Introduction to sequence alignment

Andrey Prjibelski
Center for Algorithmic Biotechnology
SPbU

Alignment

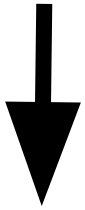
AACGCTAACGGTAA

AACCGCGAACTAA

Alignment

AACGCTAACGGTAA

AACCGCGAACTAA



AAC - GCTAACGGTAA

AACCGCGAAC - - TAA

Needleman–Wunsch algorithm

ACGTTAG

ACCTAG

Needleman–Wunsch algorithm

A C G T T A G

A C C T - A G

S(x, x) = 1 (match)

S(x, y) = -1 (mismatch)

S(x, -) = S(-, x) = -1 (Indel)

Score(ACGTTAG, ACCTAG) = 3

Dynamic programming

Solving a complex problem by breaking it into collection of simpler problems and storing intermediate results in specific data structures

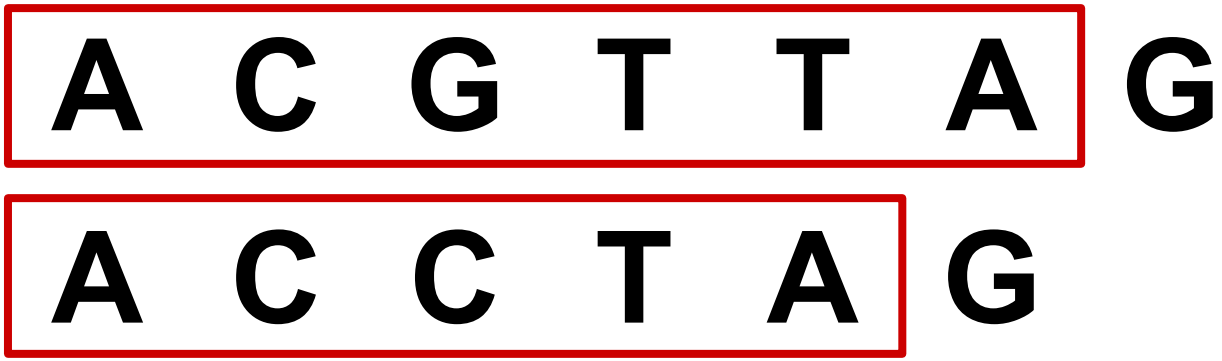
Dynamic programming

A C G T T A G

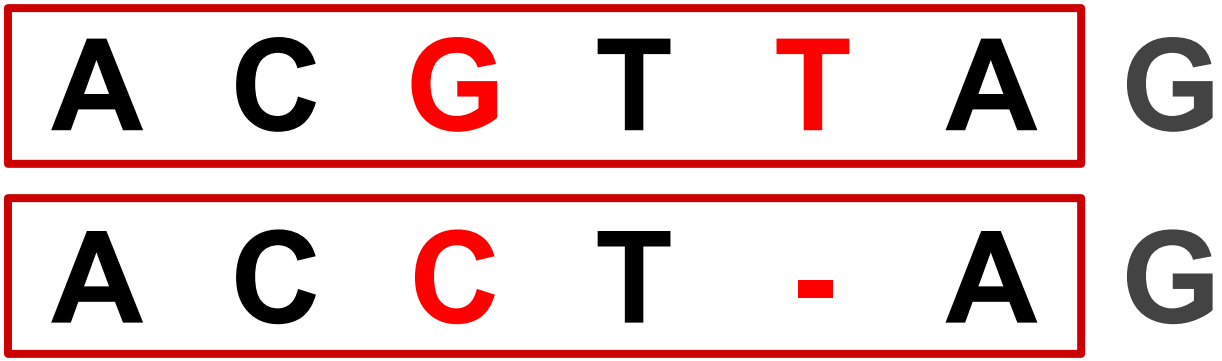
A C C T A G

Dynamic programming

Case 1

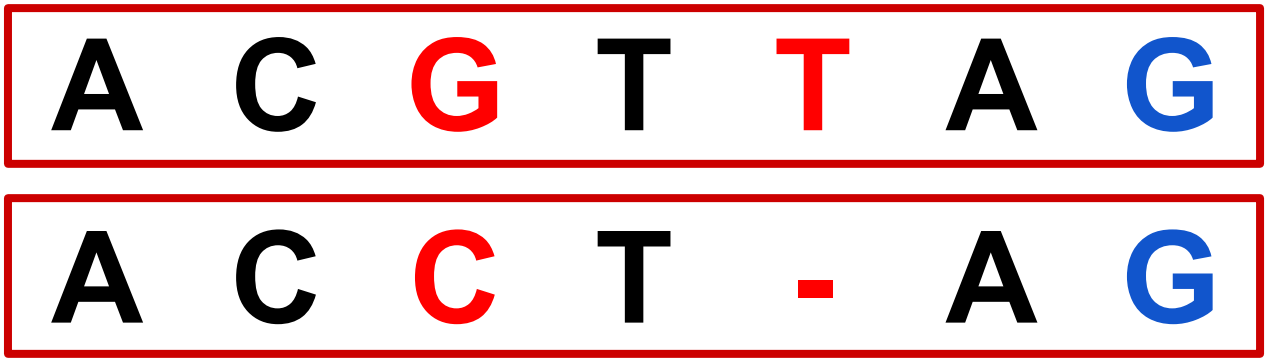


Dynamic programming



Score(ACGTTA, ACCTA) = 2

Dynamic programming

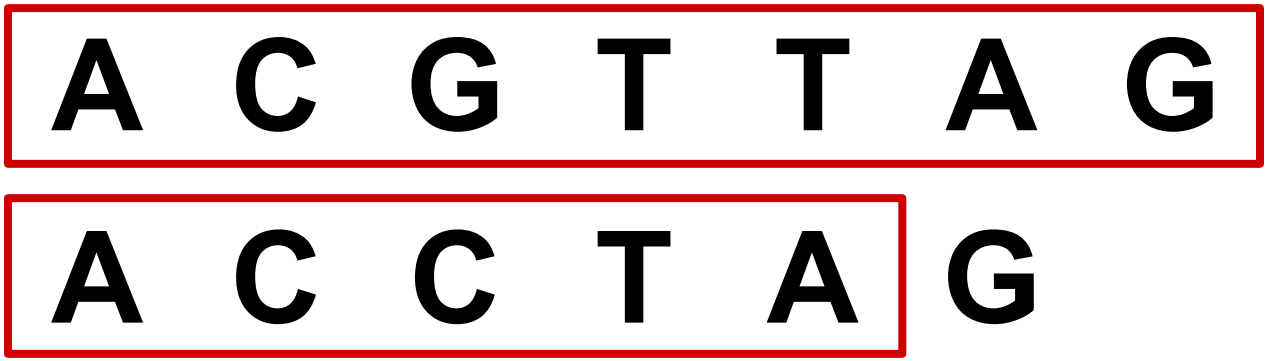


Score(ACGTTAG, ACCTAG) =

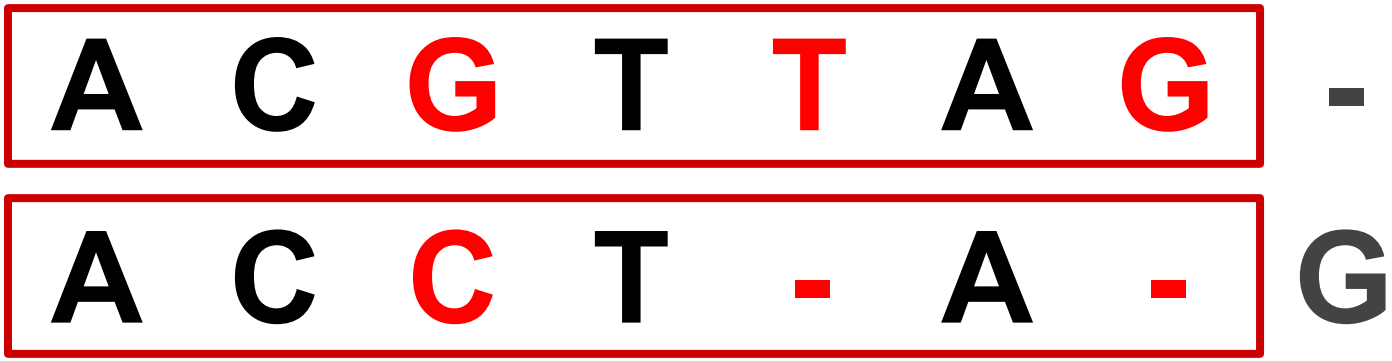
Score(ACGTTA, ACCTA) + S(G, G) = 2 + 1 = **3**

Dynamic programming

Case 2

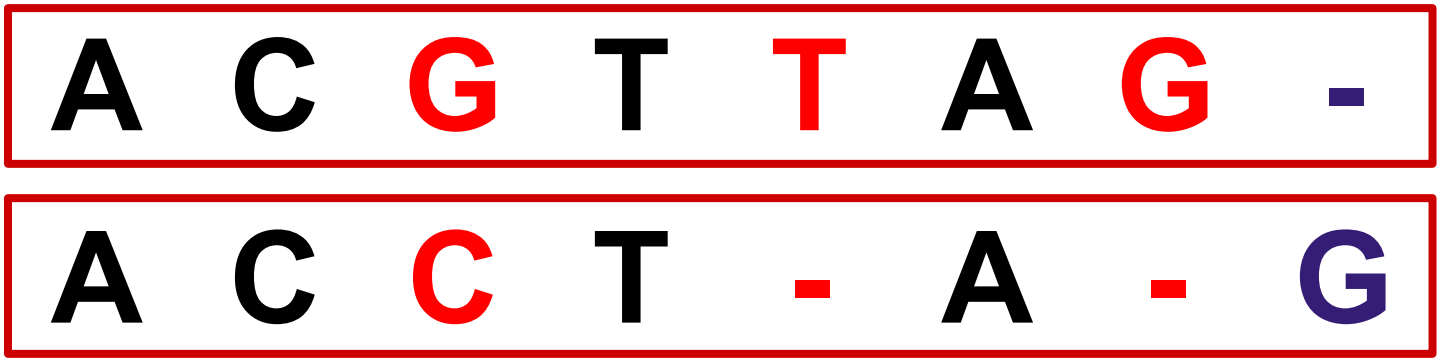


Dynamic programming



Score(ACGTTAG, ACCTA) = 1

Dynamic programming

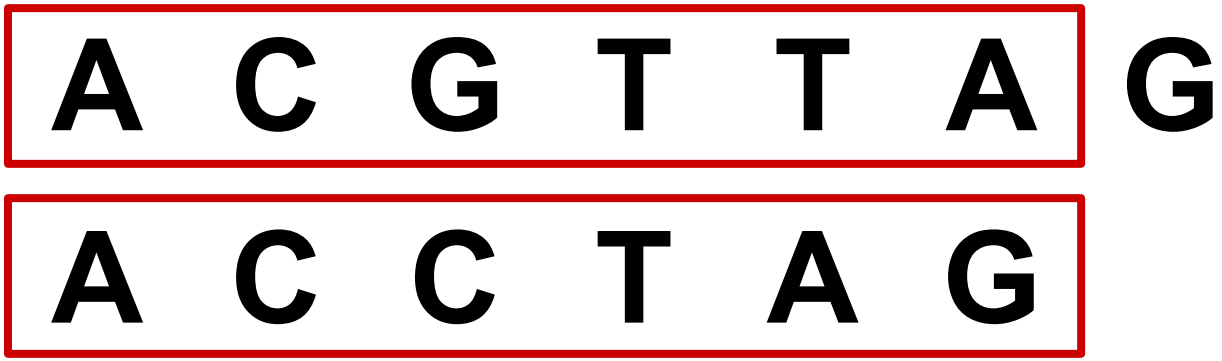


Score(ACGTTAG, ACCTAG) =

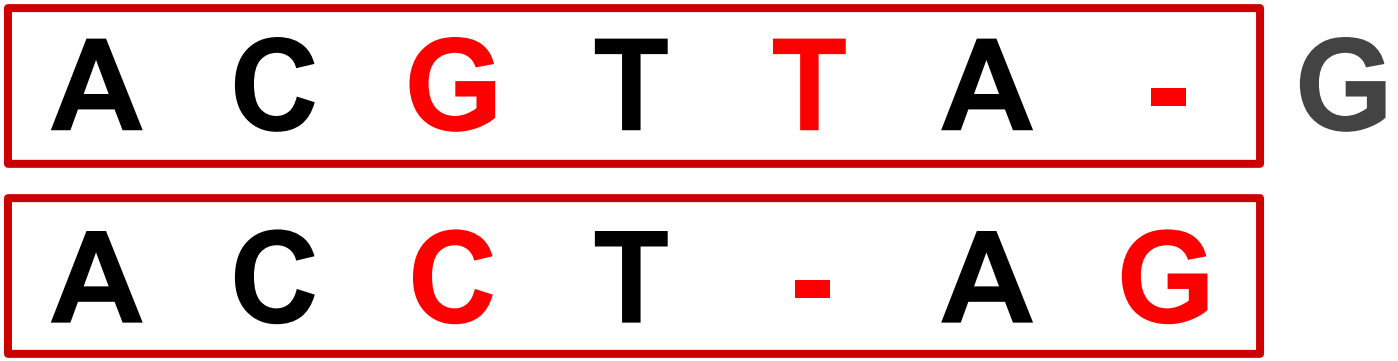
Score(ACGTTAG, ACCTA) + S(-, G) = 1 + (-1) = **0**

Dynamic programming

Case 3

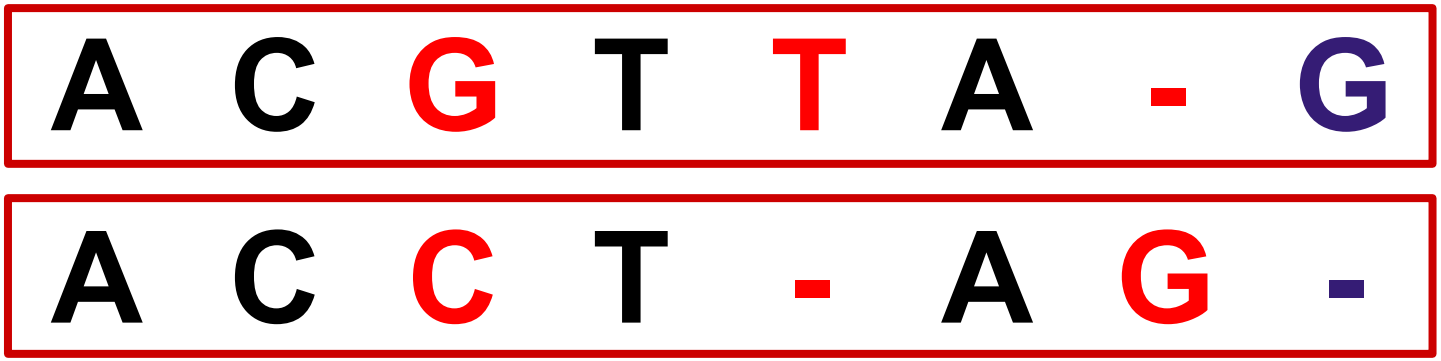


Dynamic programming



Score(ACGTTA, ACCTAG) = 1

Dynamic programming



Score(ACGTTAG, ACCTAG) =

Score(ACGTTA, ACCTAG) + S(G, -) = 1 + -1 = **0**

Dynamic programming

- Case 1

  Score(ACGTTAG, ACCTAG) = **3**

- Case 2

  Score(ACGTTAG, ACCTAG) = **0**

- Case 3

  Score(ACGTTAG, ACCTAG) = **0**

Dynamic programming

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

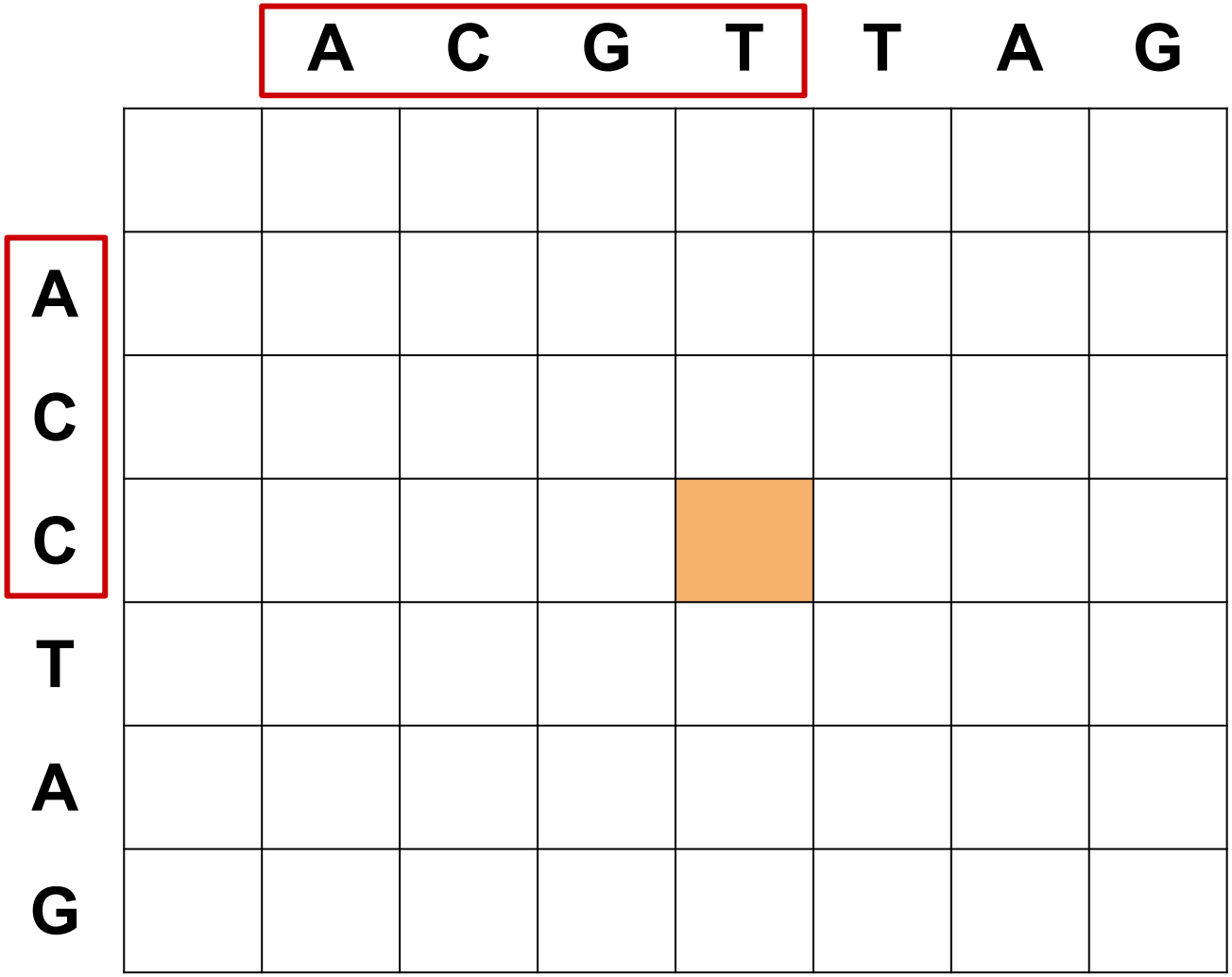
Score(ACGTTAG, ACCTAG) =

Score(ACGTTA, ACCTA) + S(G, G) = 2 + 1 = **3**

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm



Needleman–Wunsch algorithm

		A	C	G	T	T	A	G
	0							
A								
C								
C								
T								
A								
G								

Needleman–Wunsch algorithm

		A	C	G	T	T	A	G
	0	-1						
A								
C								
C								
T								
A								
G								

Needleman–Wunsch algorithm

		A	C	G	T	T	A	G
	0	-1	-2					
A								
C								
C								
T								
A								
G								

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

- - - -

A C C T

Score(ACGT, “”) = -4

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

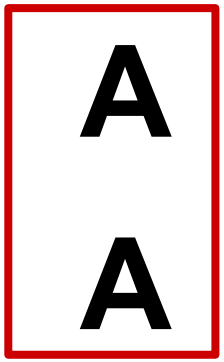
Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

Score(“ ”, “ ”) = 0

Needleman–Wunsch algorithm



Score(A, A) = Score(“ ”, “ ”) + S(A, A) = 0 + 1 = 1

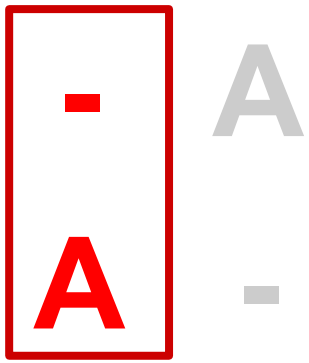
Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

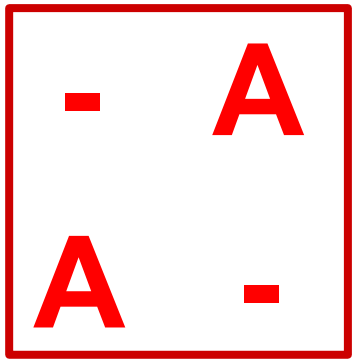
		A	C	G	T	T	A	G
	0	-1	-2	-3	-4	-5	-6	-7
A	-1	1						
C	-2							
C	-3							
T	-4							
A	-5							
G	-6							

Needleman–Wunsch algorithm



Score(“ ”, A) = -1

Needleman–Wunsch algorithm



Score(A, A) = Score(“ ”, A) + S(A, -) = -1 + -1 = -2

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

$$M(i, j) = \begin{cases} \text{MAX} \begin{cases} M(i-1, j-1) + S(X(i), Y(j)) \\ M(i-1, j) + S(X(i), \text{‘-’}) \\ M(i, j-1) + S(\text{‘-’}, Y(j)) \end{cases} \end{cases}$$

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

Needleman–Wunsch algorithm

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

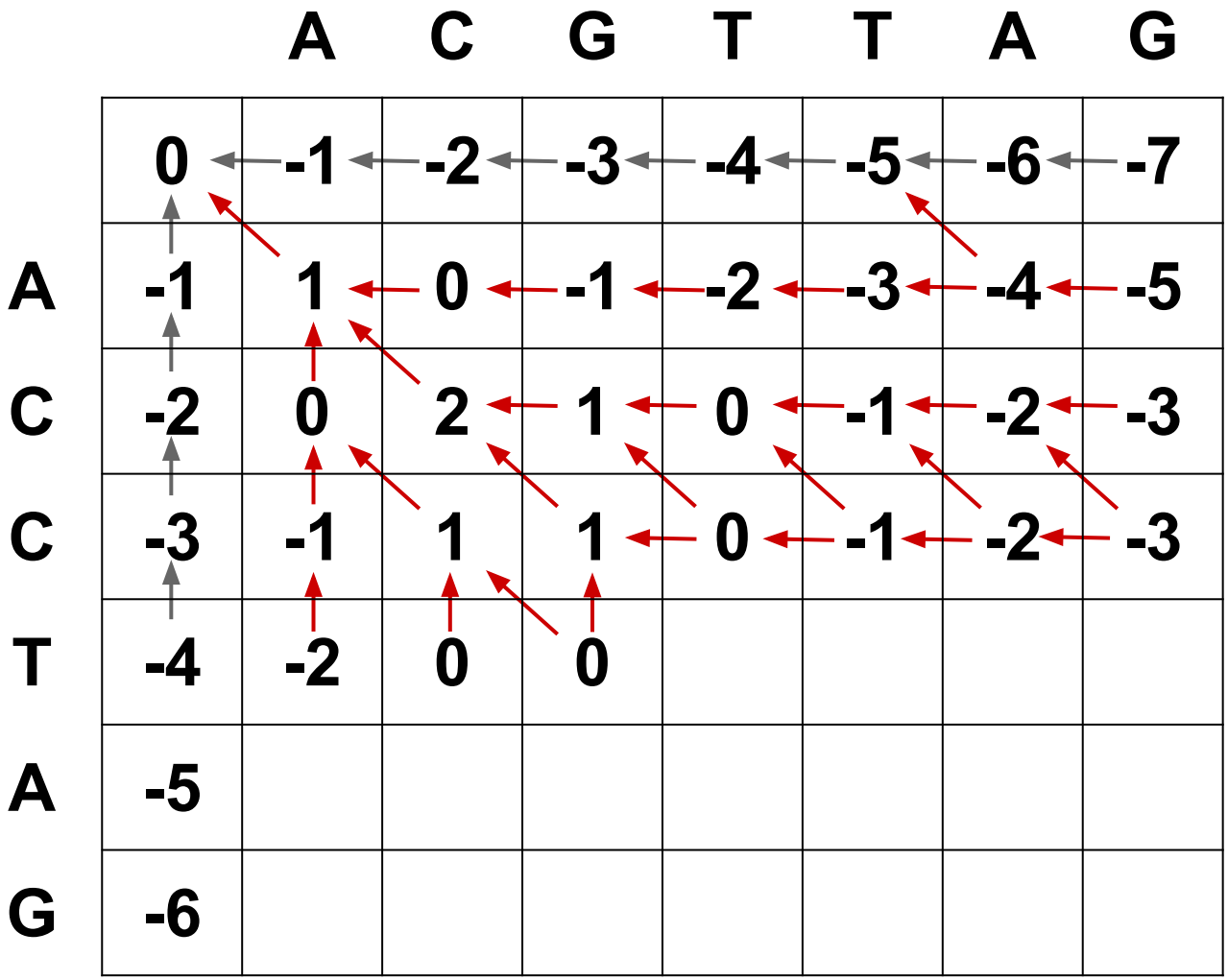
Needleman–Wunsch algorithm

$$M(i, j) = \begin{cases} \text{MAX} \begin{cases} M(i-1, j-1) + S(X(i), Y(j)) \\ M(i-1, j) + S(X(i), \text{‘-’}) \\ M(i, j-1) + S(\text{‘-’}, Y(j)) \end{cases} \end{cases}$$

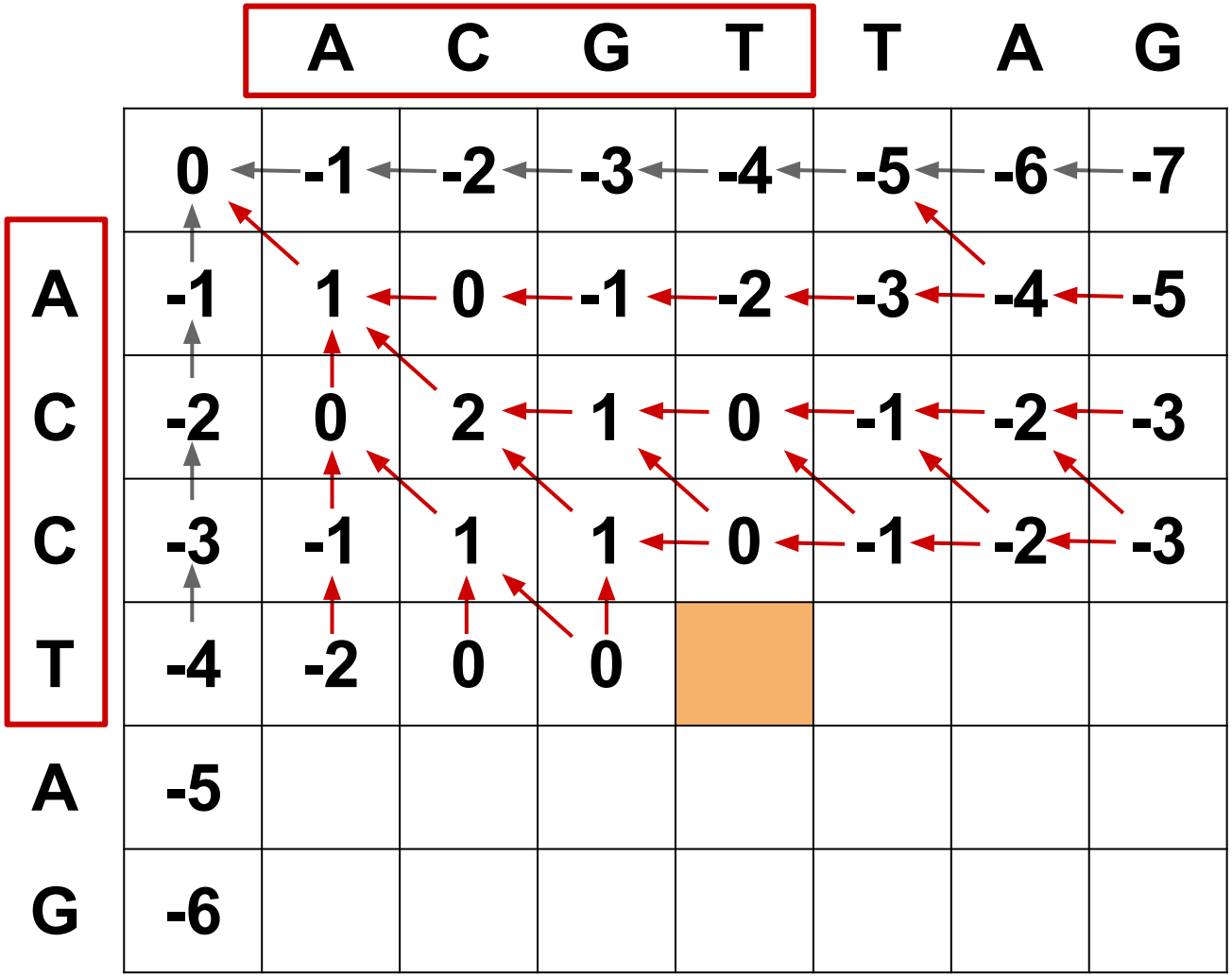
Needleman–Wunsch algorithm

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

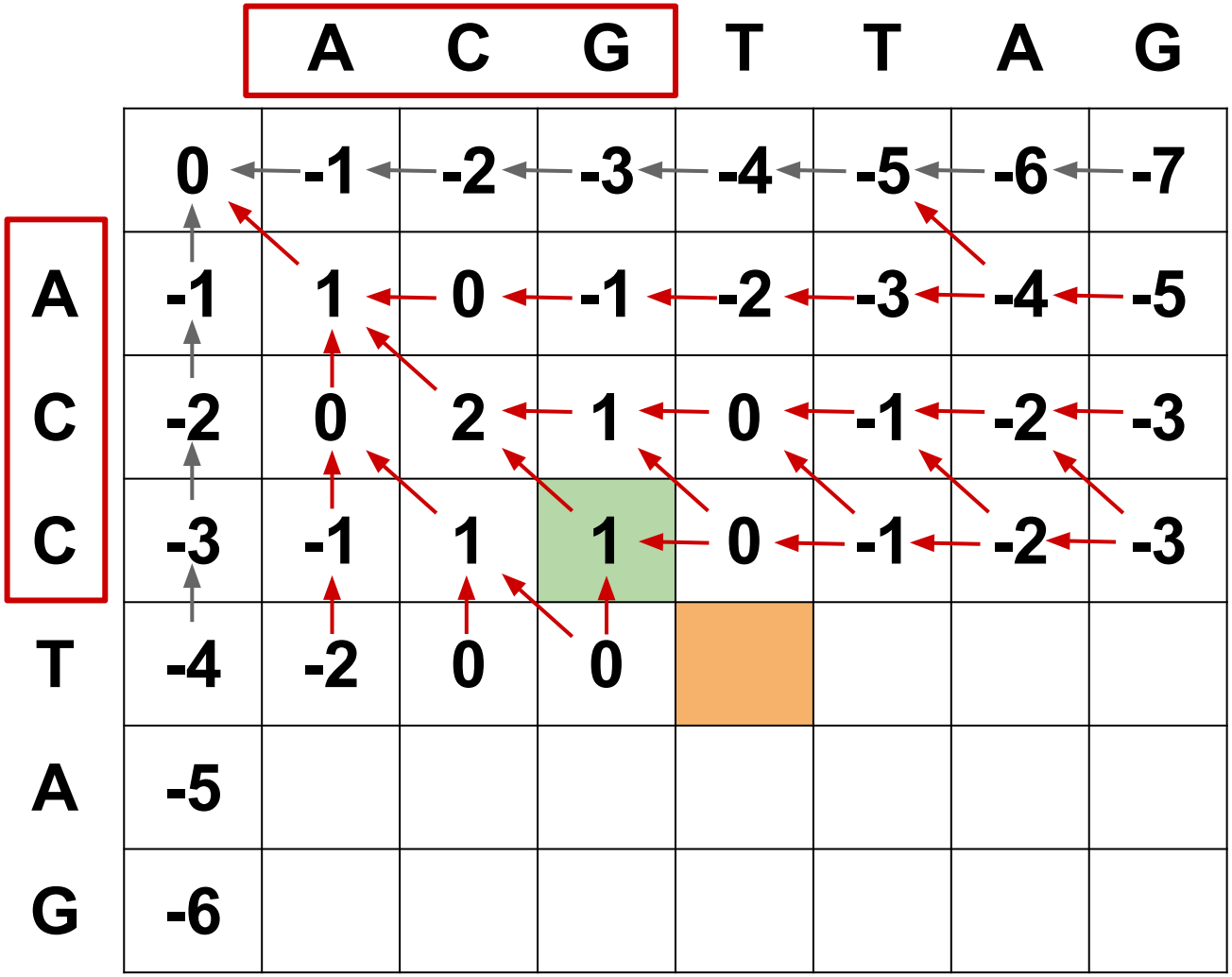
Needleman–Wunsch algorithm



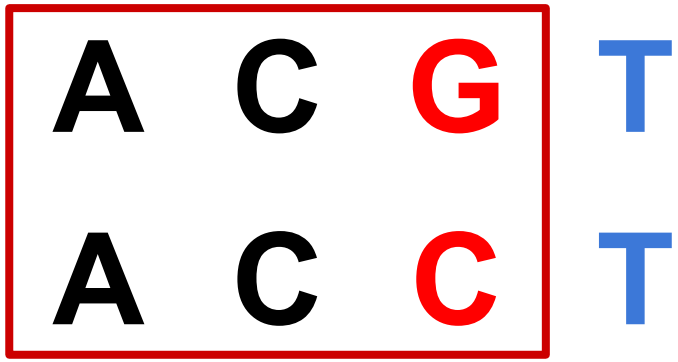
Needleman–Wunsch algorithm



Needleman–Wunsch algorithm



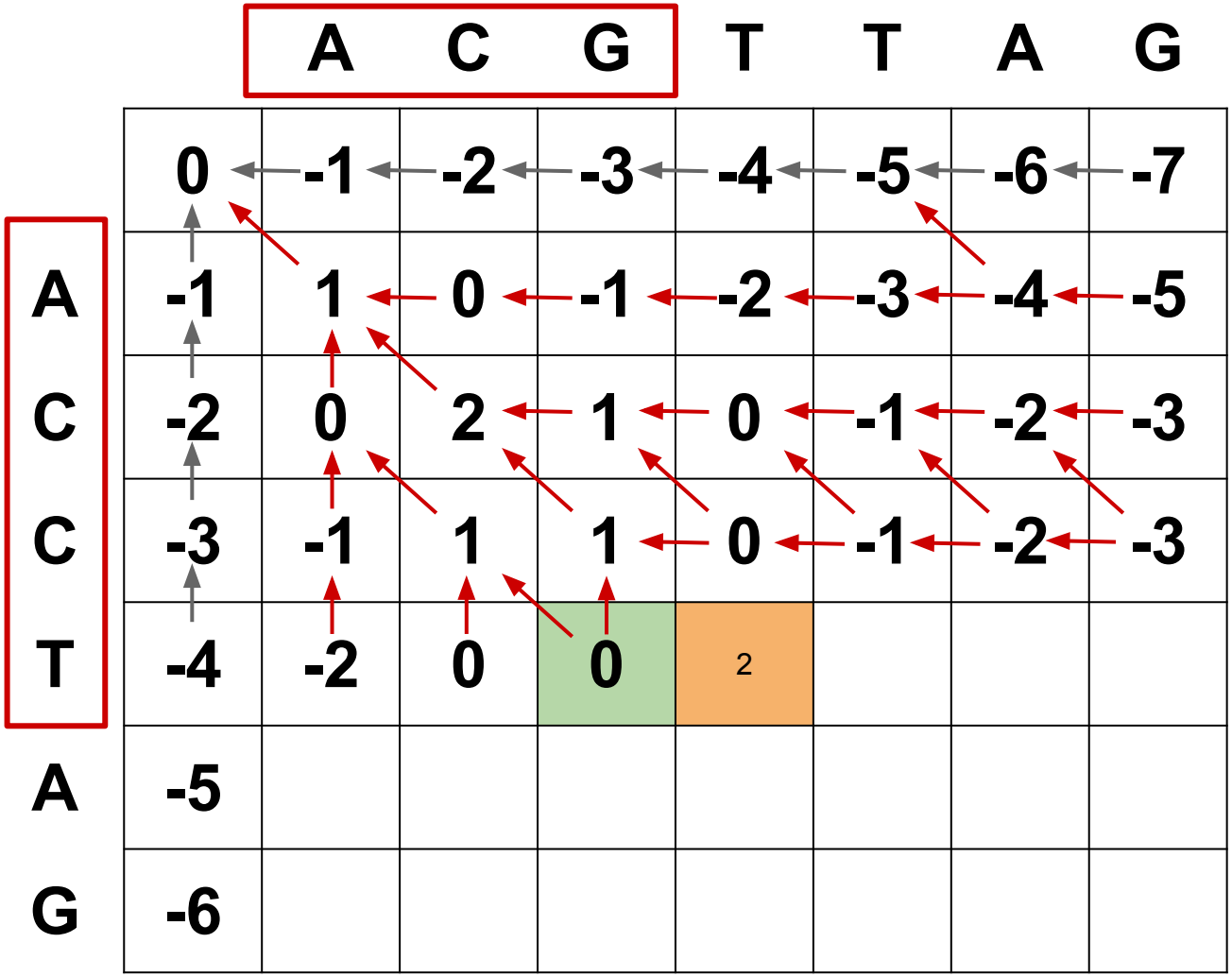
Dynamic programming



Score(ACGT, ACCT) =

Score(ACG, ACC) + S(T, T) = 1 + 1 = 2

Needleman–Wunsch algorithm



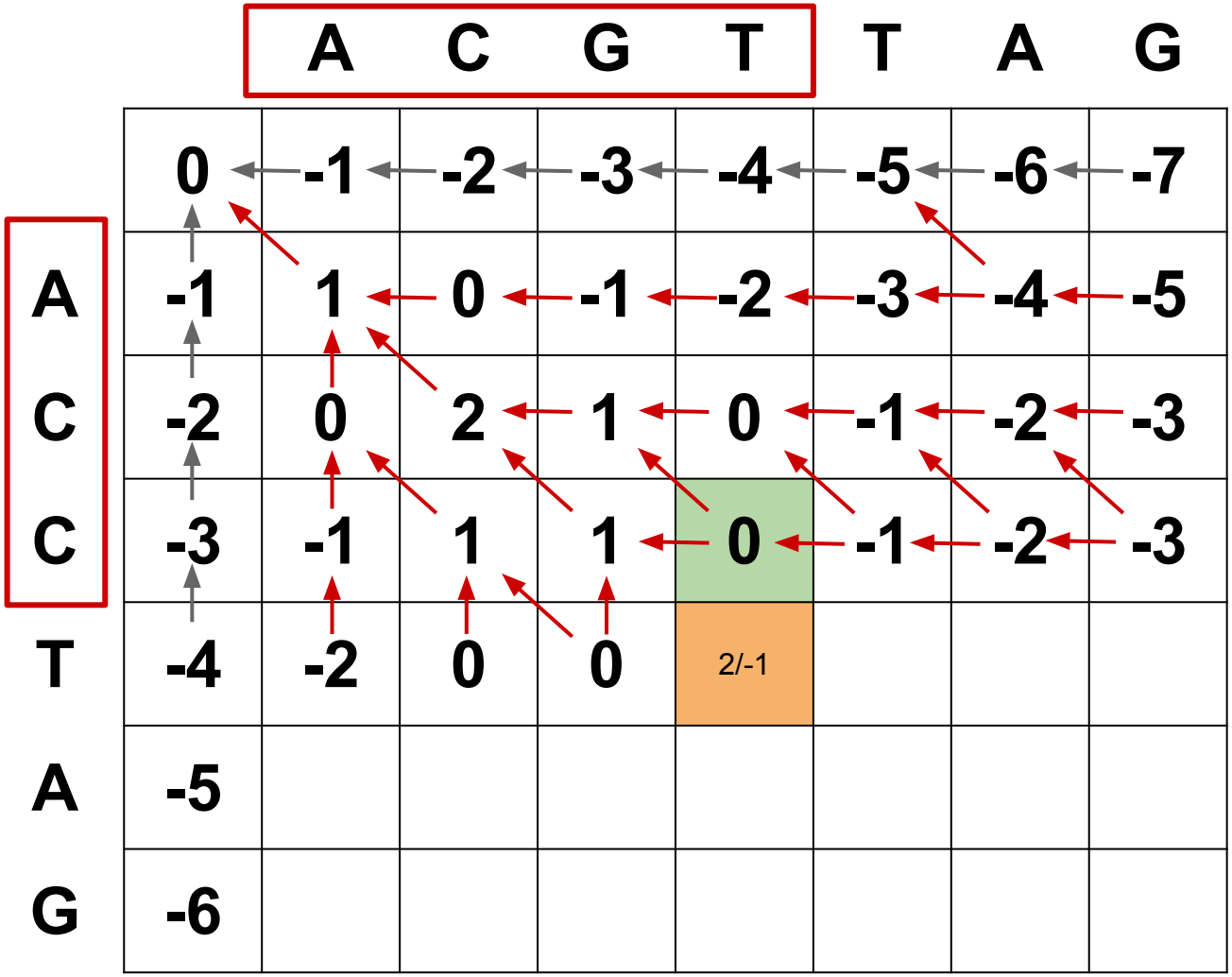
Dynamic programming

A	C	G	-	T
A	C	C	T	-

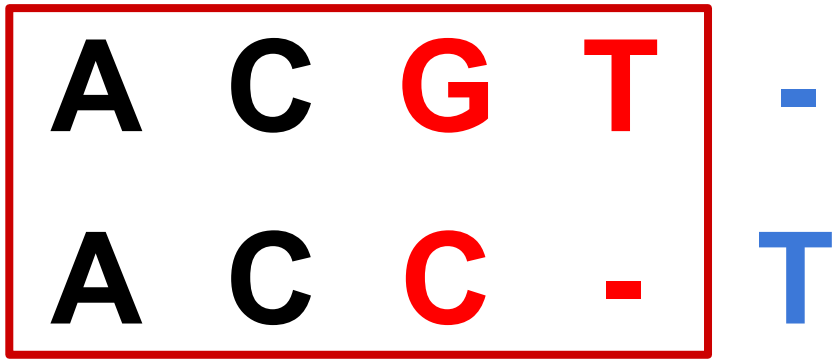
Score(ACGT, ACCT) =

Score(ACG, ACCT) + S(T, -) = 0 + -1 = -1

Needleman–Wunsch algorithm



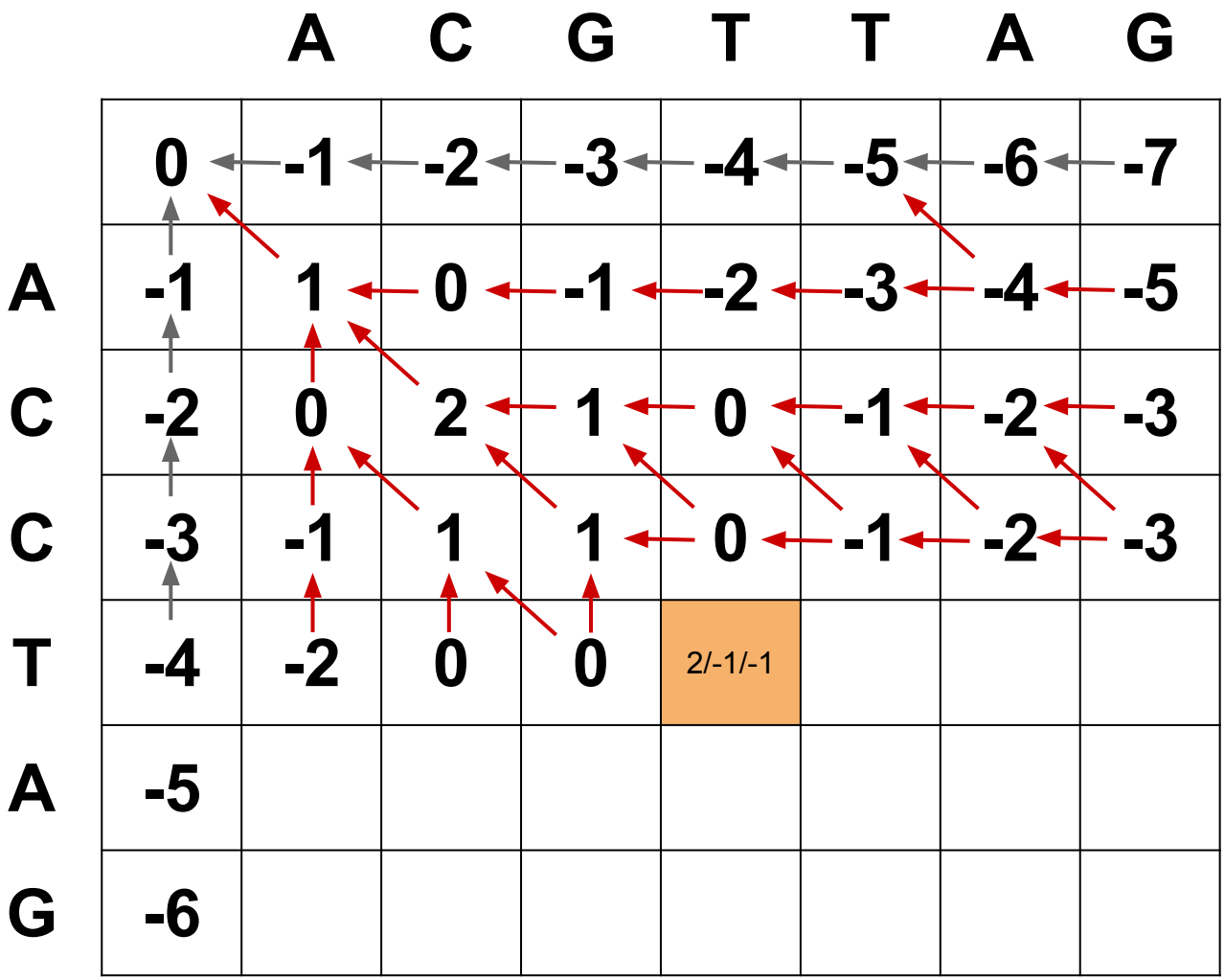
Dynamic programming



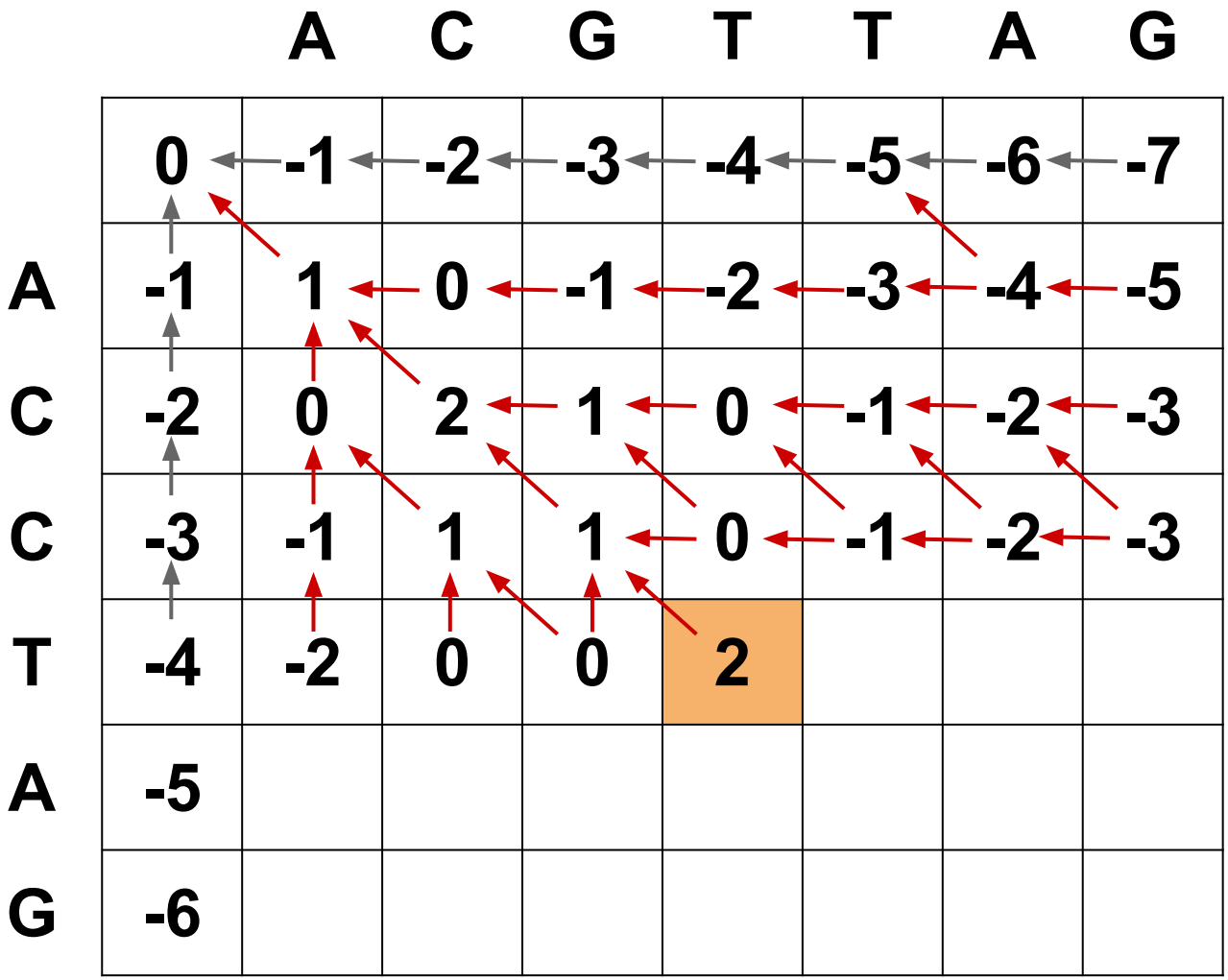
Score(ACGT, ACCT) =

Score(ACGT, ACC) + S(-, T) = 0 + -1 = -1

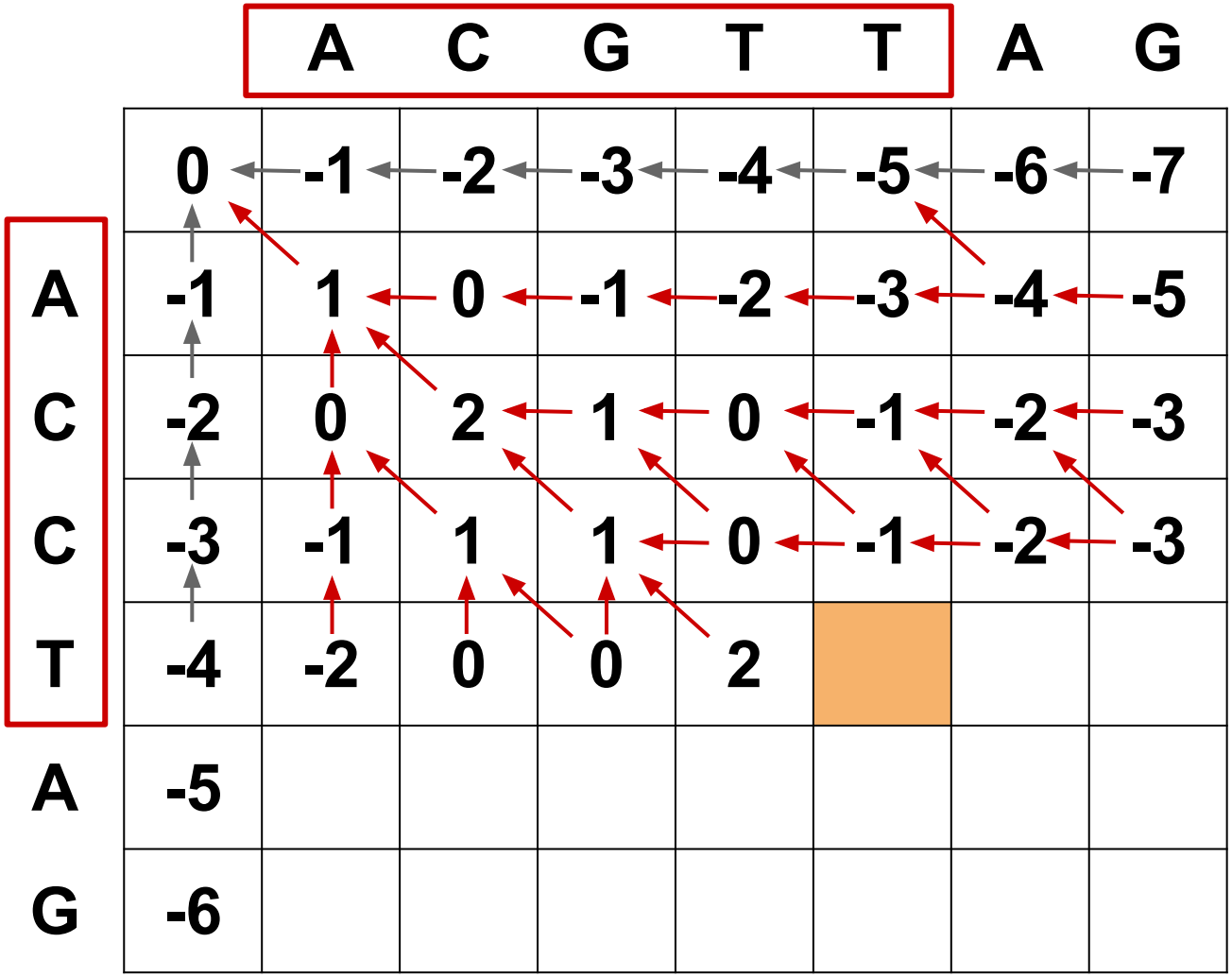
Needleman–Wunsch algorithm



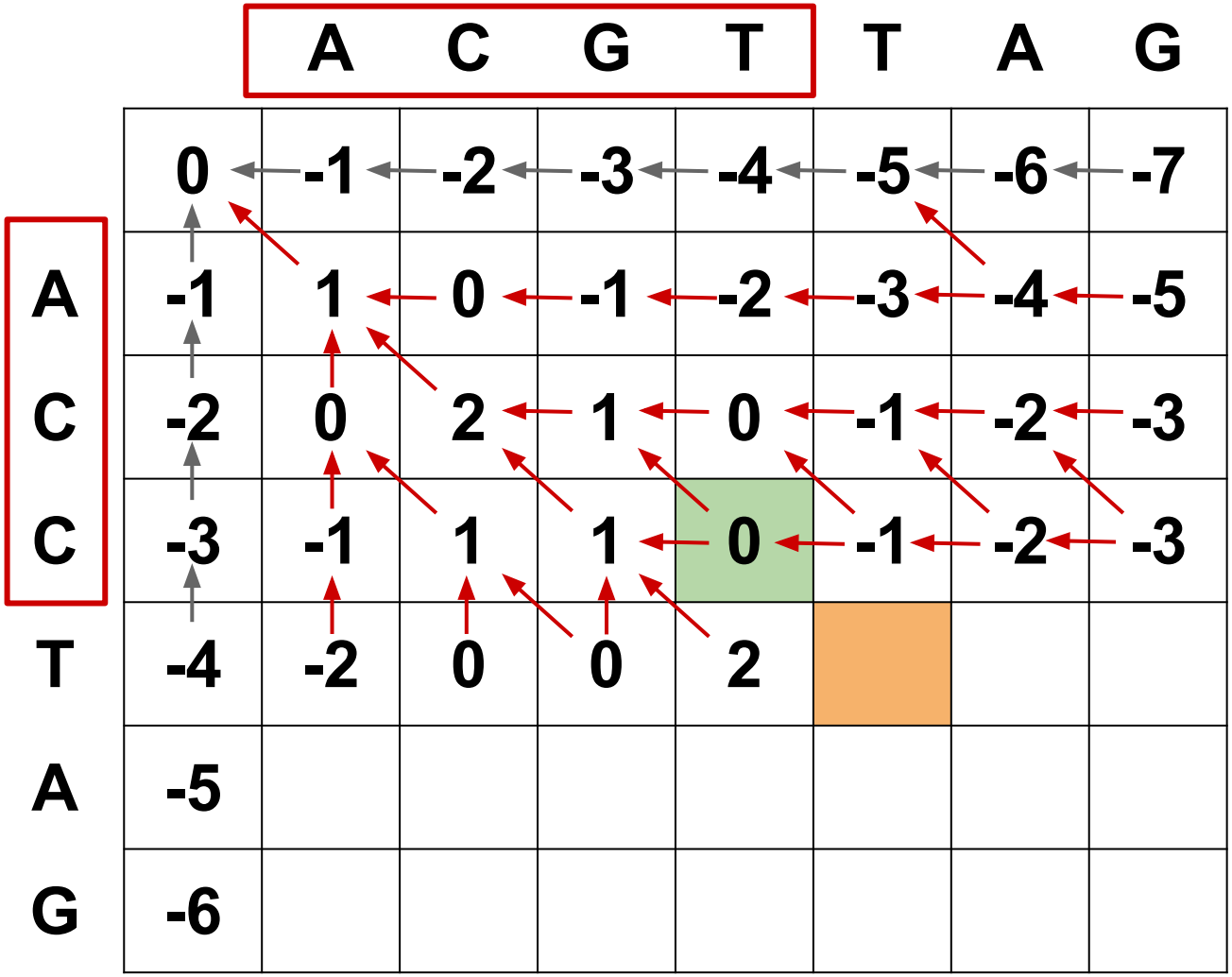
Needleman–Wunsch algorithm



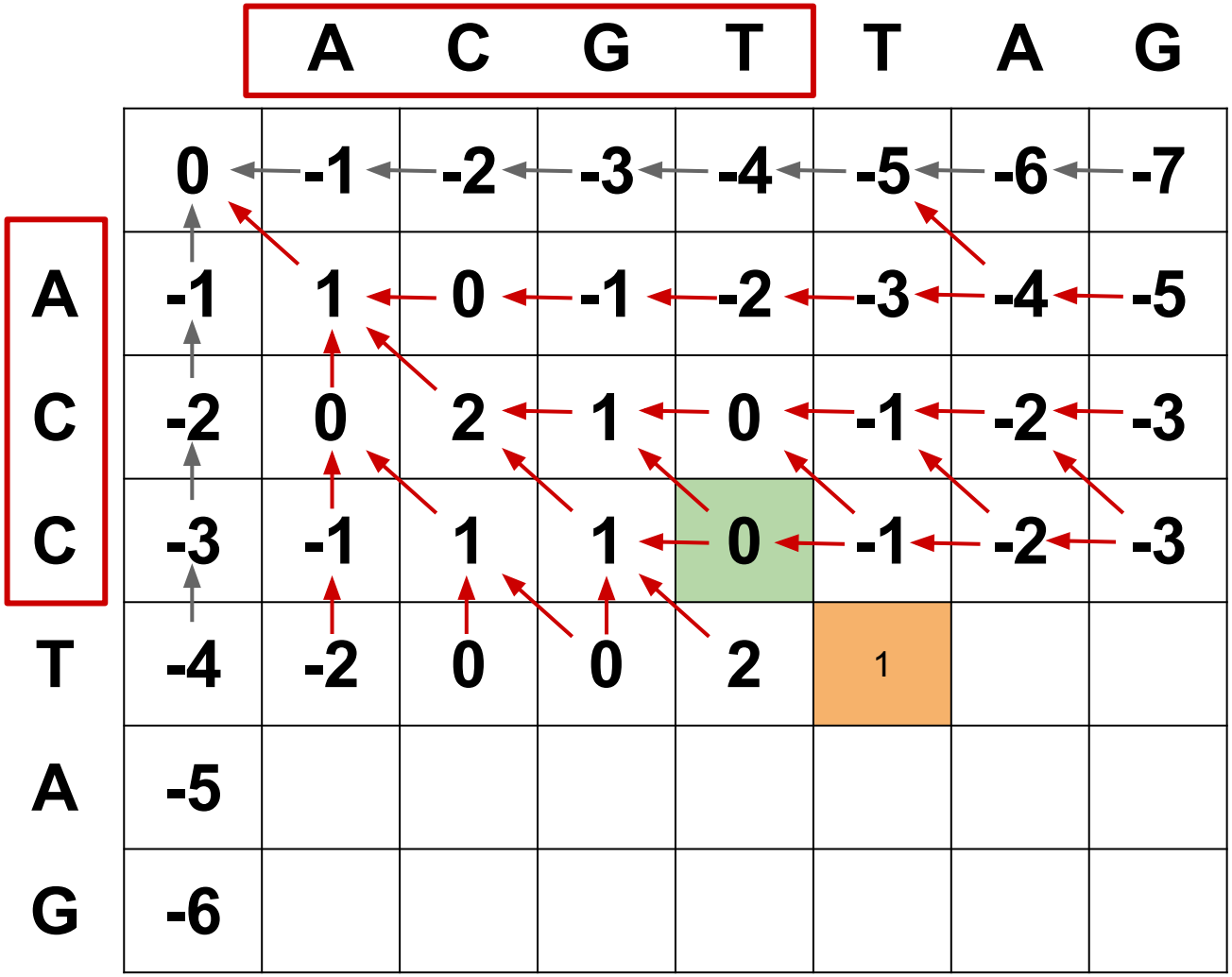
Needleman–Wunsch algorithm



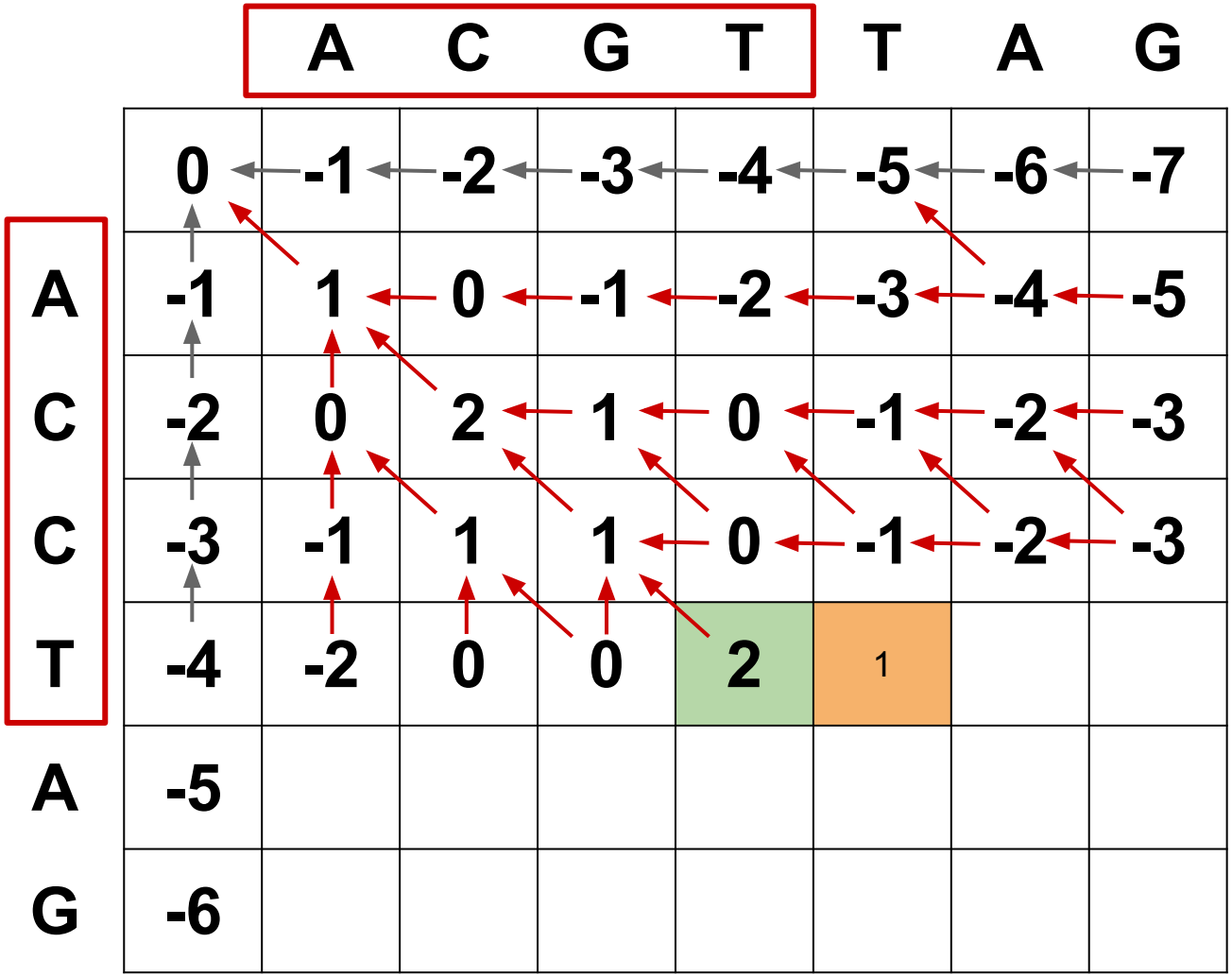
Needleman–Wunsch algorithm



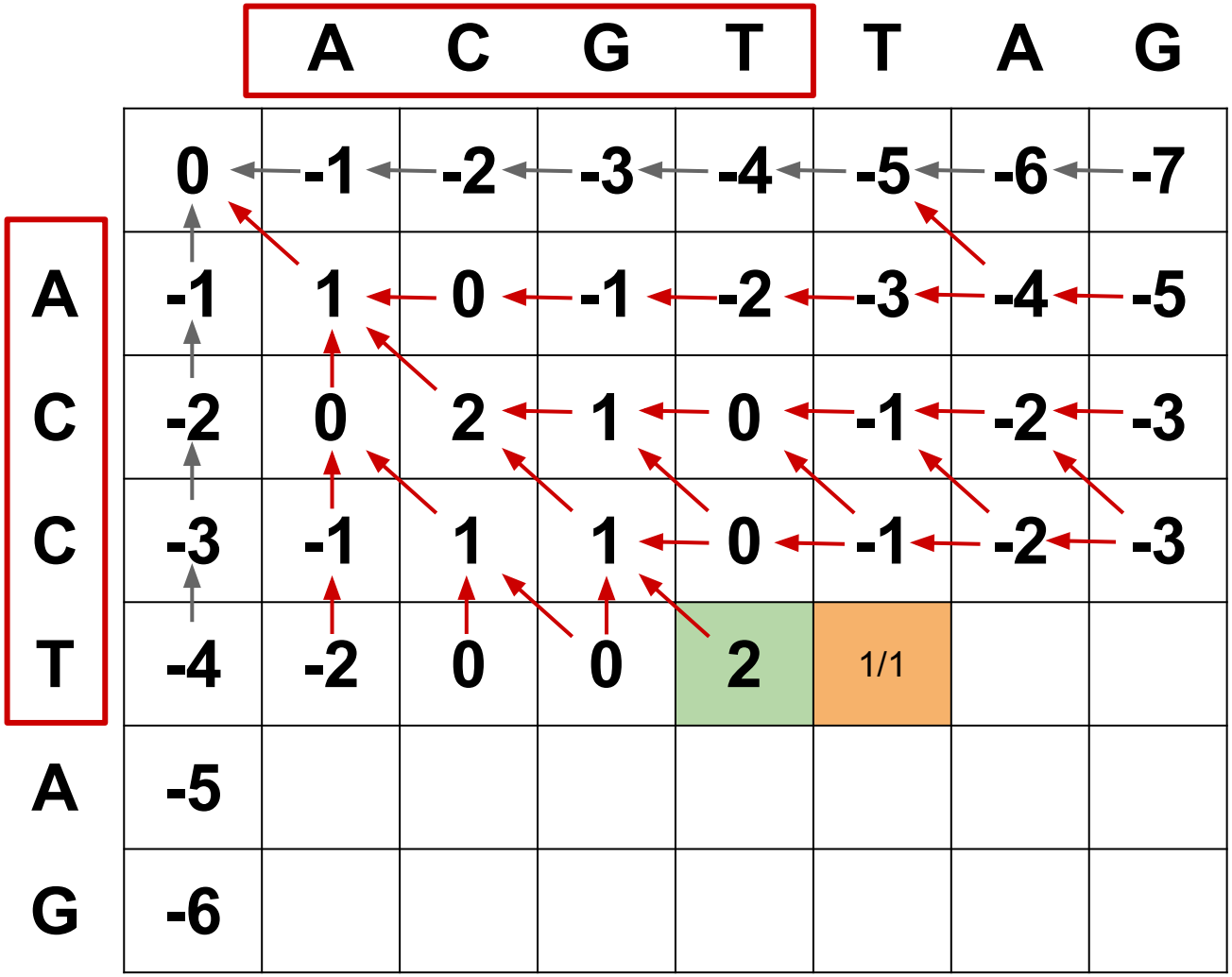
Needleman–Wunsch algorithm



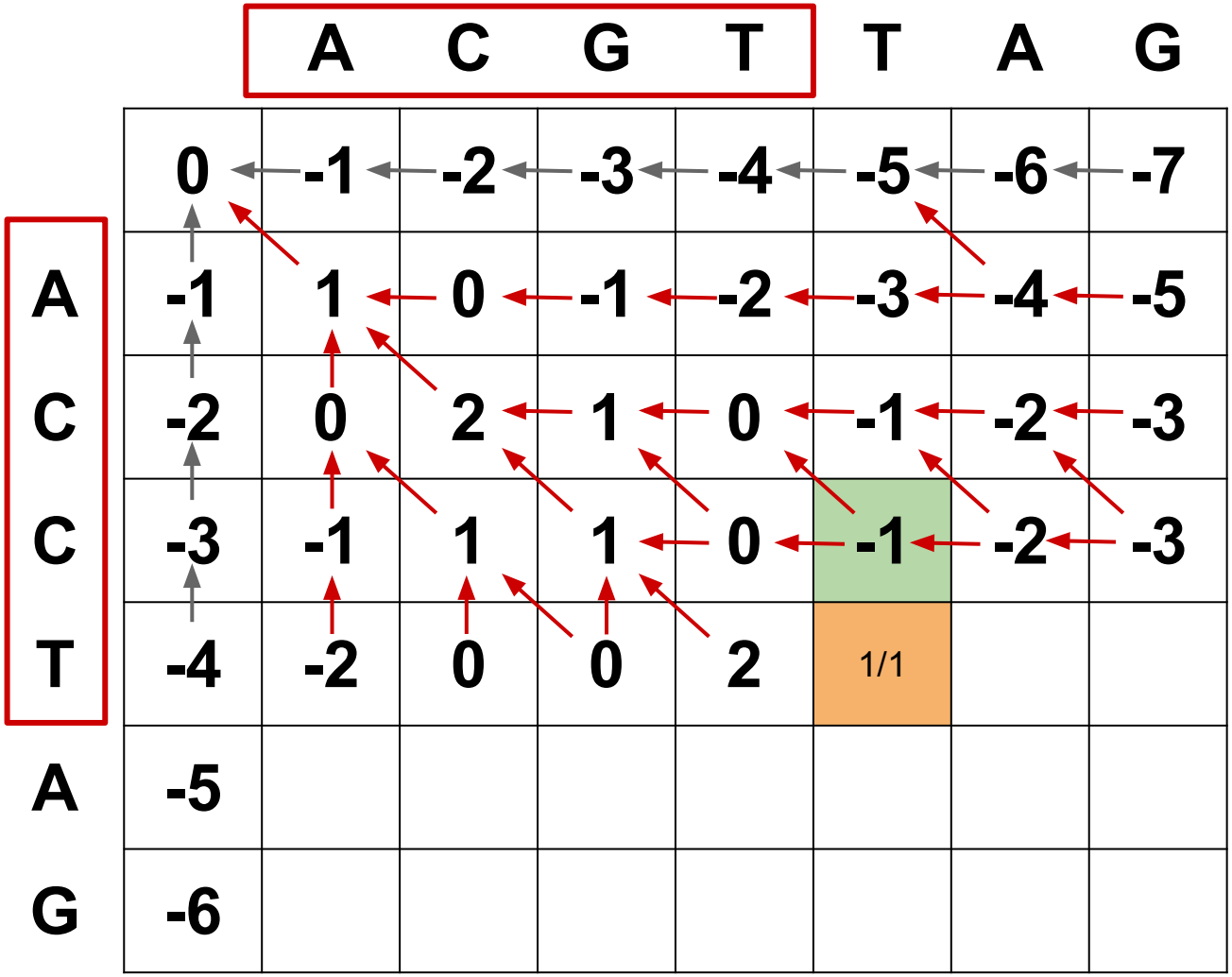
Needleman–Wunsch algorithm



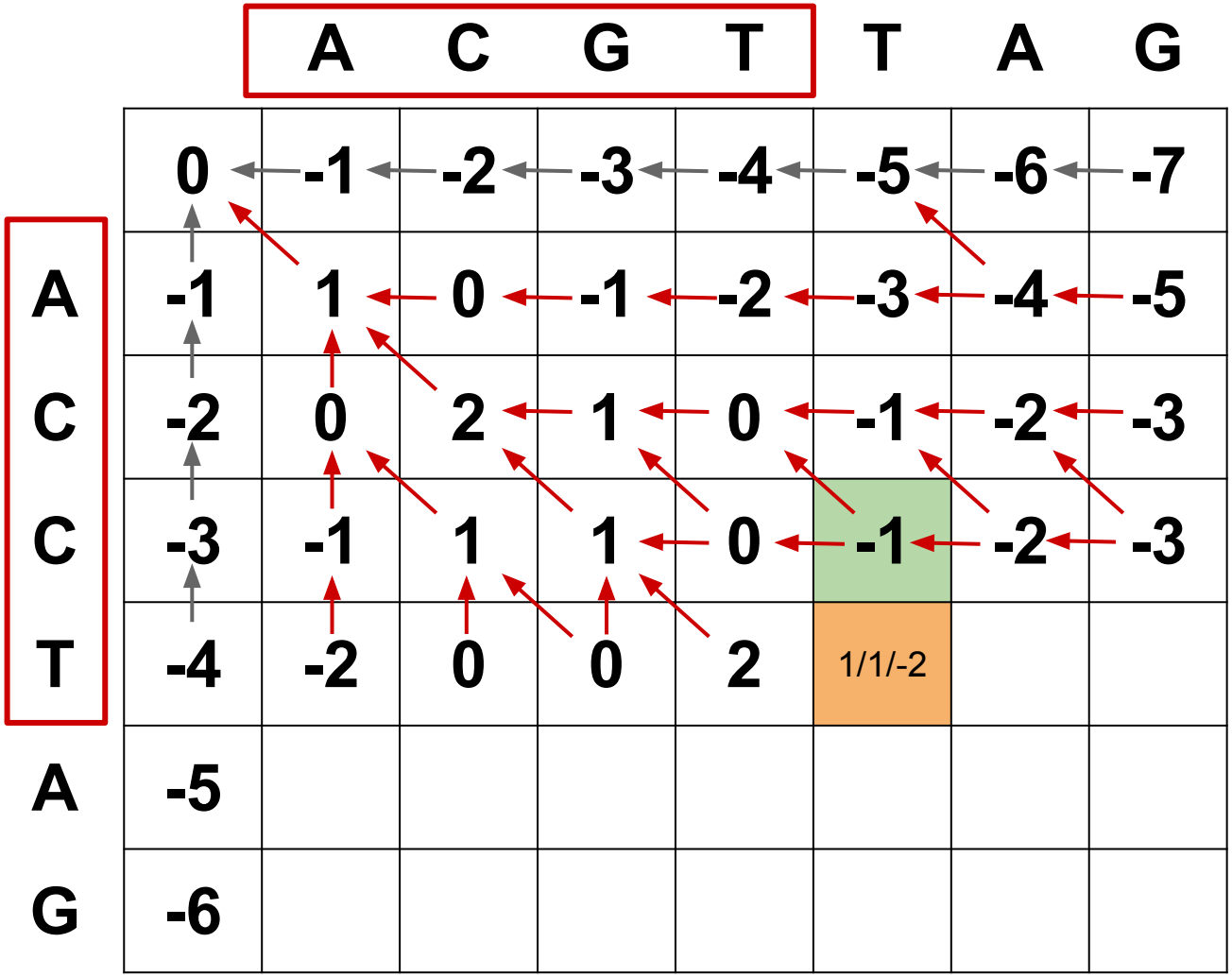
Needleman–Wunsch algorithm



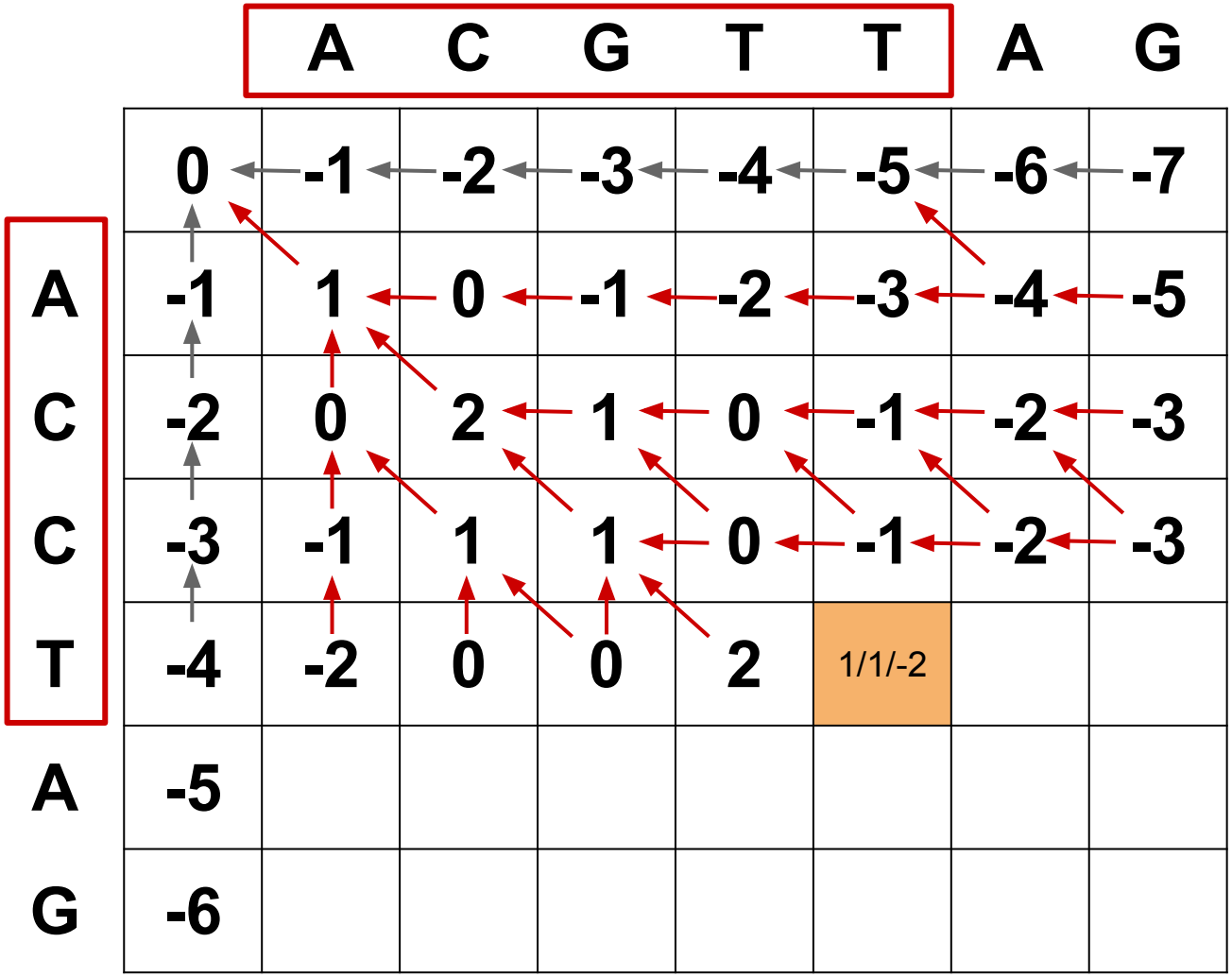
Needleman–Wunsch algorithm



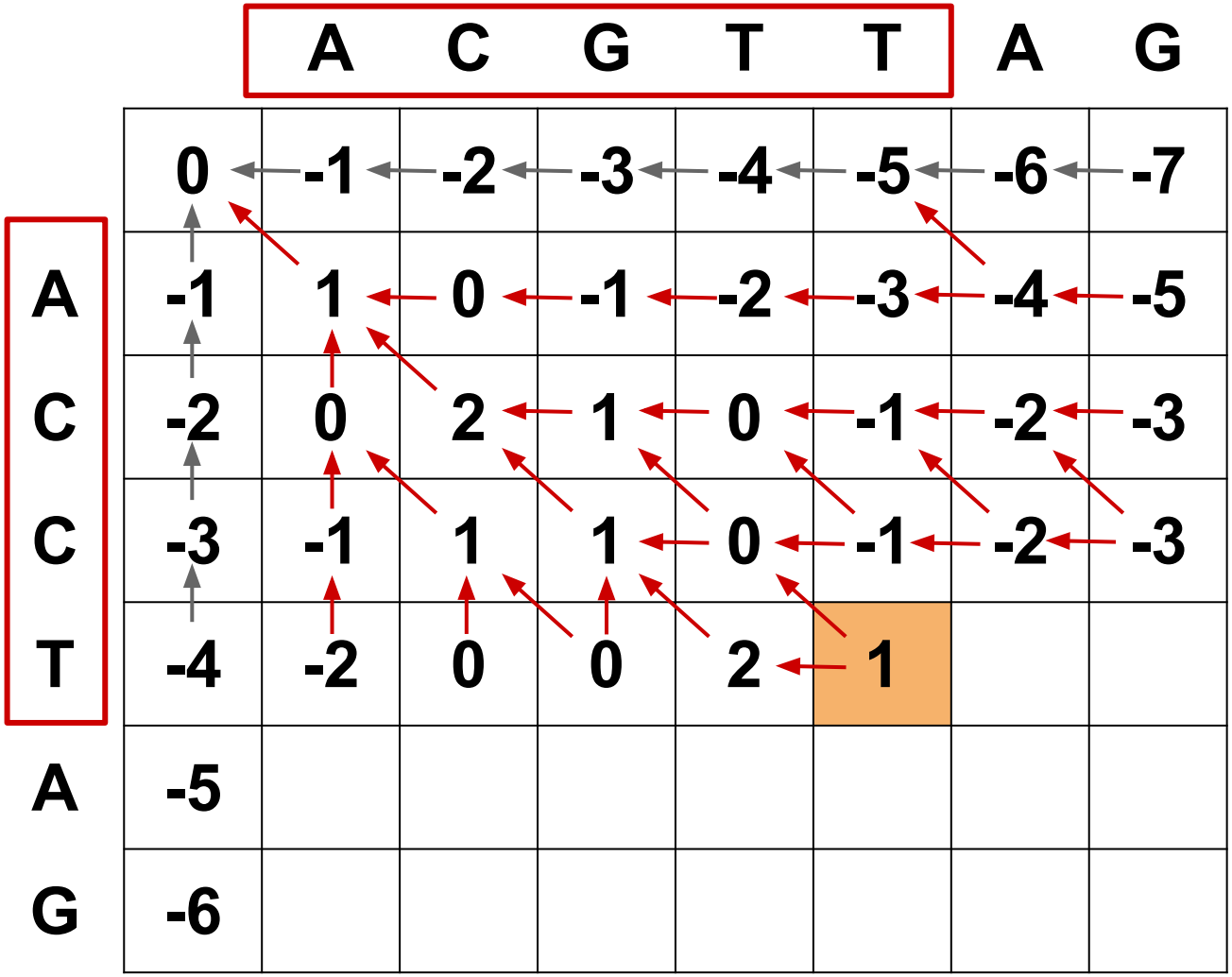
Needleman–Wunsch algorithm



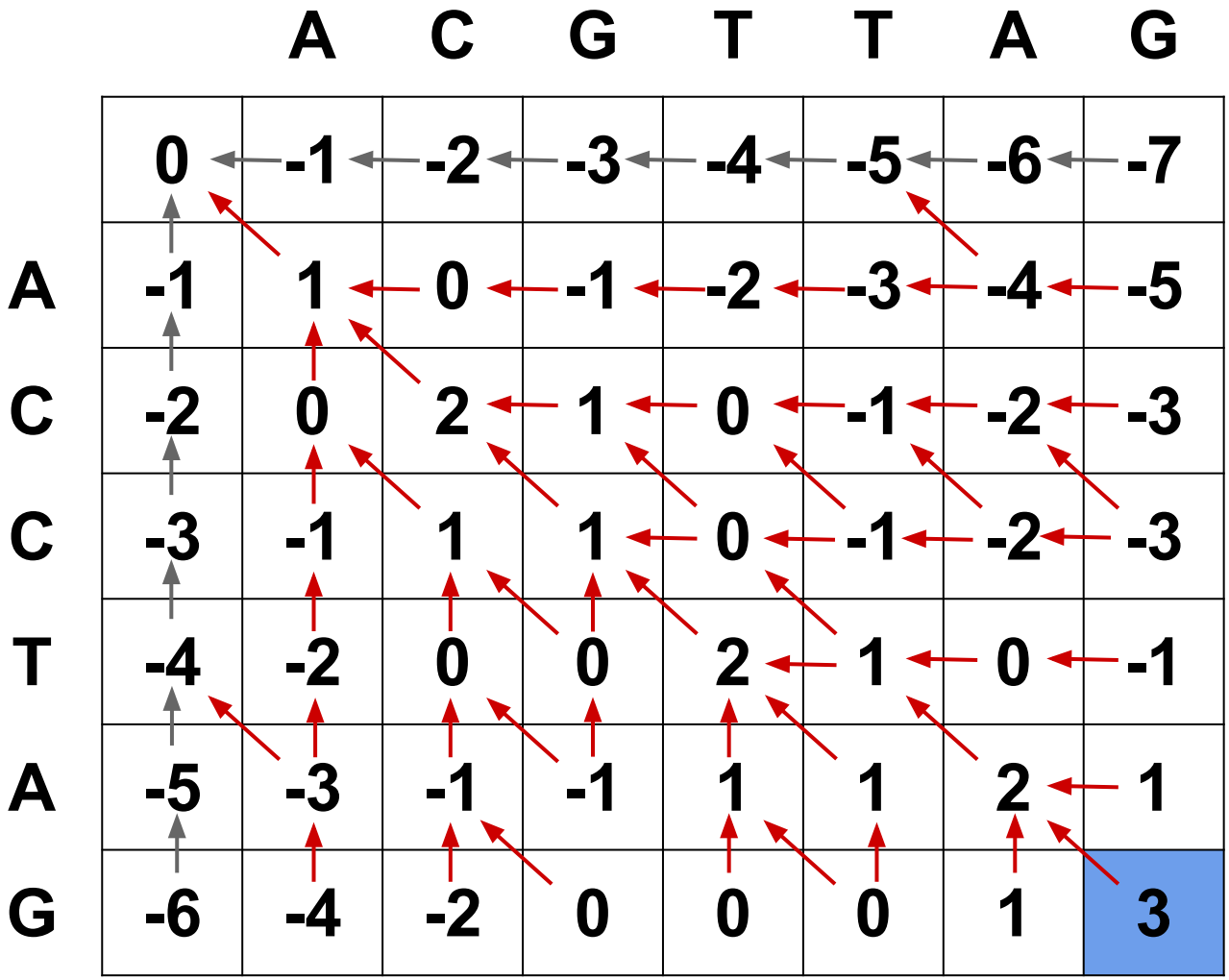
Needleman–Wunsch algorithm



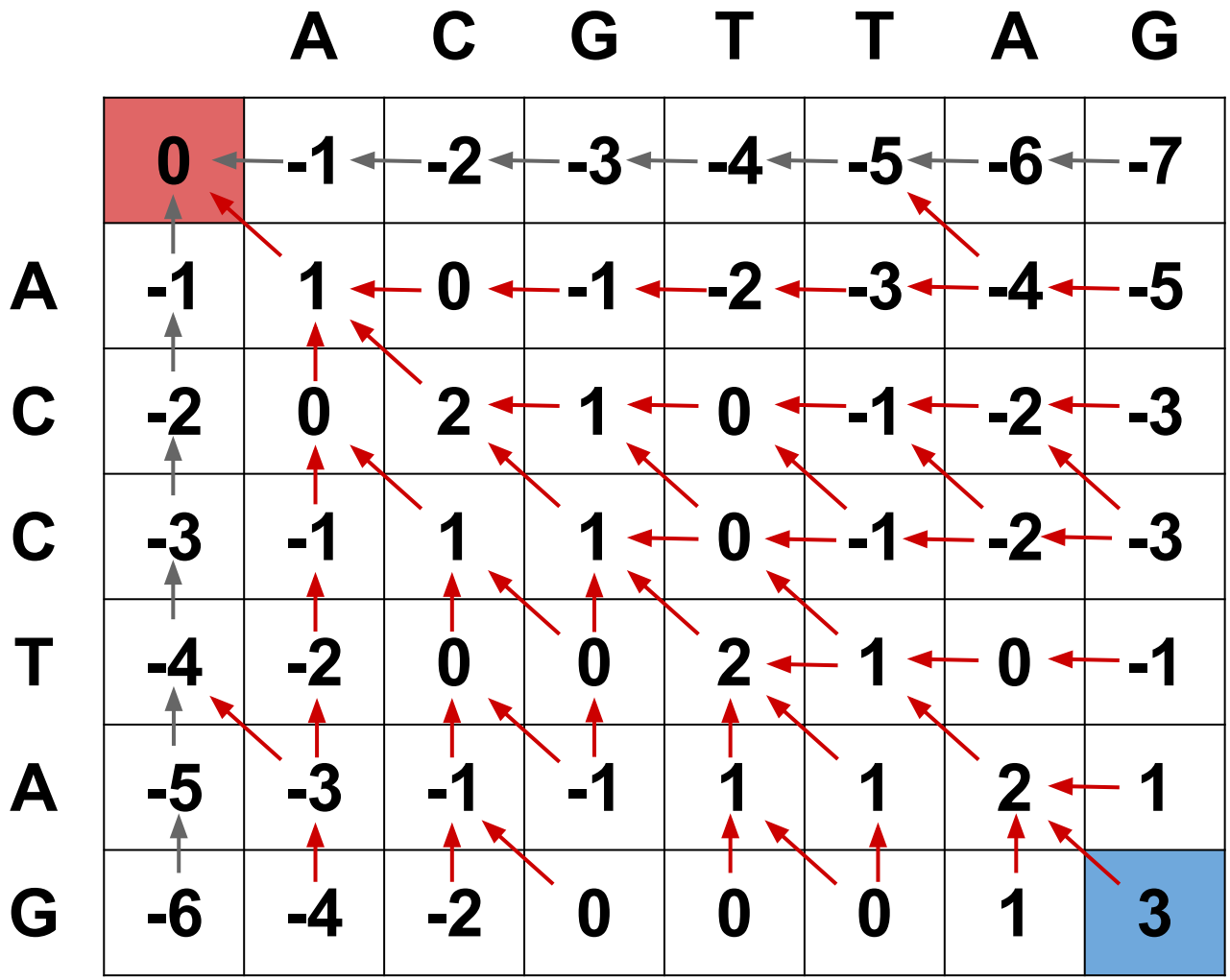
Needleman–Wunsch algorithm



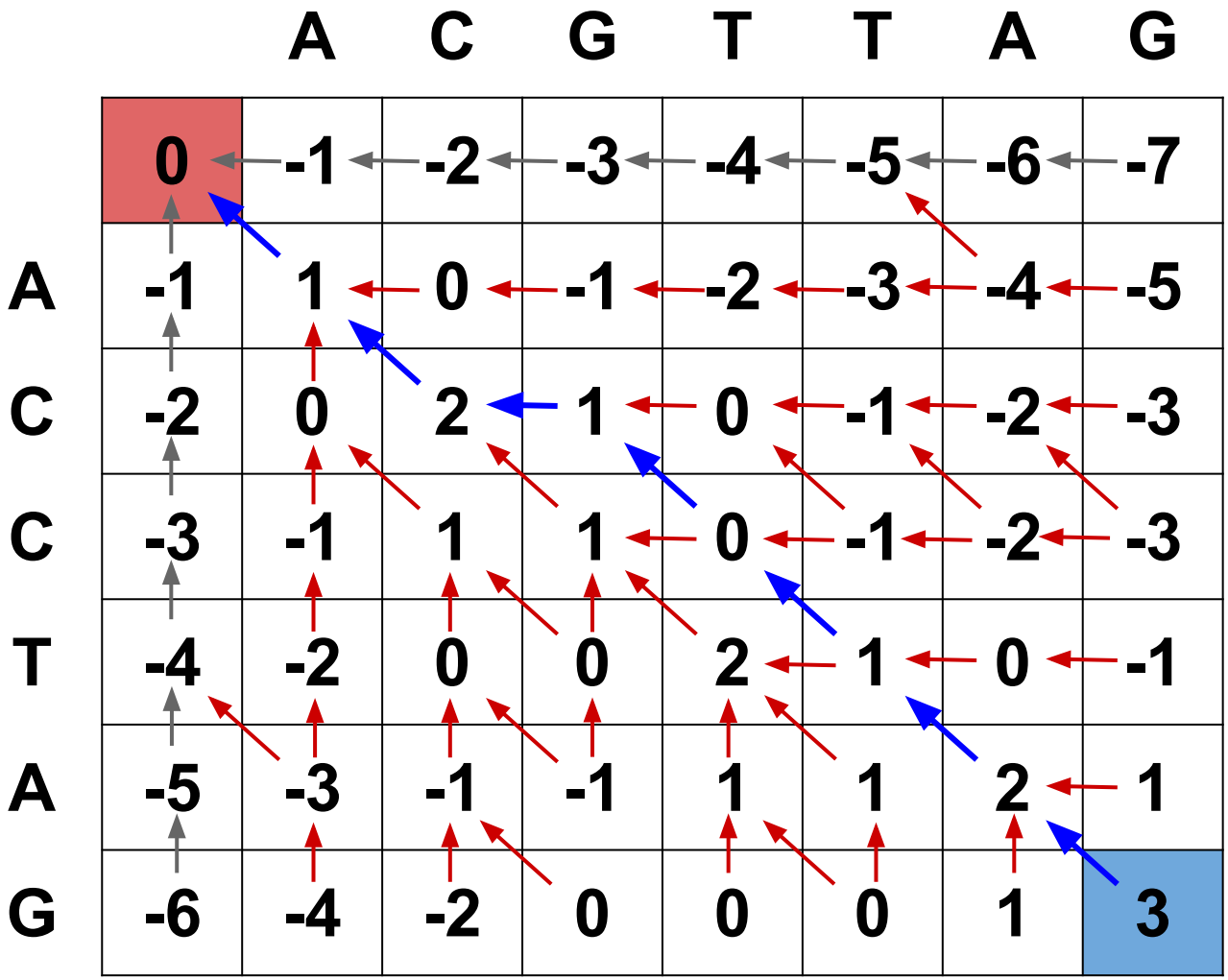
Needleman–Wunsch algorithm



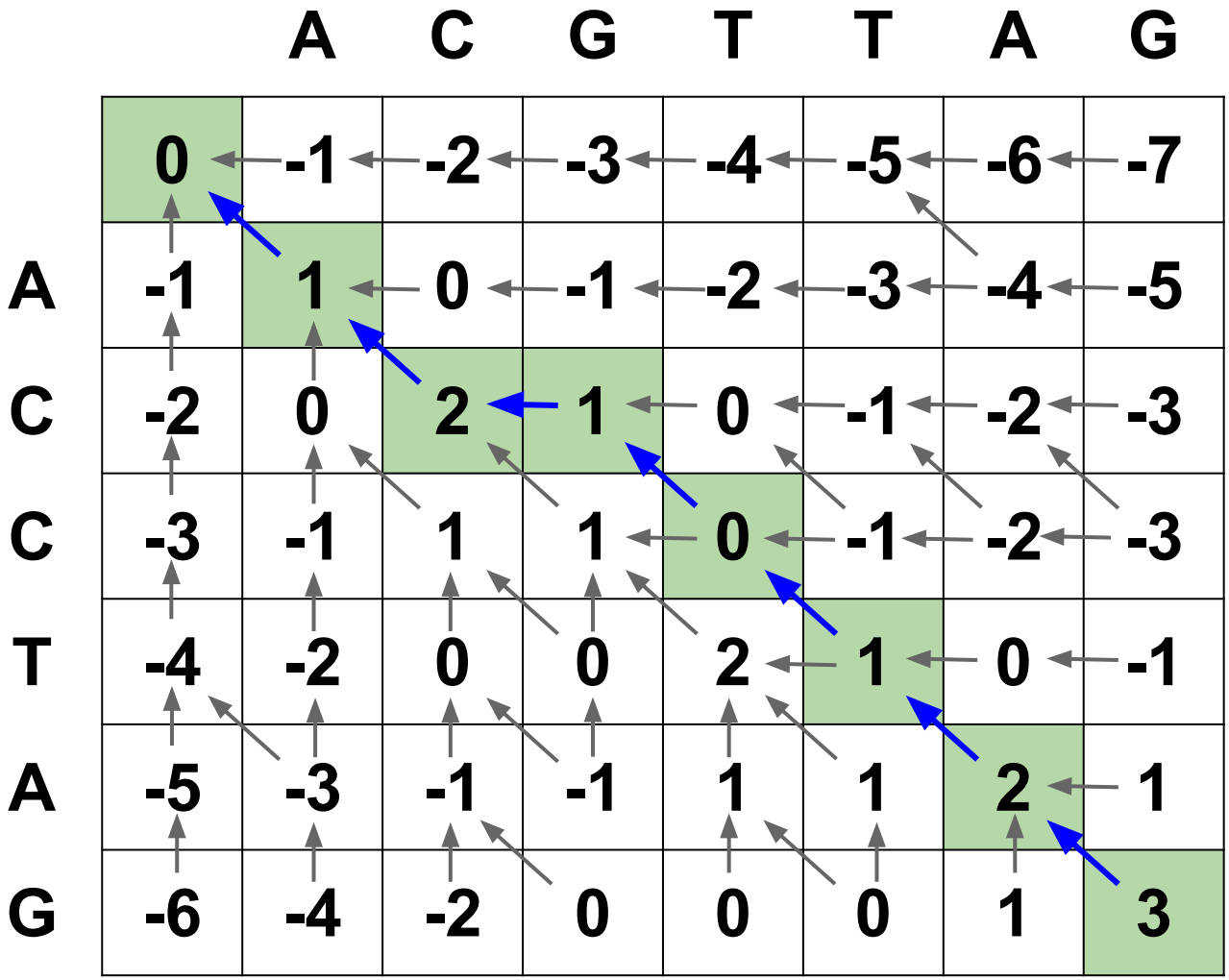
Needleman–Wunsch algorithm



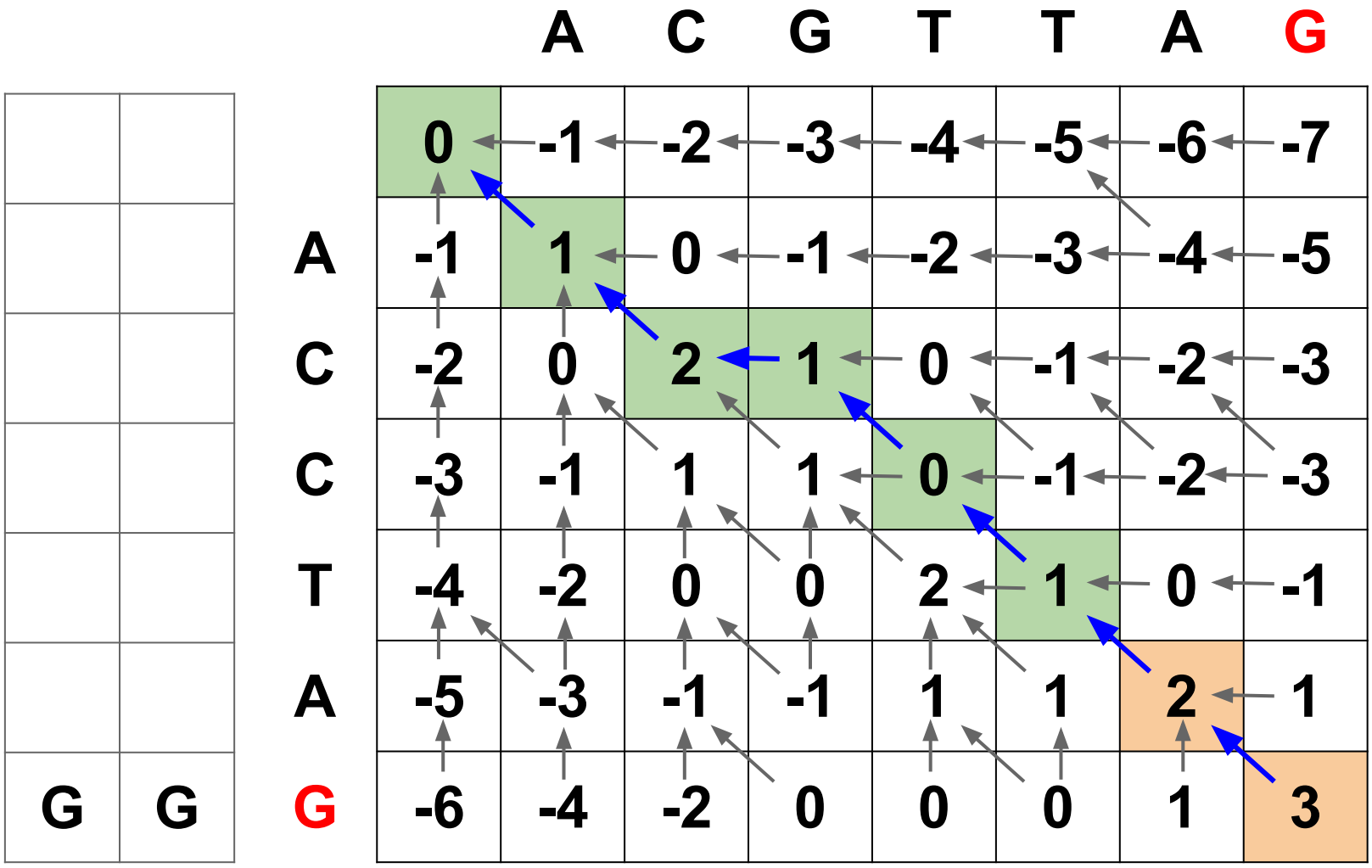
Needleman–Wunsch algorithm



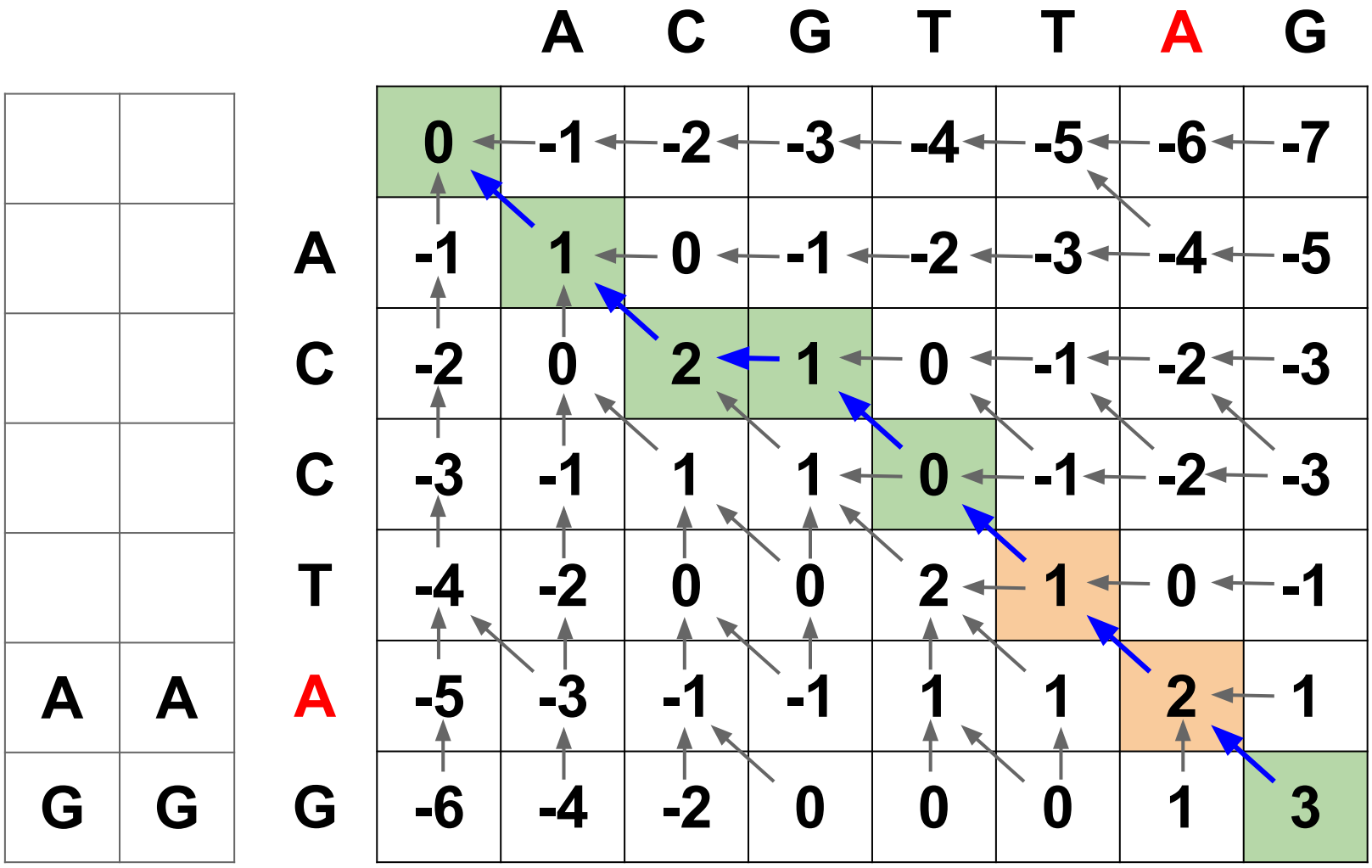
Needleman–Wunsch algorithm



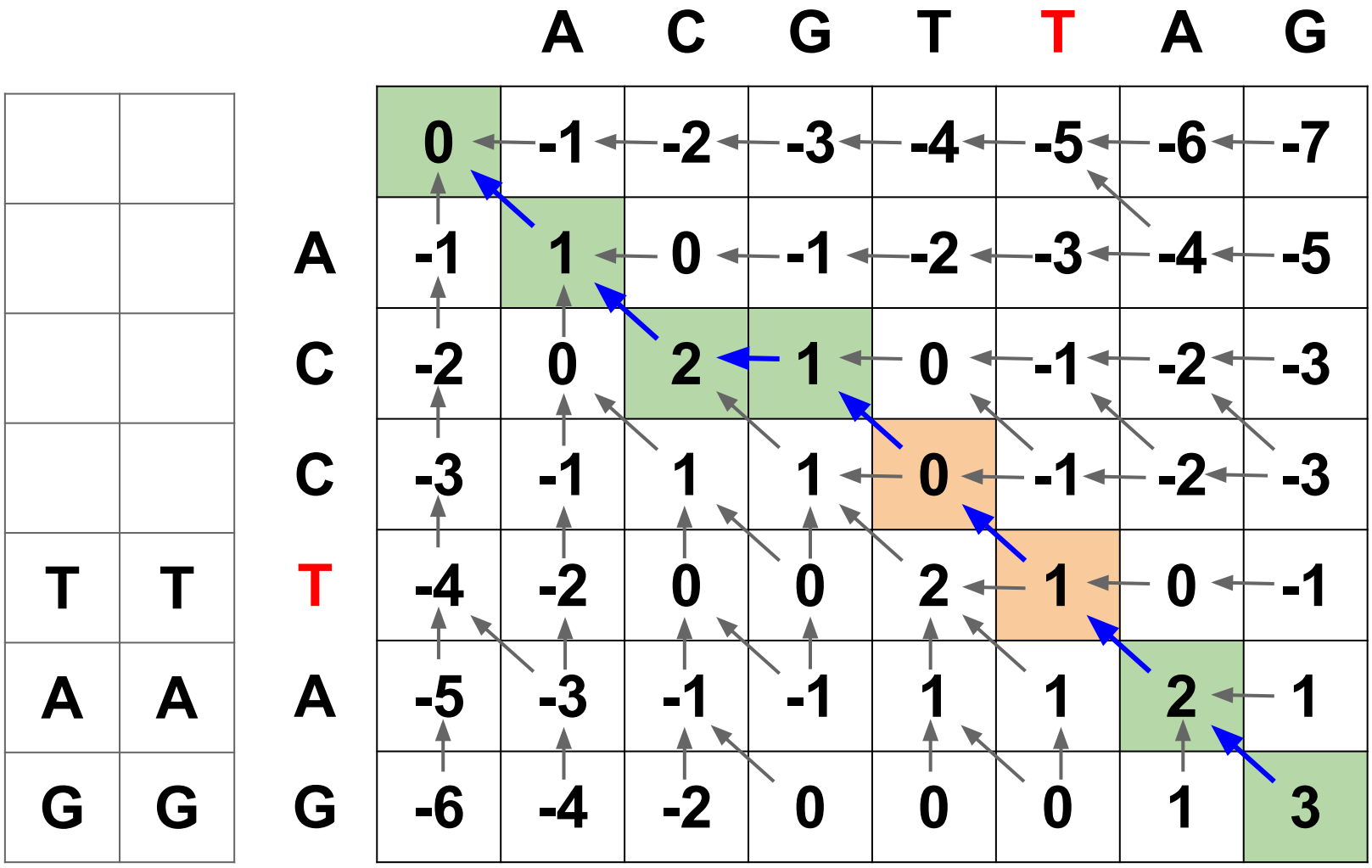
Needleman–Wunsch algorithm



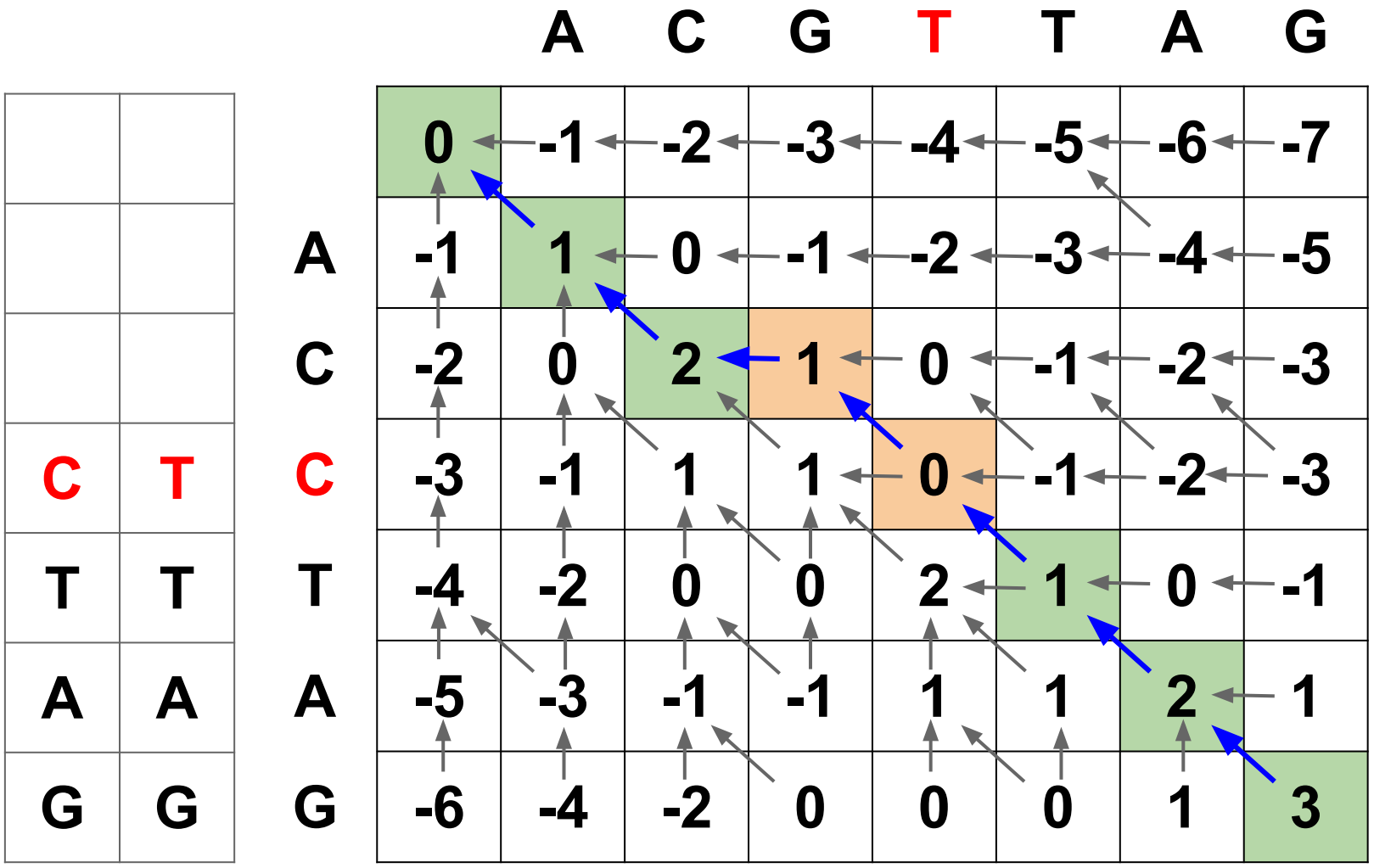
Needleman–Wunsch algorithm



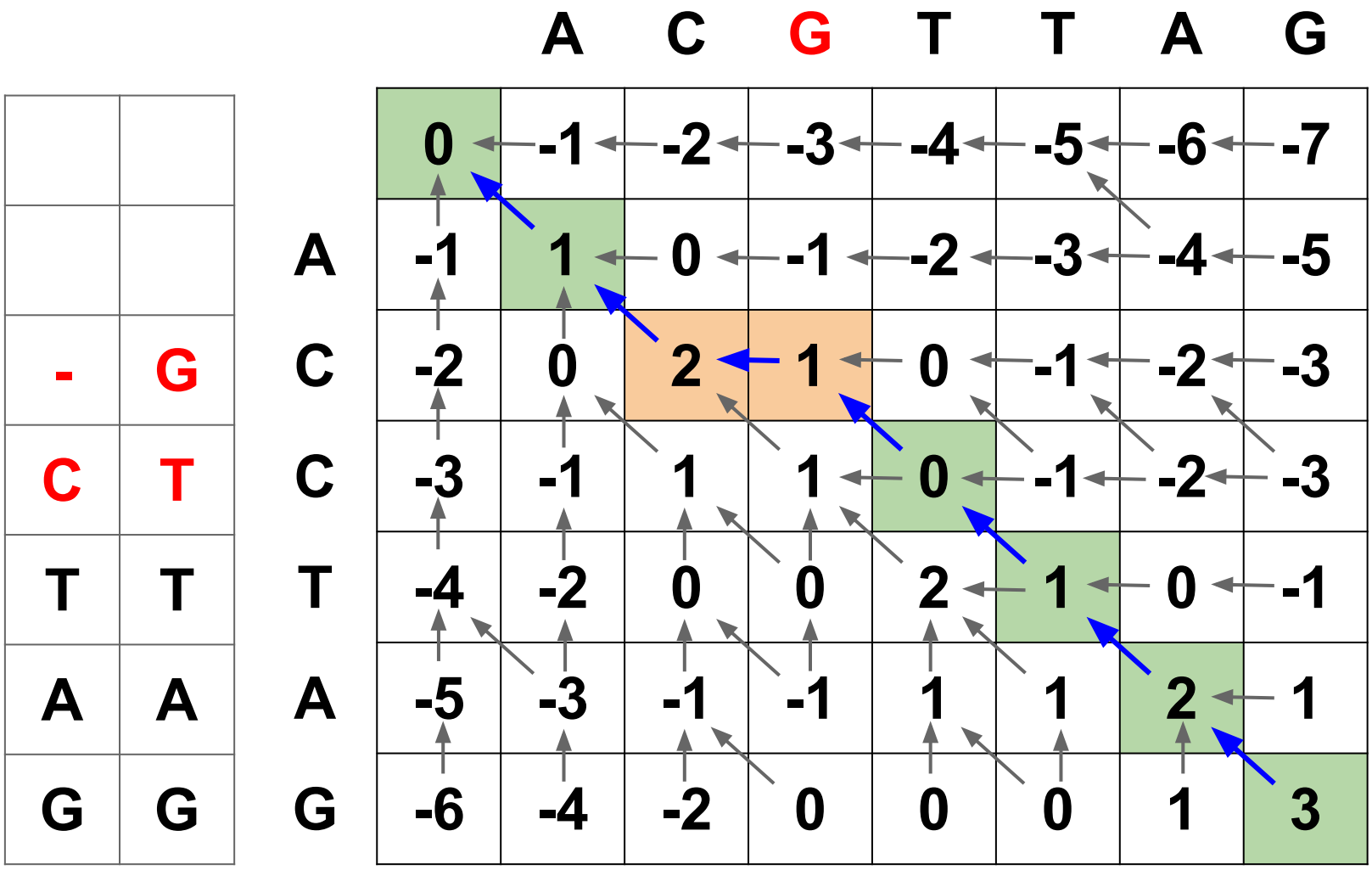
Needleman–Wunsch algorithm



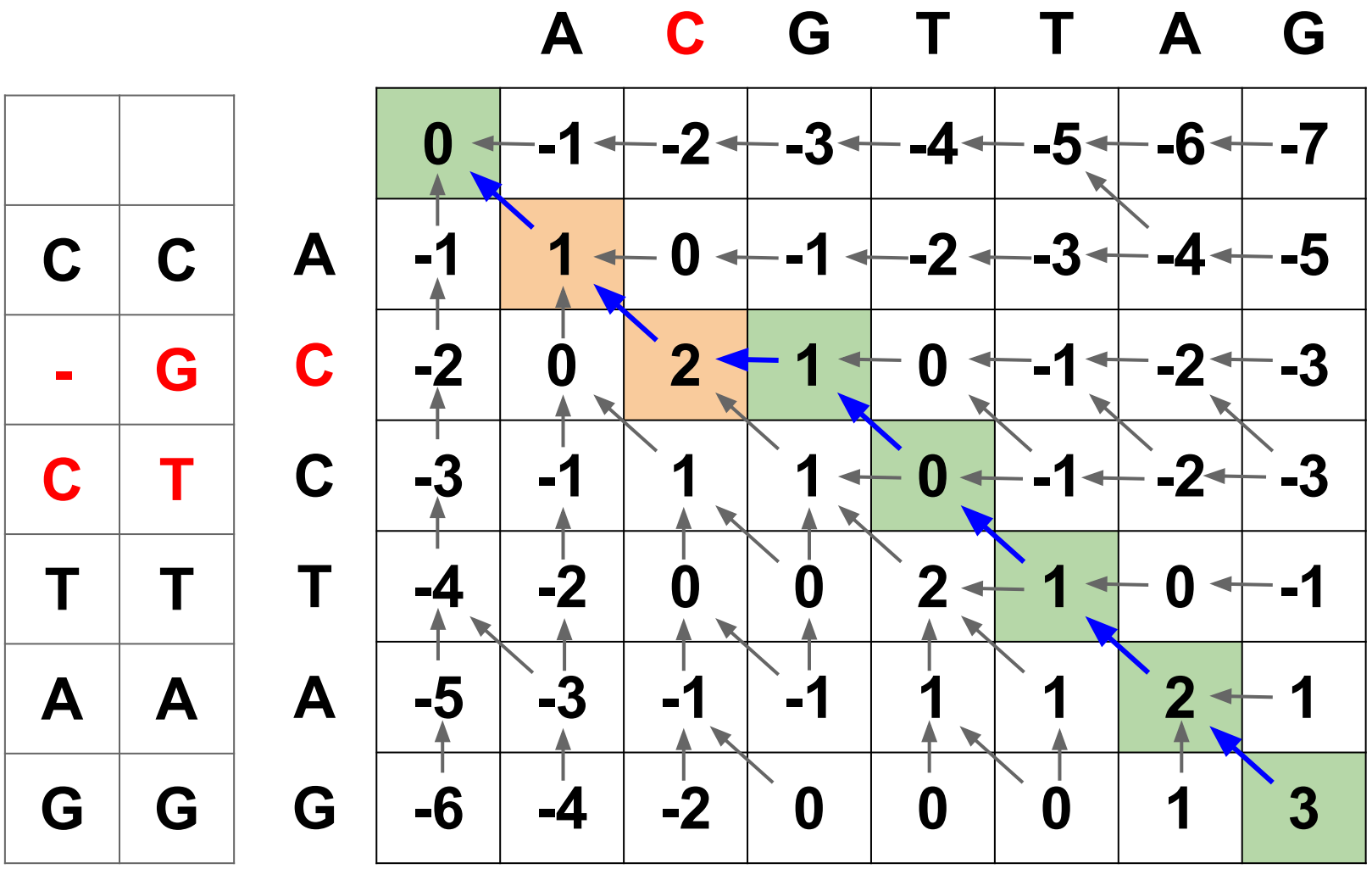
Needleman–Wunsch algorithm



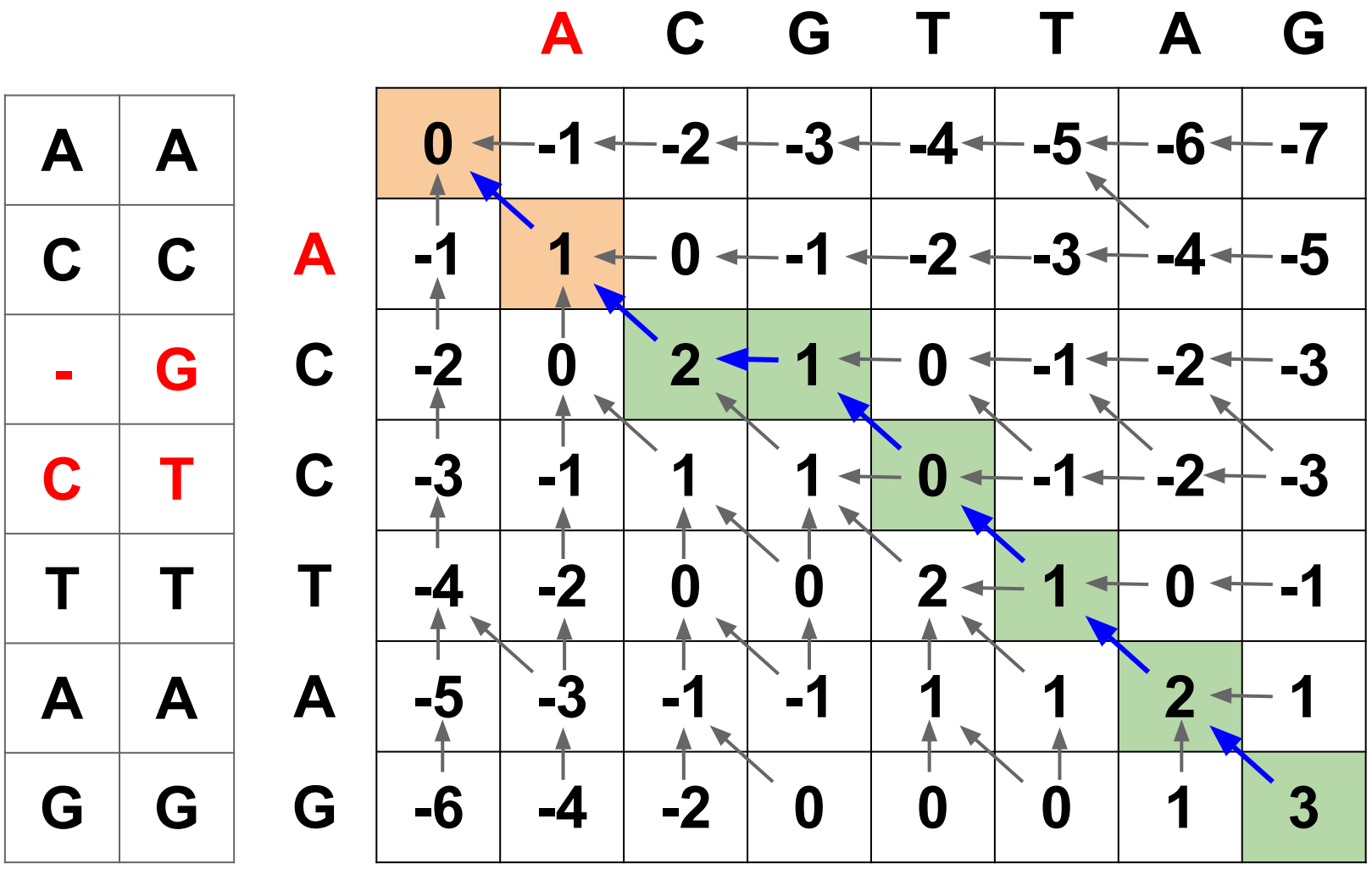
Needleman–Wunsch algorithm



Needleman–Wunsch algorithm



Needleman–Wunsch algorithm



Needleman–Wunsch algorithm

A C G T T A G

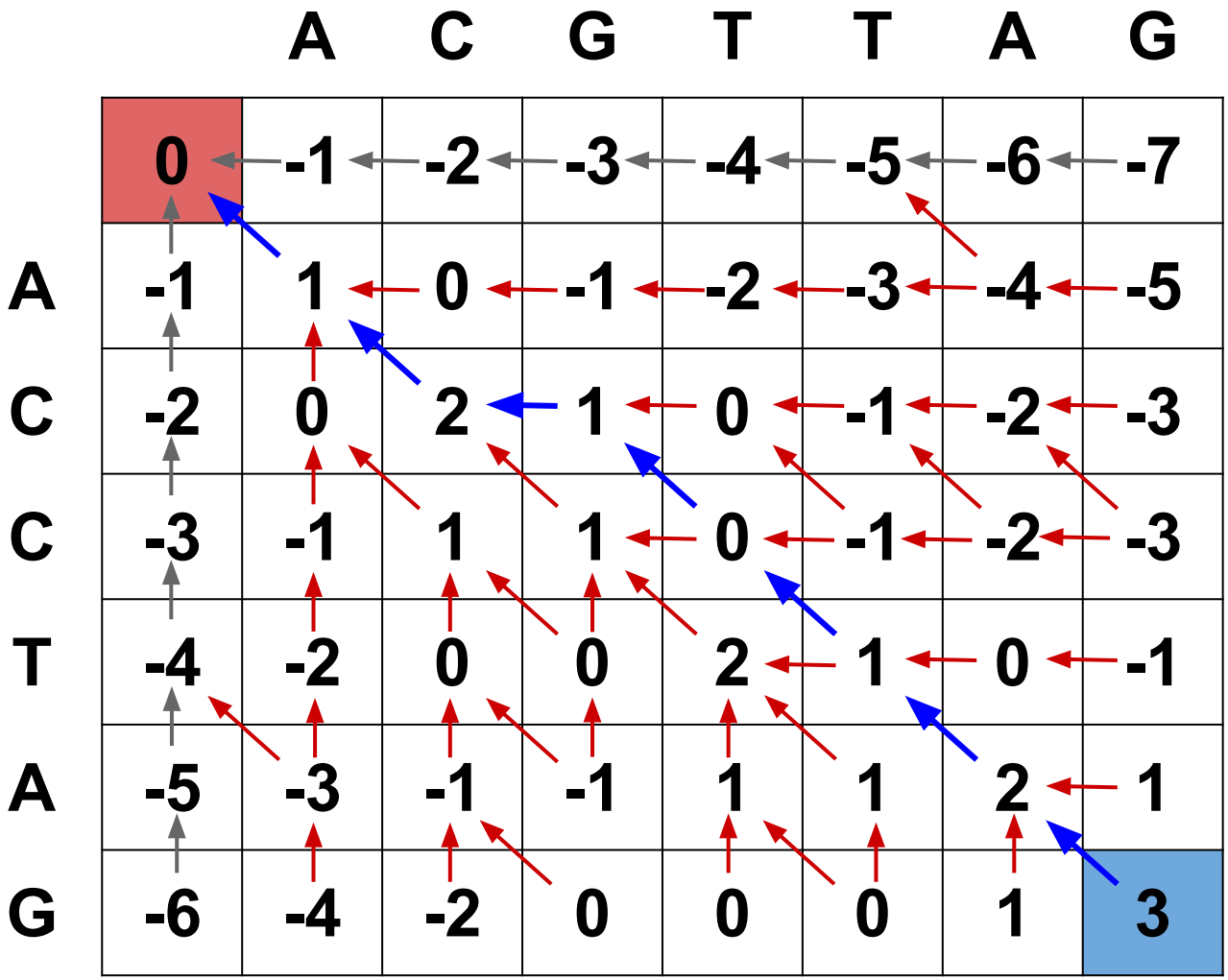
A C C T - A G

A C G T T A G

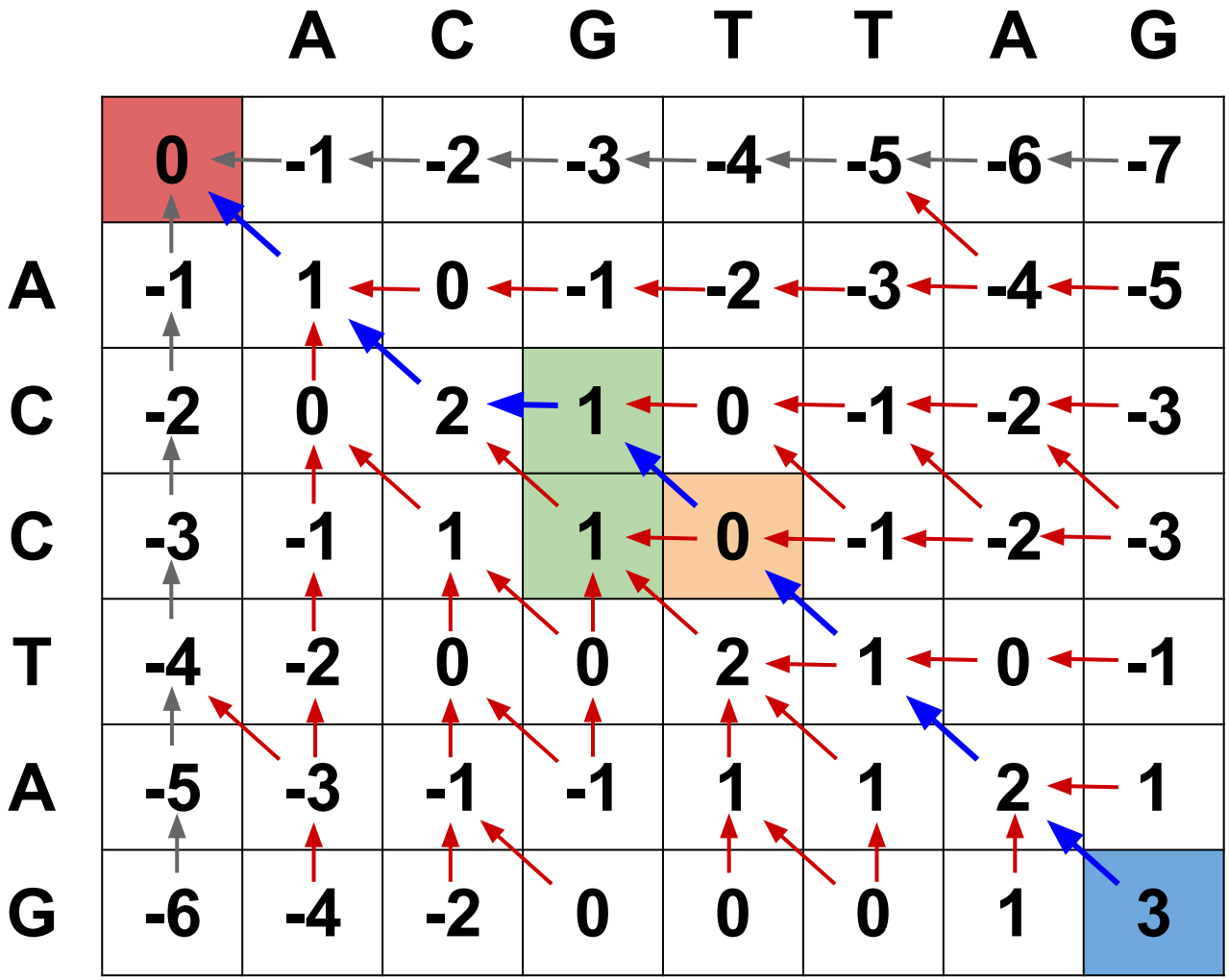
A C - C T A G

Score(ACGTTAG, ACCTAG) = 3

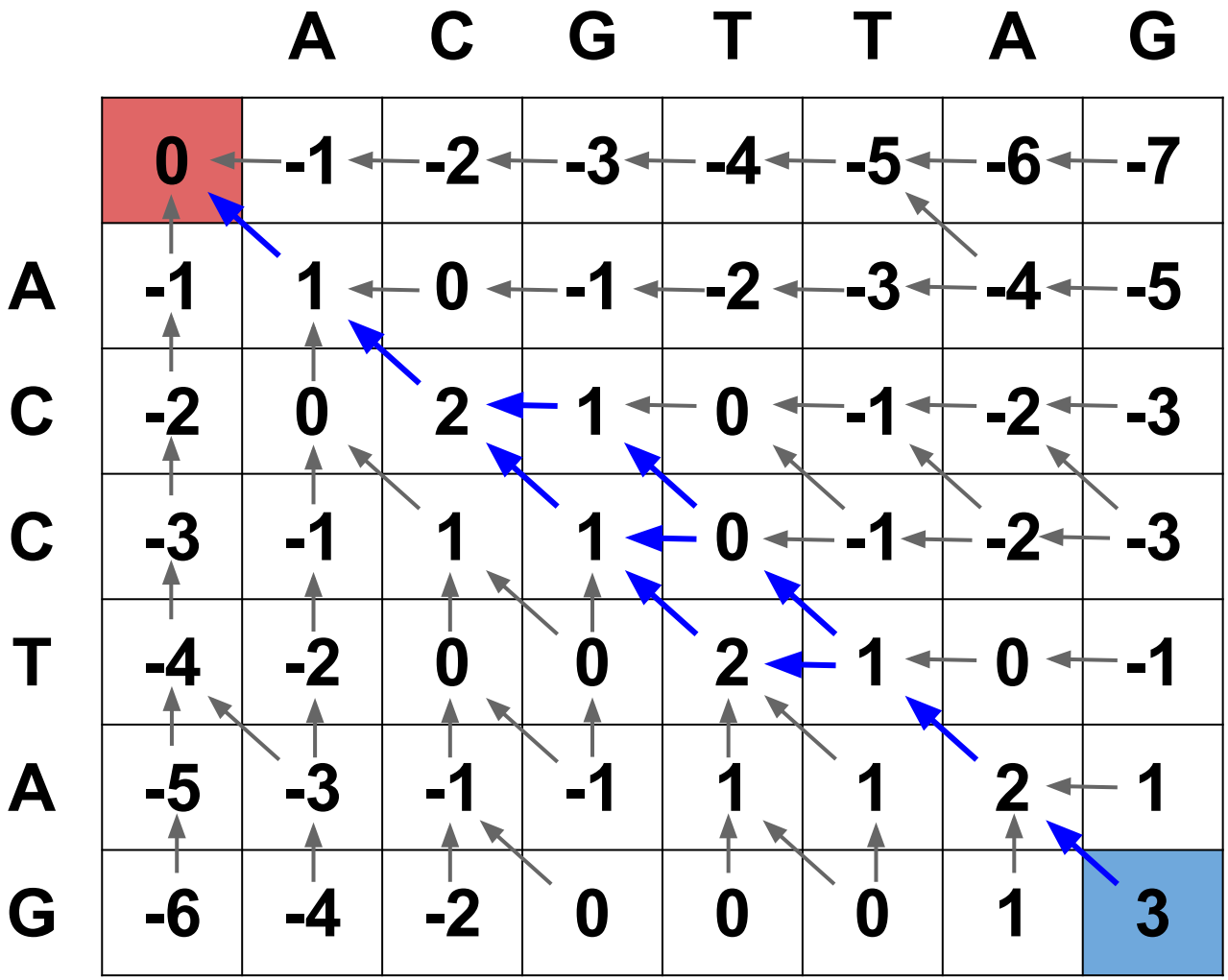
Needleman–Wunsch algorithm



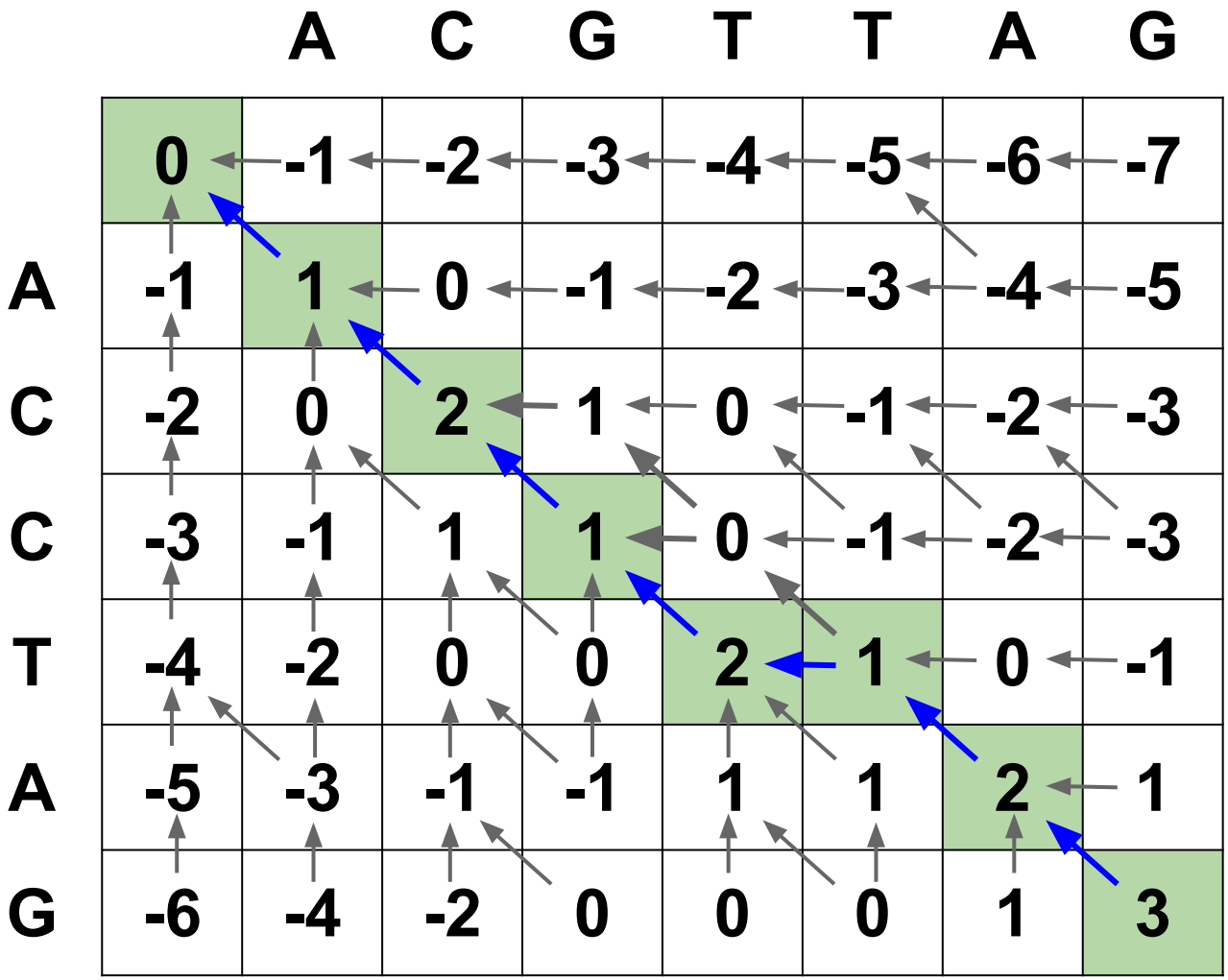
Needleman–Wunsch algorithm



Needleman–Wunsch algorithm



Needleman–Wunsch algorithm



Needleman–Wunsch algorithm

Time =

Memory =

Needleman–Wunsch algorithm

$$\text{Time} = O(n^2)$$

$$\text{Memory} = O(n^2)$$

Problem solving

stepik.org/56393/

Problem 0

Parse FASTA format from stdin

```
>Header_1
ACGGCTTGACGGT
>Header_2
ATACAATCCGCGAACTAAGCGGGA
....
```

Report total number of sequences and bases to stdout

Hint: use BioPython `from Bio import SeqIO`

Problem 1a

Implement Needleman-Wunsch algorithm

- Input: FASTA with 2 sequences
- Output: alignment score

Problem 1b

Implement Needleman-Wunsch algorithm

- Input: FASTA with 2 sequences
- Output: alignment score and alignment itself

A C G T T A G

A C - C T A G

Problem 1b*

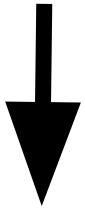
Implement Needleman-Wunsch algorithm

- Input: FASTA with 2 sequences
- Output: alignment score and alignment itself; if multiple alignments exist, output all

Global alignment

ATCTGATCGCCA

ATACAATCCGCGAACTAAGCGGGA



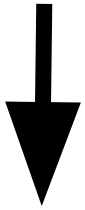
AT- C - -T - - G - - A - - T - - - CGCCA

ATACAATCCGCGAACTAAGCGGGA

Local alignment

ATCTGATCGCCA

ATACAATCCGCGAACTAAGCGGGA



AT CTGATCGCCA

ATACAATCCGC-

GAACTAAGCGGGA

Problem 2

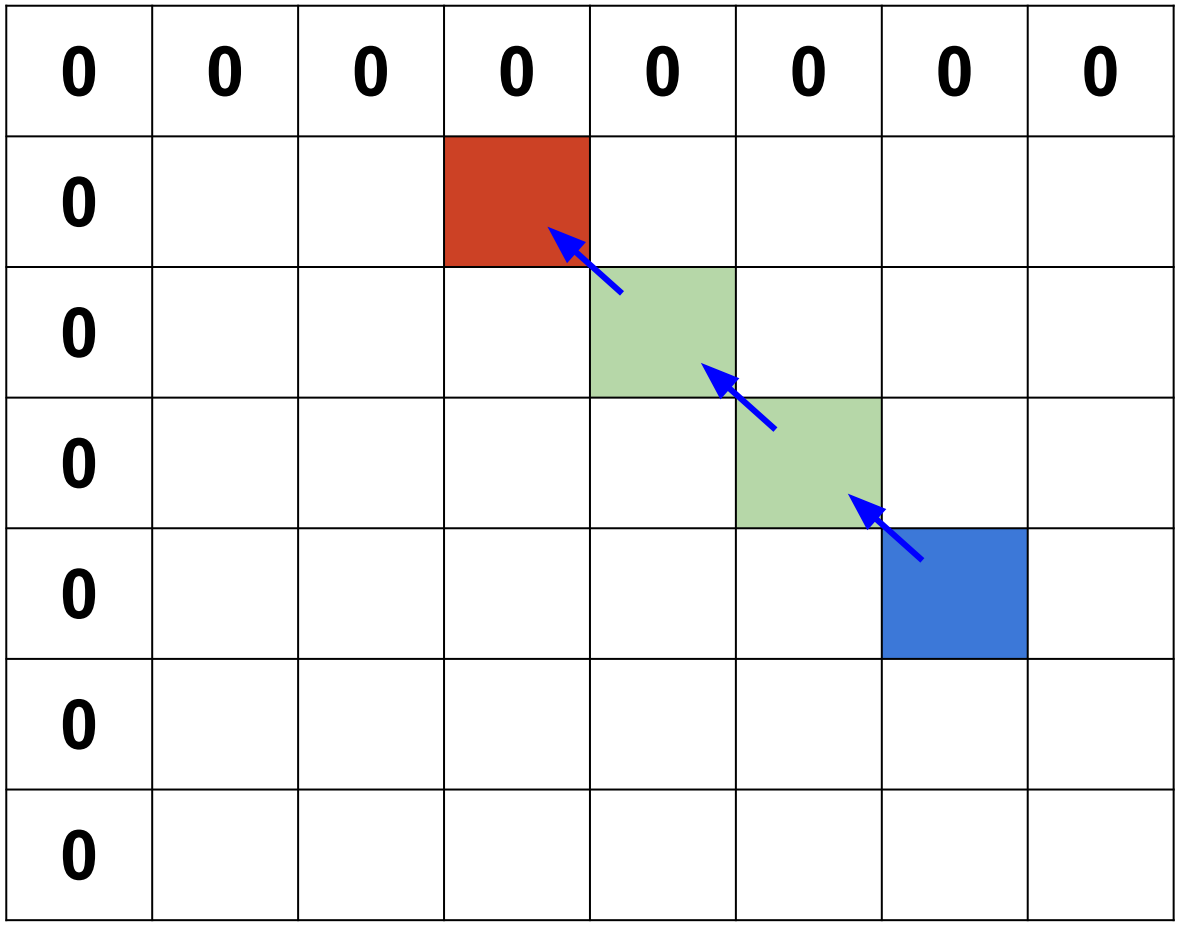
Implement Smith-Waterman algorithm

- Input: FASTA with 2 sequences
- Output: alignment score and alignment itself

Smith-Waterman algorithm

0	0	0	0	0	0	0	0
0							
0							
0							
0							
0							
0							

Smith-Waterman algorithm



Scoring matrix

In real life different substitutions may have different probability

	A	C	G	T
A	3	-3	-2	-3
C	-3	3	-3	-2
G	-2	-3	3	-3
T	-3	-2	-3	3

Problem 3

Implement SW or NW algorithm with scoring matrix

- Input: Scoring matrix (4 rows with 4 numbers each) and FASTA with 2 sequences
- Output: alignment score and alignment itself

Affine gap penalty

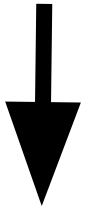
ACGCCTTAACGTGCA

ACGCCTACCGGTGCA

Affine gap penalty

ACGCCTTAACCGGTGCA

ACGCTACGTGCA



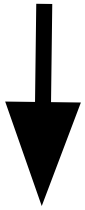
A C G C C T T A A C C G G T G C A

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

Affine gap penalty

ACGCCTTAACCGGTGCA

ACGCTACGTGCA



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

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

Affine gap penalty

- Scoring function S(a, b)
- Gap open score S_{op}
- Gap extension score S_{ex}

Affine gap penalty

- 3 matrices
  - **M**atches/mismatches (diagonal arrows)
  - **I**nsertions (vertical arrows)
  - **D**eletions (horizontal arrows)
- Moving from **M** to **D** or **I** adds gap open penalty S_{op}

Affine gap penalty

- $I[i, j] = \max \{ I[i - 1, j] - S_{ex}, M[i - 1, j] - S_{op} \}$
- $D[i, j] = \max \{ D[i, j - 1] - S_{ex}, M[i, j - 1] - S_{op} \}$
- $M[i, j] = \max \{ D[i, j], I[i, j], M[i - 1, j - 1] + S(A[i], B[j]) \}$

Problem 4

Implement affine gap penalty algorithm

- Input: 1 line with 4 numbers (match, mismatch, gap open and gap extend scores) and FASTA with 2 sequences
- Output: alignment score and alignment itself

Problem 5

Given 2 long sequences ($L > 10^5$) that have less than k errors ($k << L$). Quickly compute

a. Alignment score
b. Alignment itself

If there are more than k errors, print “NULL”

- Input: integer k and FASTA with 2 sequences
- Output: alignment score and alignment itself