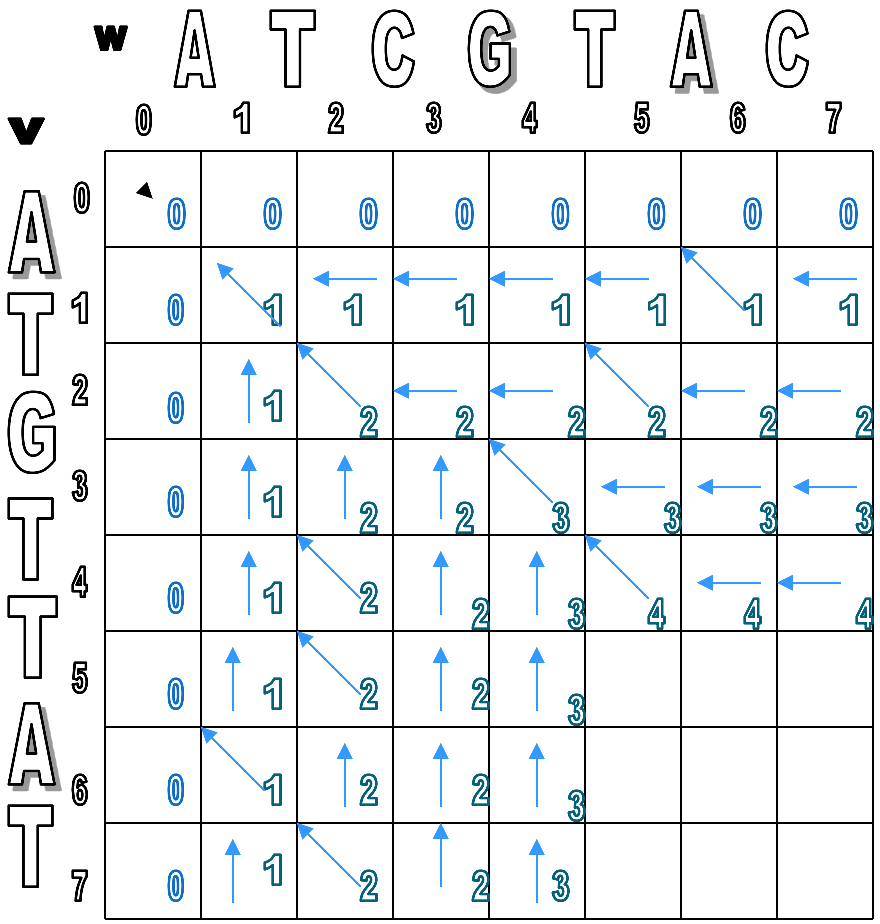
Departamento de Informática

Dynamic Programming Example

- Continuing with the dynamic programming algorithm fills the table.

- We first look for matches, highlighted in red, and then we fill in the rest of that row and column.

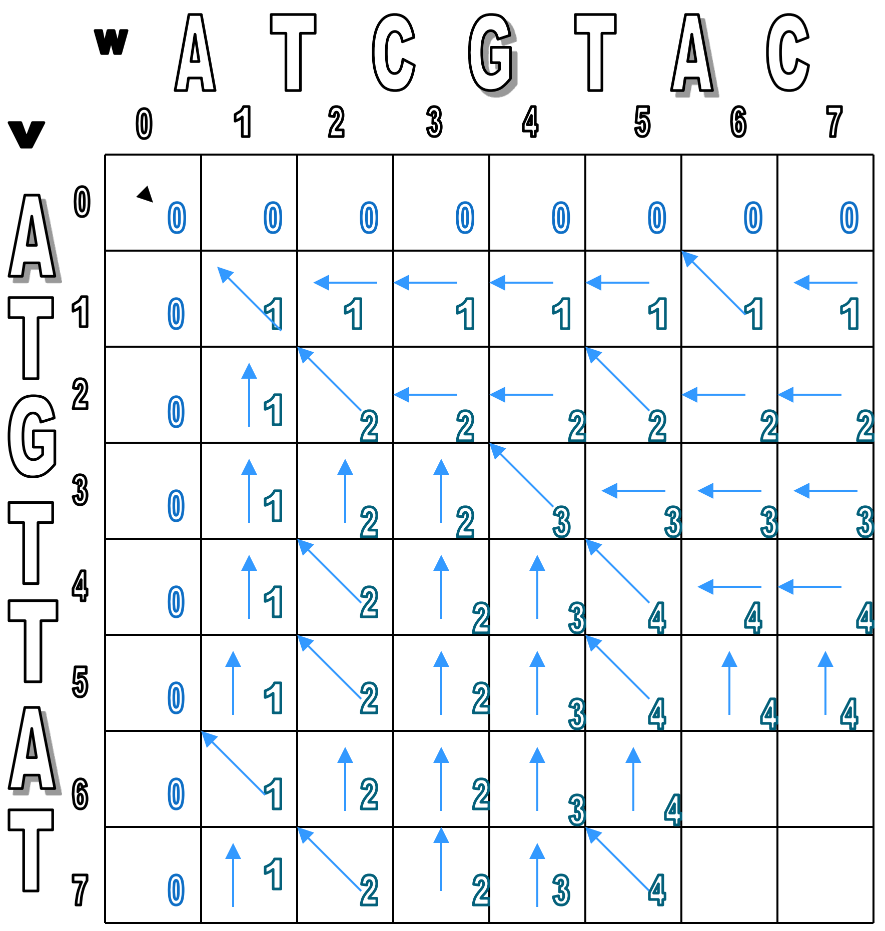
- As we can see, we do not simply add 1 each time.



	w		A	T	C	G	T	A	C
v		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
A	1	0	↖1	←1	←1	←1	←1	↖1	←1
T	2	0	↑1	↖2	←2	←2	↖2	←2	←2
G	3	0	↑1	↑2	↑2	↖3	←3	←3	←3
T	4	0	↑1	↖2	↑2	↑3	↖4	←4	←4
T	5	0	↑1	↖2	↑2	↑3			
A	6	0	↖1	↑2	↑2	↑3			
T	7	0	↑1	↖2	↑2	↑3			

Dynamic Programming Example

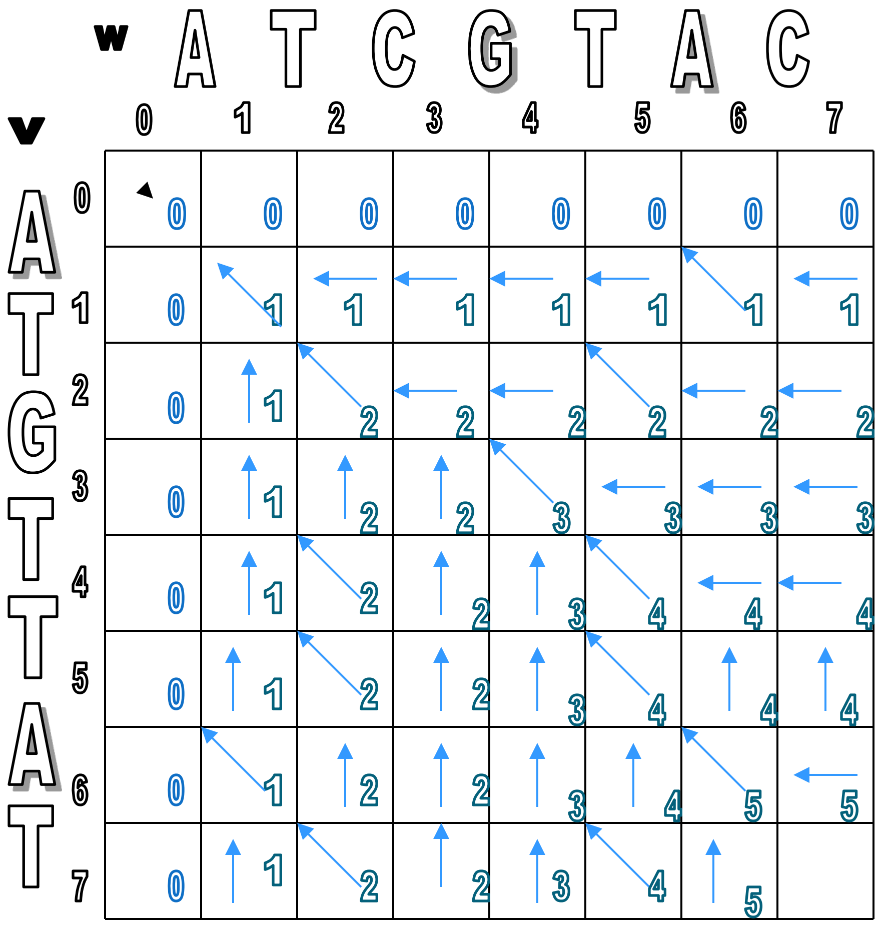
- Continuing with the dynamic programming algorithm fills the table.
- We first look for matches, highlighted in red, and then we fill in the rest of that row and column.
- As we can see, we do not simply add 1 each time.



	w	A	T	C	G	T	A	C
v	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
A 1	0	↖1	←1	←1	←1	←1	↖1	←1
T 2	0	↑1	↖2	←2	←2	↖2	←2	←2
G 3	0	↑1	↑2	↑2	↖3	←3	←3	←3
T 4	0	↑1	↖2	↑2	↑3	↖4	←4	←4
T 5	0	↑1	↖2	↑2	↑3	↖4	↑4	↑4
A 6	0	↖1	↑2	↑2	↑3	↑4		
T 7	0	↑1	↖2	↑2	↑3	↖4		

Dynamic Programming Example

- Continuing with the dynamic programming algorithm fills the table.
- We first look for matches, highlighted in red, and then we fill in the rest of that row and column.
- As we can see, we do not simply add 1 each time.

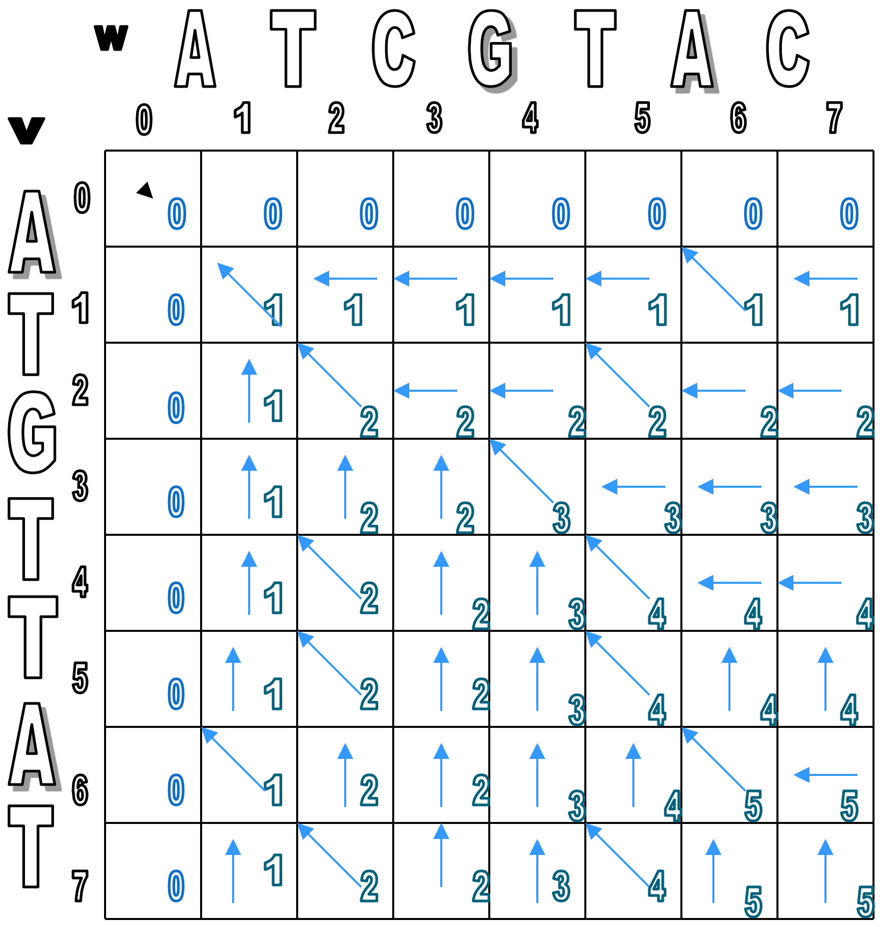


v \ w		A	T	C	G	T	A	C
	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
A 1	0	↖1	←1	←1	←1	←1	↖1	←1
T 2	0	↑1	↖2	←2	←2	↖2	←2	←2
G 3	0	↑1	↑2	↑2	↖3	←3	←3	←3
T 4	0	↑1	↖2	↑2	↑3	↖4	←4	←4
T 5	0	↑1	↖2	↑2	↑3	↖4	↑4	↑4
A 6	0	↖1	↑2	↑2	↑3	↑4	↖5	←5
T 7	0	↑1	↖2	↑2	↑3	↖4	↑5	

Departamento de Informática

Dynamic Programming Example

- Continuing with the dynamic programming algorithm fills the table.
- We first look for matches, highlighted in red, and then we fill in the rest of that row and column.
- As we can see, we do not simply add 1 each time.



v	w		A	T	C	G	T	A	C
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
A	1	0	↖ 1	← 1	← 1	← 1	← 1	↖ 1	← 1
T	2	0	↑ 1	↖ 2	← 2	← 2	↖ 2	← 2	← 2
G	3	0	↑ 1	↑ 2	↑ 2	↖ 3	← 3	← 3	← 3
T	4	0	↑ 1	↖ 2	↑ 2	↑ 3	↖ 4	← 4	← 4
T	5	0	↑ 1	↖ 2	↑ 2	↑ 3	↖ 4	↑ 4	↑ 4
A	6	0	↖ 1	↑ 2	↑ 2	↑ 3	↑ 4	↖ 5	← 5
T	7	0	↑ 1	↖ 2	↑ 2	↑ 3	↖ 4	↑ 5	↑ 5

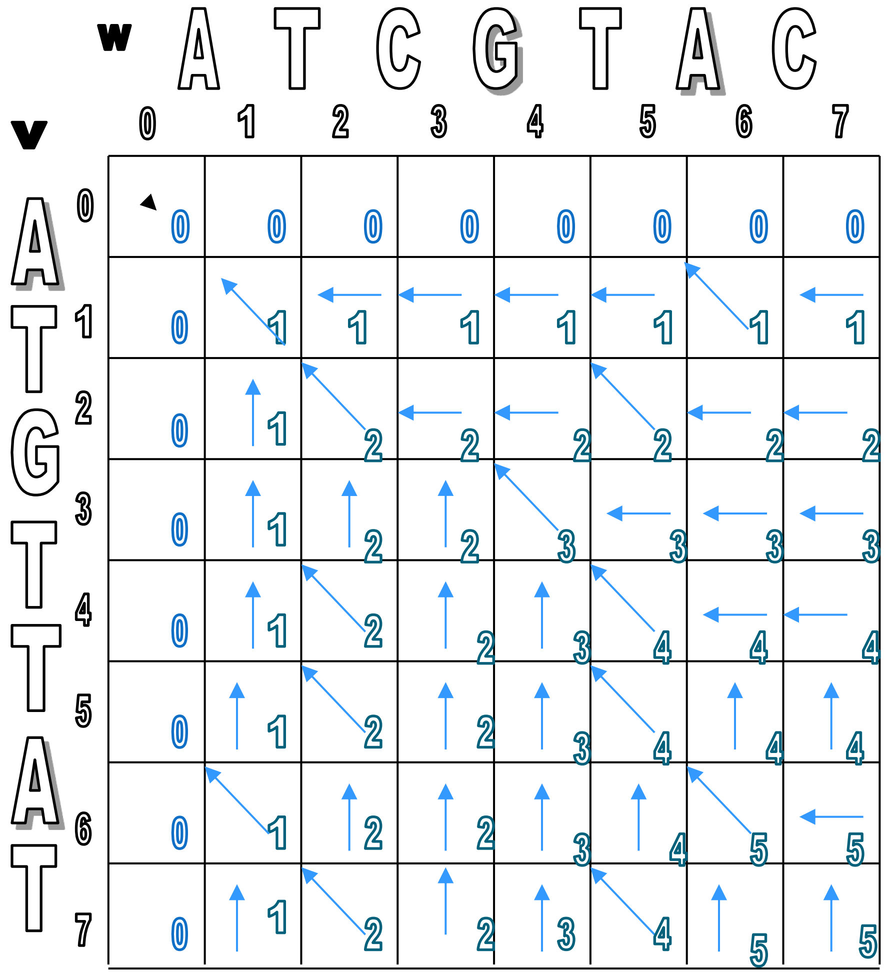
Departamento de Informática

Dynamic Alignment: Pseudocode

```
1. LCS(v,w)
2.    for $i \leftarrow 1$ to $n$
3.       $s_{i,0} \leftarrow 0$
4.    for $j \leftarrow 1$ to $m$
5.       $s_{0,j} \leftarrow 0$
6.    for $i \leftarrow 1$ to $n$
7.       for $j \leftarrow 1$ to $m$
8.
9.          $s_{i,j} \leftarrow \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1} + 1, \text{ if } v_i = w_j \end{cases}$
10.
11.
•           $b_{i,j} \leftarrow \begin{cases} \text{“} \uparrow \text{“} & \text{if } s_{i,j} = s_{i-1,j} \\ \text{“} \leftarrow \text{“} & \text{if } s_{i,j} = s_{i,j-1} \\ \text{“} \nwarrow \text{“} & \text{if } s_{i,j} = s_{i-1,j-1} + 1 \end{cases}$
•
•     return ($s_{n,m}$, $b$)
```

Finding an Optimal Alignment

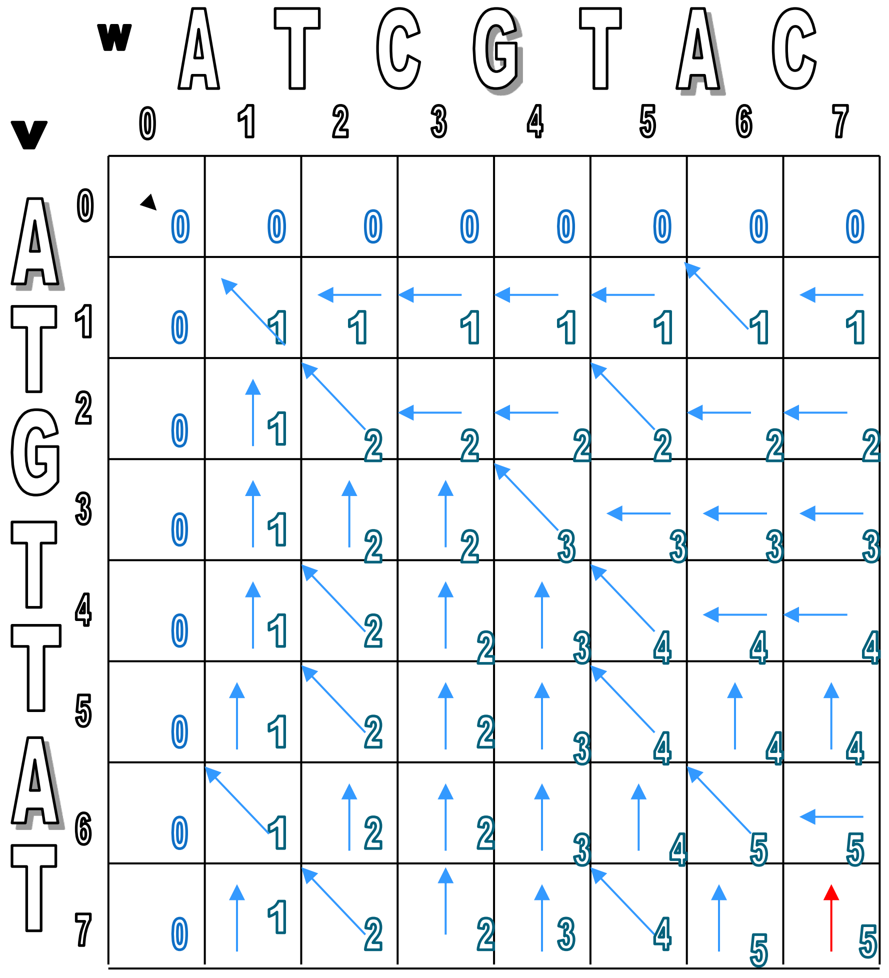
- LCS(v,w) created the alignment grid.
- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:



Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.
- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

7
T
_
7



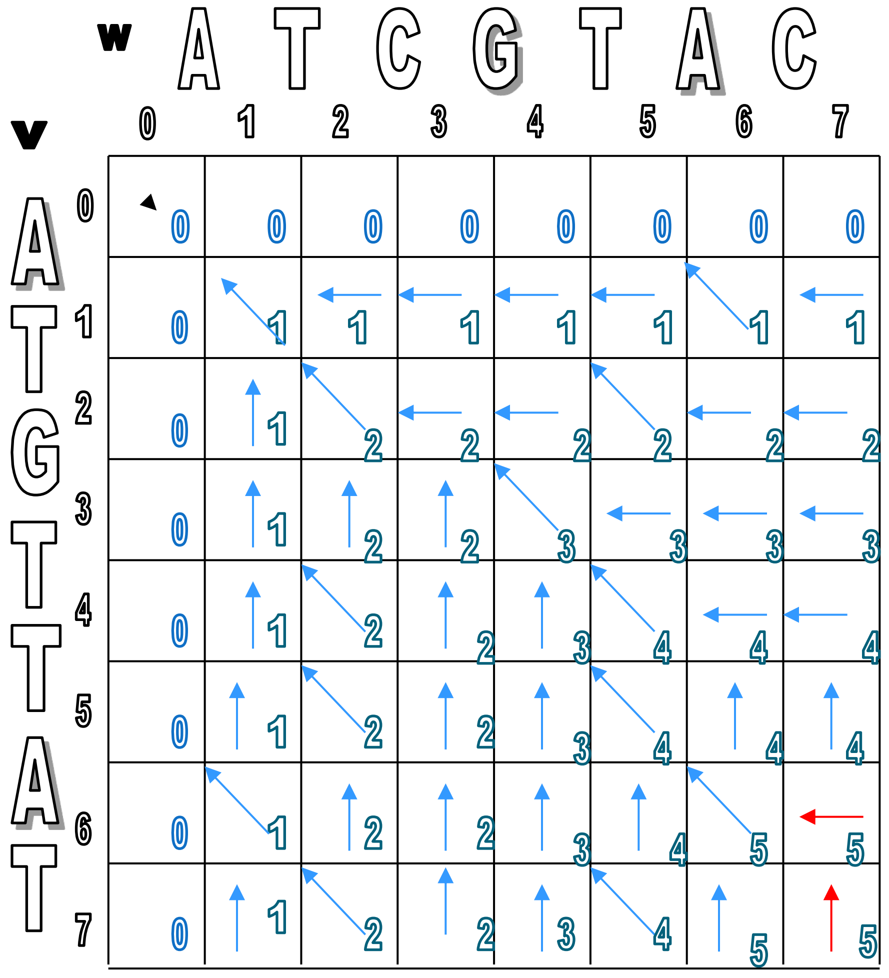
Departamento de Informática

FACULDADE ENGENHARIA

Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.
- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

6	7
_	T
C	_
7	7



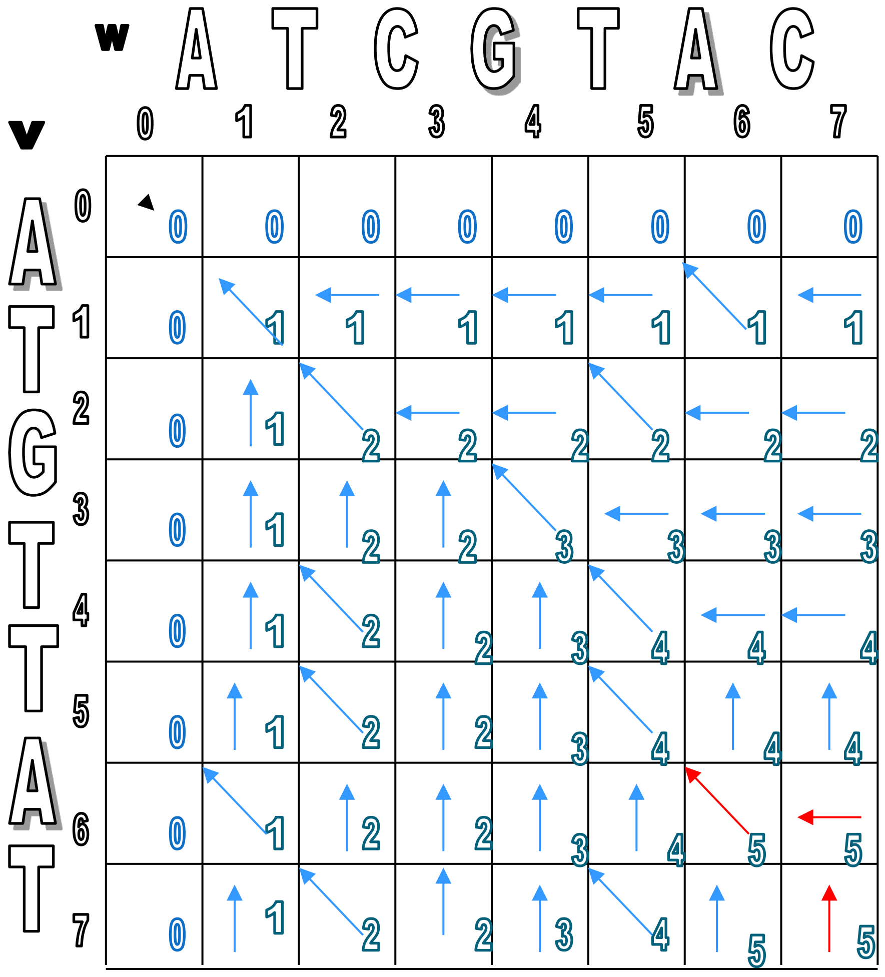
v \ w		A	T	C	G	T	A	C
	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
A 1	0	↖ 1	← 1	← 1	← 1	← 1	↖ 1	← 1
T 2	0	↑ 1	↖ 2	← 2	← 2	↖ 2	← 2	← 2
G 3	0	↑ 1	↑ 2	↑ 2	↖ 3	← 3	← 3	← 3
T 4	0	↑ 1	↖ 2	↑ 2	↑ 3	↖ 4	← 4	← 4
T 5	0	↑ 1	↖ 2	↑ 2	↑ 3	↖ 4	↑ 4	↑ 4
A 6	0	↖ 1	↑ 2	↑ 2	↑ 3	↑ 4	↖ 5	← 5
T 7	0	↑ 1	↖ 2	↑ 2	↑ 3	↖ 4	↑ 5	↑ 5

Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.

- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

6	6	7
A	_	T
A	C	_
6	7	7



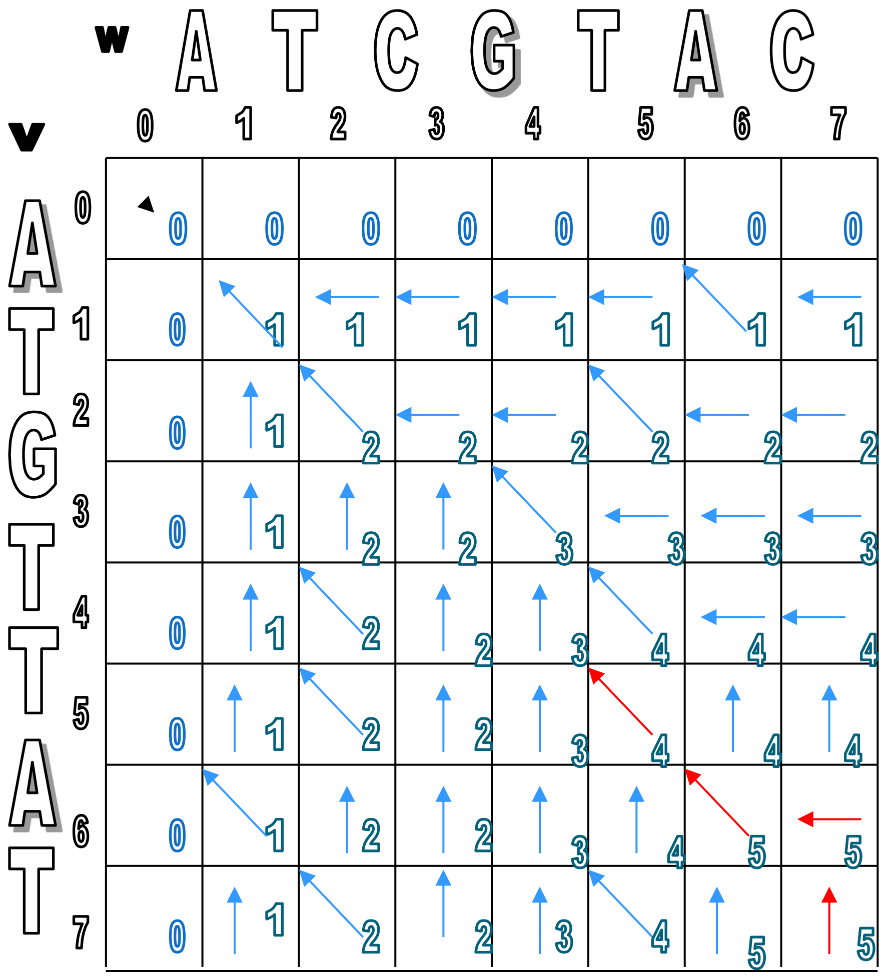
	w		A	T	C	G	T	A	C
v		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
A	1	0	1	1	1	1	1	1	1
T	2	0	1	2	2	2	2	2	2
G	3	0	1	2	2	3	3	3	3
T	4	0	1	2	2	3	4	4	4
T	5	0	1	2	2	3	4	4	4
A	6	0	1	2	2	3	4	5	5
T	7	0	1	2	2	3	4	5	5

Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.

- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

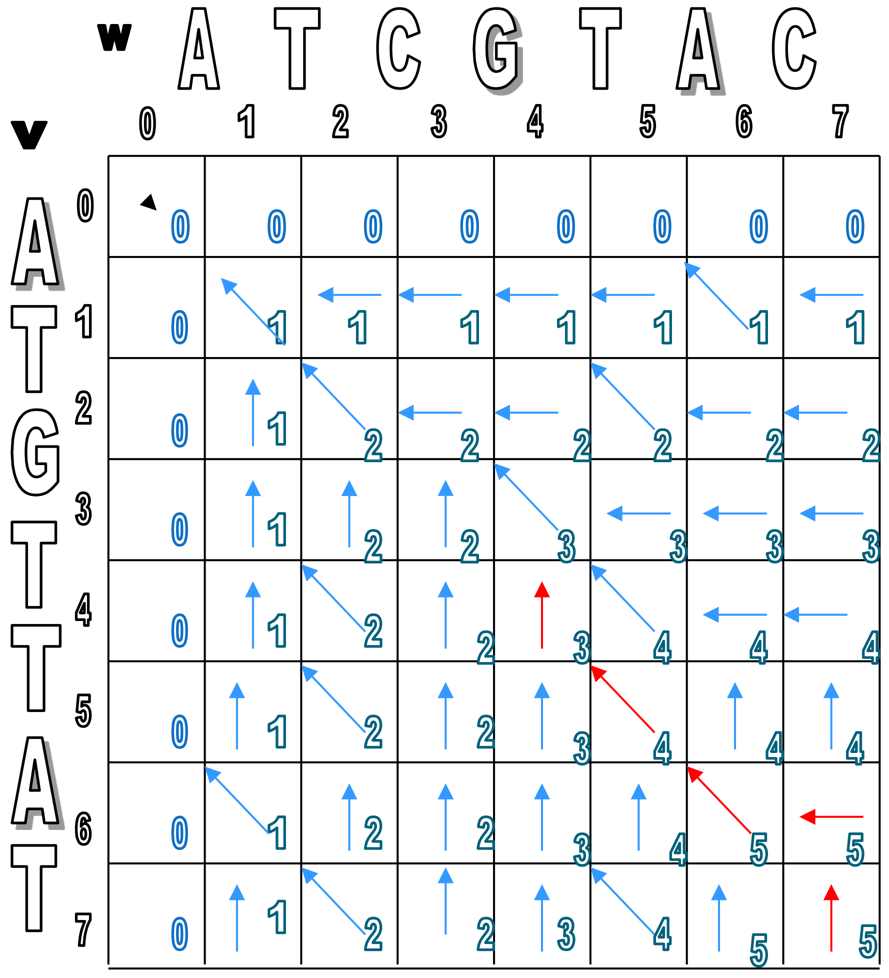
```
5 6 6 7
T A _ T
T A C _
5 6 7 7
```



Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.
- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

4	5	6	6	7
T	T	A	_	T
_	T	A	C	_
4	5	6	7	7

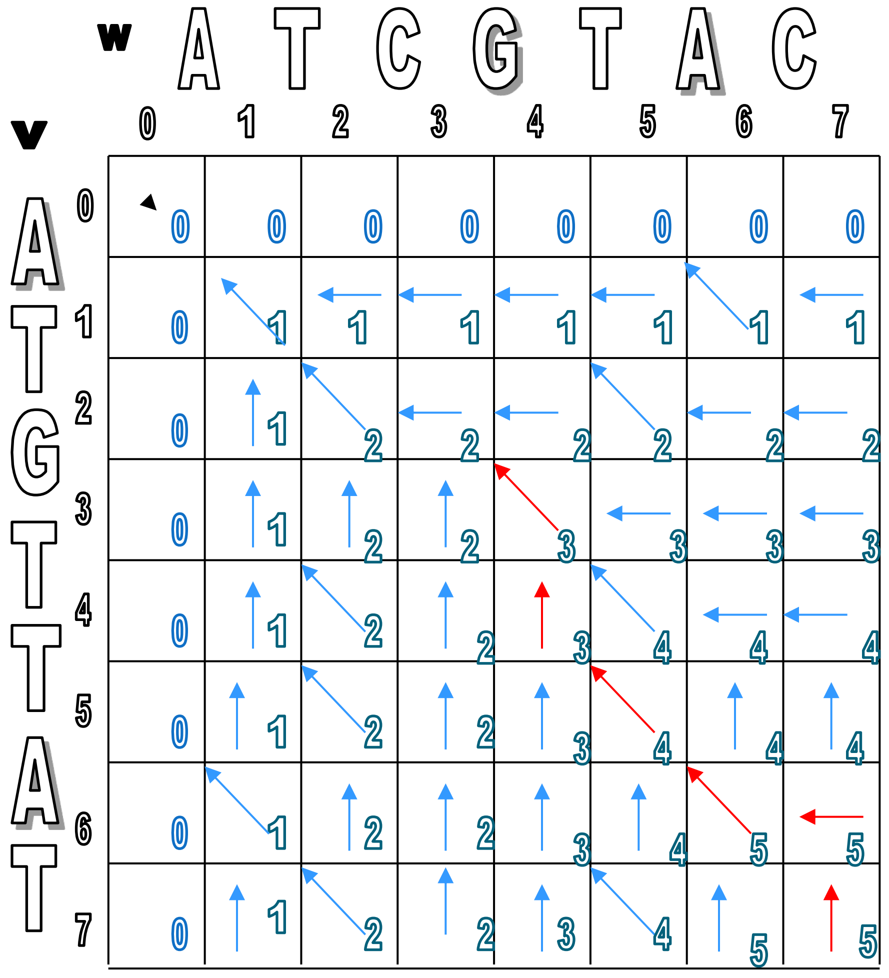


Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.

- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

3	4	5	6	6	7
G	T	T	A	_	T
G	_	T	A	C	_
4	4	5	6	7	7

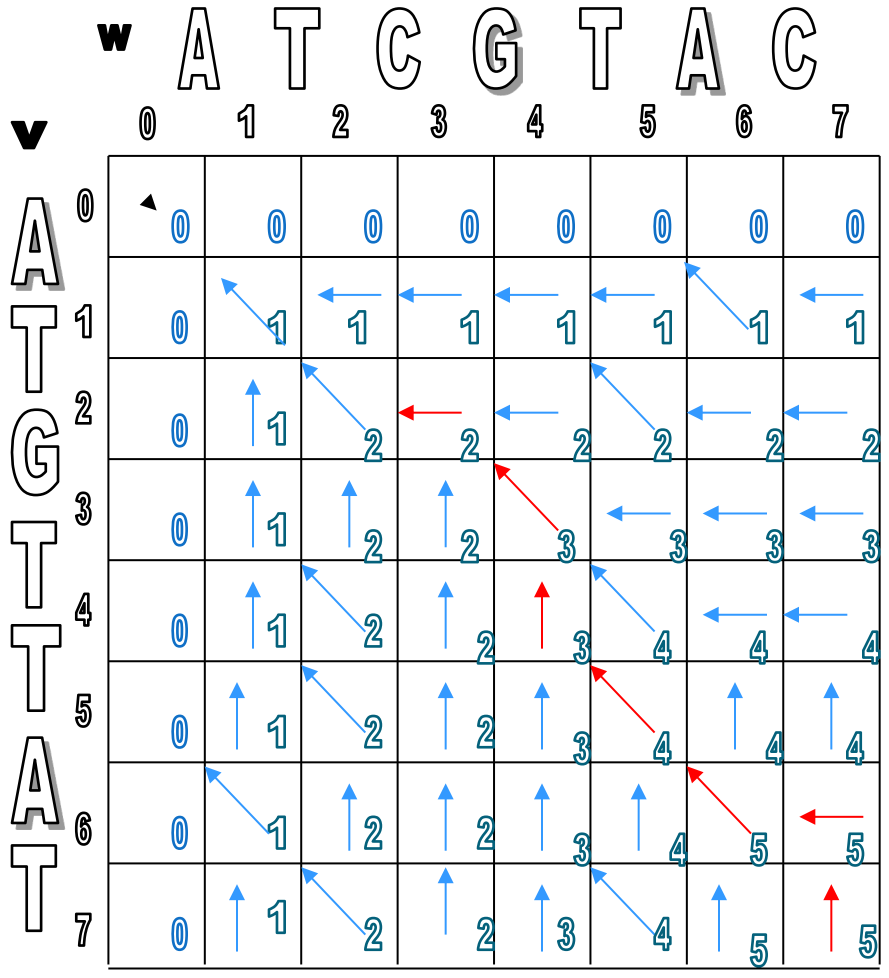


Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.

- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

2	3	4	5	6	6	7
_	G	T	T	A	_	T
C	G	_	T	A	C	_
3	4	4	5	6	7	7

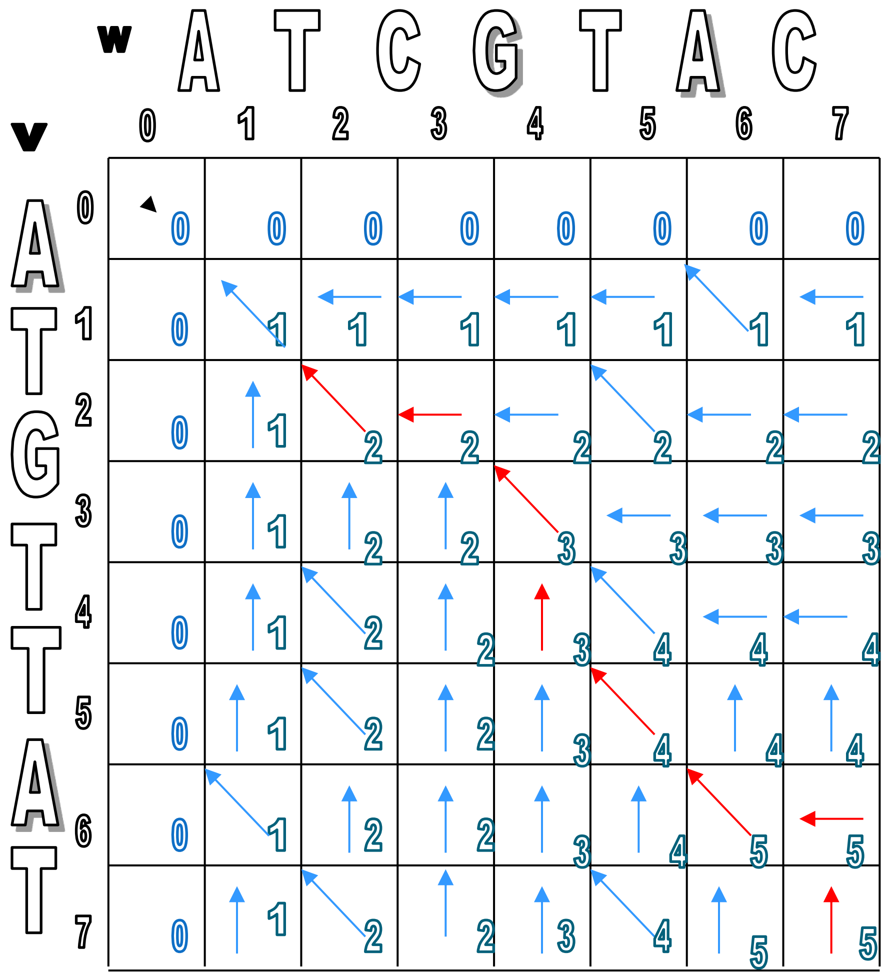


Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.

- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

2	2	3	4	5	6	6	7
T	_	G	T	T	A	_	T
T	C	G	_	T	A	C	_
2	3	4	4	5	6	7	7



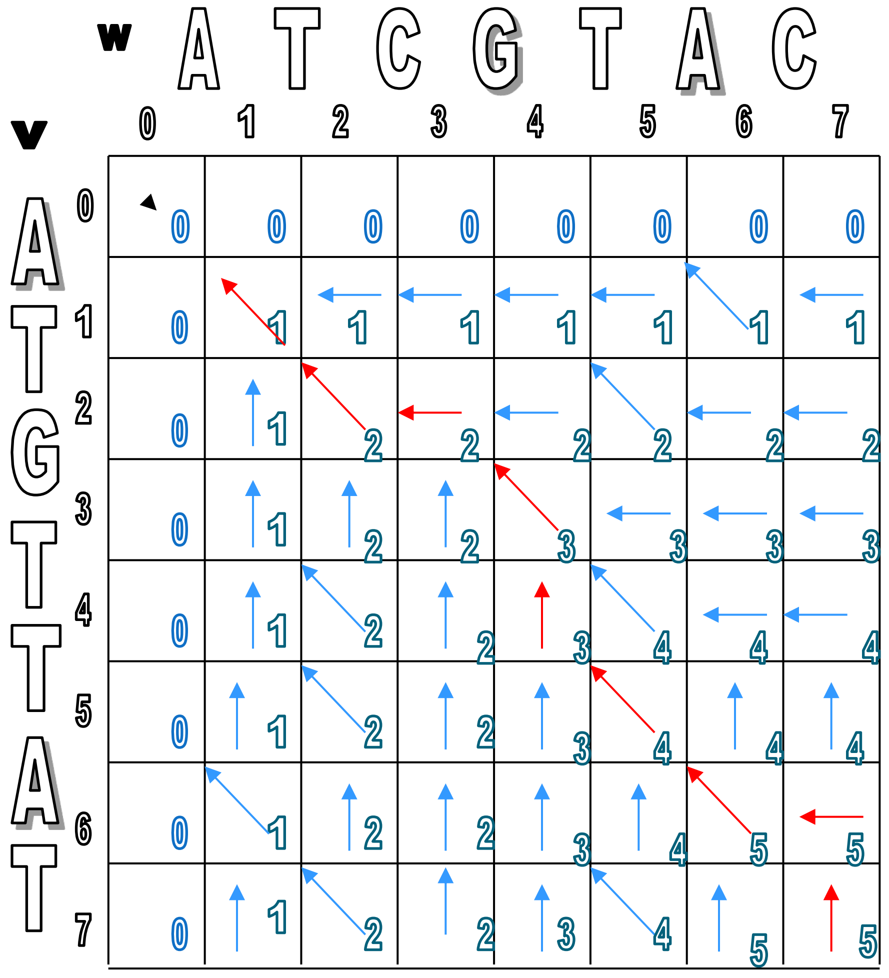
Departamento de Informática

Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.

- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

0	1	2	2	3	4	5	6	6	7
	A	T	_	G	T	T	A	_	T
	A	T	C	G	_	T	A	C	_
0	1	2	3	4	4	5	6	7	7

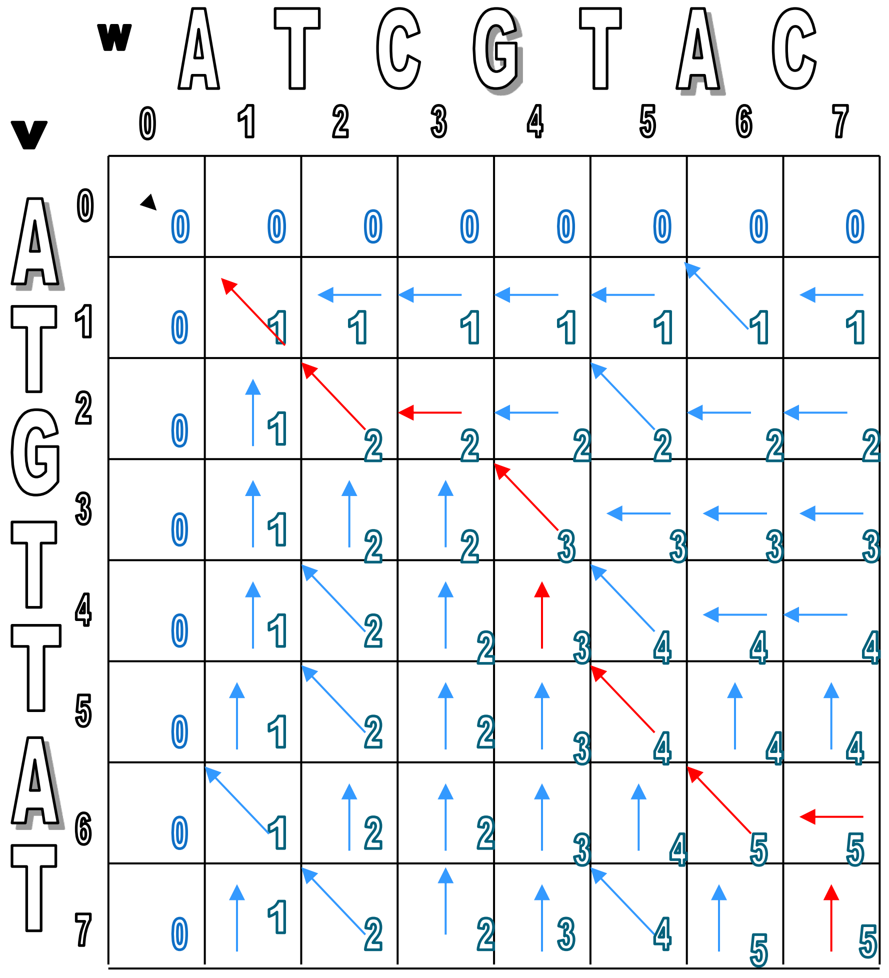


Departamento de Informática

Finding an Optimal Alignment

- LCS(v,w) created the alignment grid.
- Follow the arrows backwards from the sink to the source to obtain the path corresponding to an optimal alignment:

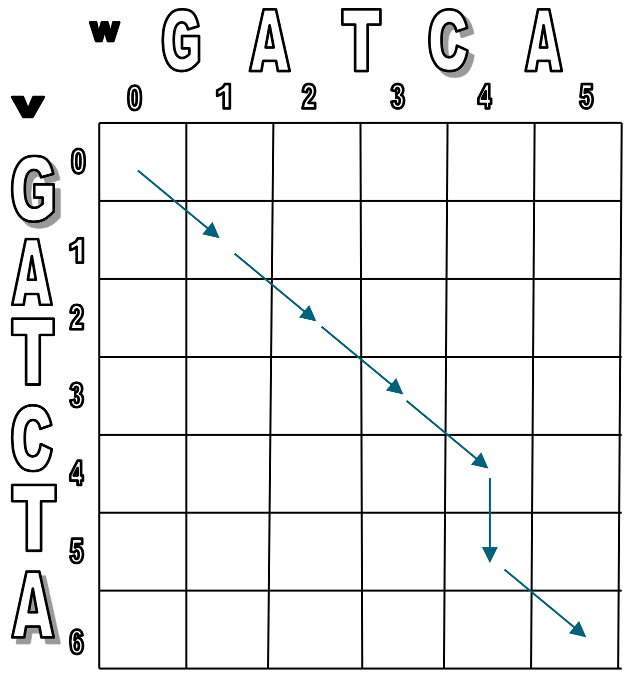
0	1	2	2	3	4	5	6	6	7
	A	T	_	G	T	T	A	_	T
	A	T	C	G	_	T	A	C	_
0	1	2	3	4	4	5	6	7	7



Exercise#3

Find the alignment as a path for the following sequences:

V=GATCTA, and W=GATCA.



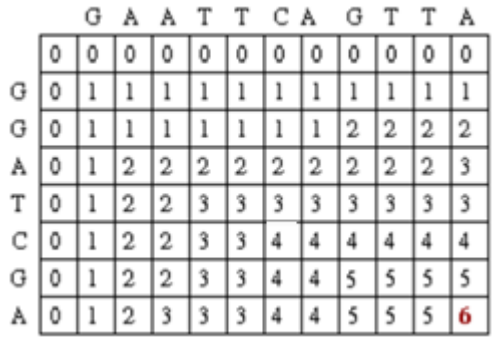
G A T C T A
G A T C _ A

Departamento de
Informática

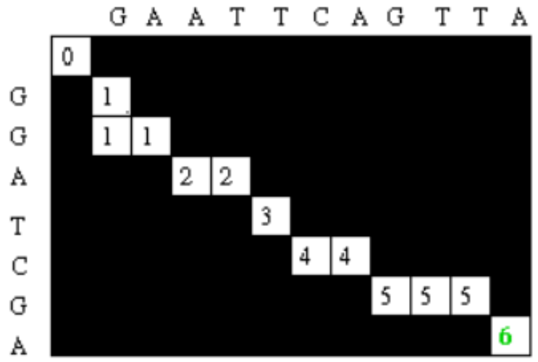
Exercise#4

Find an optimal (global) alignment for the following sequences: GAATTCAGTTA, and GGATCGA.

G _ A A T T C A G T T A
G G _ A _ T C _ G_ _ A



		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3	3
C	0	1	2	2	3	3	4	4	4	4	4	4
G	0	1	2	2	3	3	4	4	5	5	5	5
A	0	1	2	3	3	3	4	4	5	5	5	6



		G	A	A	T	T	C	A	G	T	T	A
	0											
G		1										
G		1	1									
A				2	2							
T						3						
C							4	4				
G									5	5	5	
A												6

Departamento de Informática

Exercise#5

Explain the differences between local and global sequence alignment. In your answer, include:

a) The primary algorithmic difference between the Smith-Waterman and Dynamic Programming (Needleman-Wunsch algorithms).

b) A scenario where local alignment is preferred over global alignment and vice versa.

FACULDADE ENGENHARIA

Departamento de Informática

Exercise#5

Algorithmic Differences:

- **Local Alignment (Smith-Waterman)**: Identifies the optimal alignment for any substring of the given sequences. It initializes the first row and column with zeros (allowing alignments to start anywhere) and uses a traceback starting from the highest-scoring cell, stopping when a cell with a score of zero is reached.
- **Global Alignment (Needleman-Wunsch)**: Aligns two sequences from start to end, optimizing the alignment across their entire length. It initializes the first row and column with an increasing gap penalty and performs traceback from the bottom-right corner to the top-left, ensuring the entirety of both sequences is aligned.

FACULDADE ENGENHARIA

Departamento de Informática

Exercise#5

Preferred Scenarios

- **Local Alignment:** Preferred when the goal is to identify regions of high similarity within larger sequences that may be otherwise dissimilar. This is particularly useful in identifying functional domains or motifs within proteins or genes that have conserved functions across species.
- **Global Alignment:** Used when comparing sequences of roughly the same length and where the interest lies in aligning the sequences in their entirety. This is suitable for closely related sequences or when evolutionary changes between the sequences are to be studied along their full length.

Departamento de Informática