

MA3259 Lecture 2

# Dynamic Programming Algorithms for Pairwise Alignment

LX Zhang  
Department of Mathematics  
National University of Singapore  
matzlx@nus.edu.sg

# 1. Quick Review: Alignment Definition



An alignment among  $k$  sequences is a  $k$ -row matrix such that

- The row  $j$  contains the  $j$ -th sequence and between two consecutive residues there may be one or more ‘-’s;
- Each column contains at least one residue.

# Scoring Scheme Used in this Lecture

- Scores for aligned residues and gaps form a basic scoring scheme:

	A	G	C	T	-
A	8	-5	-5	-5	-3
G		8	-5	-5	-3
C			8	-5	-3
T				8	-3

a g t c t c c  
a - t c - a c

has score

$$8 + (-1) + 8 + 8 + (-1) + (-5) + 8 = 25$$

Match scores 8;  
Mismatch scores -5  
Indel scores -3

## 2. Algorithms for Aligning Two Sequences

---

**Pairwise (Global) Alignment Problem:**

**Input:** Two sequence S and T and a scoring scheme;

**Question:** Find an optimal alignment of S and T that has the highest alignment score.

~~A straightforward approach is to enumerate and check alignments one by one~~

## 2.1. The number of pairwise alignments

---

- Alignments between two 1-character sequences:

X	--	X	X	--
Y	Y	--	--	Y

- Alignments between two 2-char sequences *AB* and *CD*:

A B	--	A B	--	A B	A B --	A -- B	A B --
C D	C D --	C -- D	C -- D	C D --	-- C D		
A -- B	A B -- --	A -- B --					
-- C D	-- -- C D	-- C -- D					

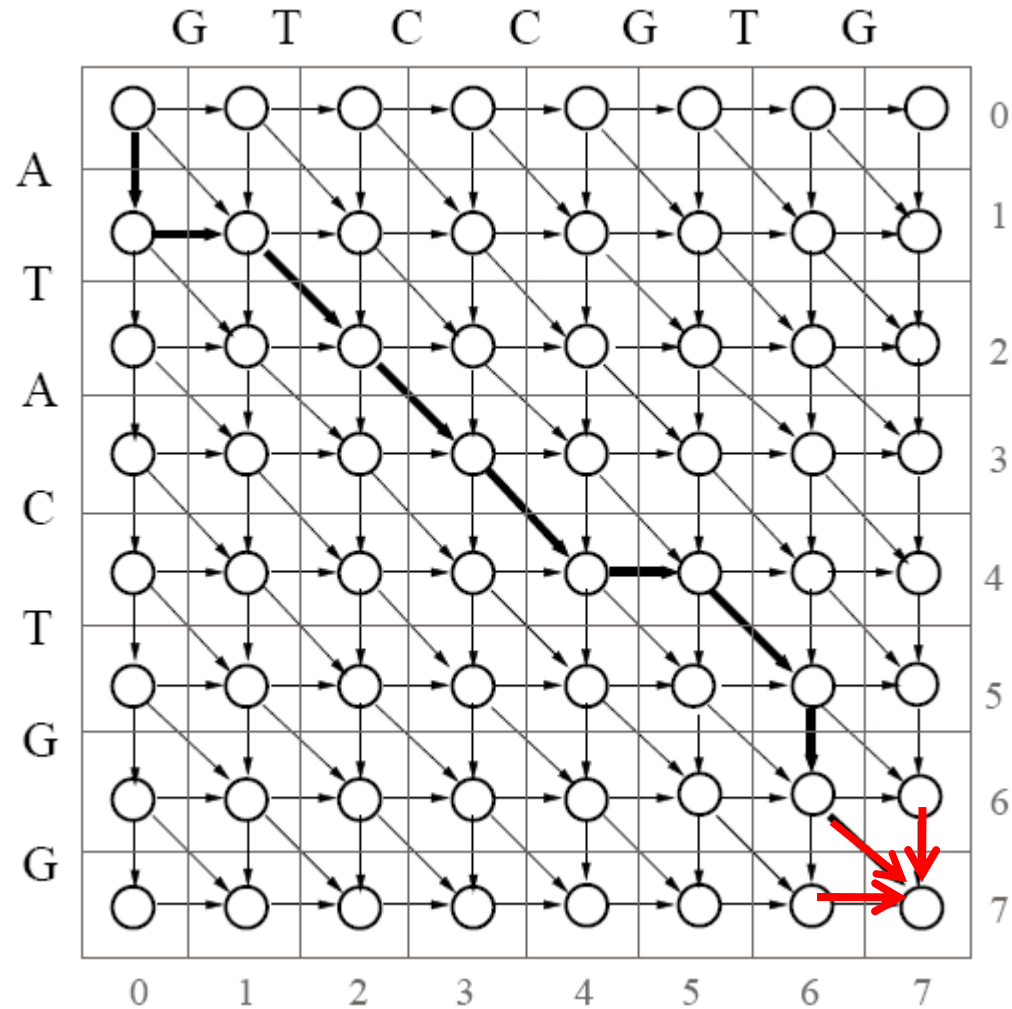
And 4 more

Let  $N(j, k)$  denote the number of possible alignments between sequences of length  $m$  and  $n$ , respectively.

The empty sequence is of length 0.

Every path must enter the rightmost vertex at the bottom by one of the three highlight arcs.

There is a unique alignment between the empty and another other seq.



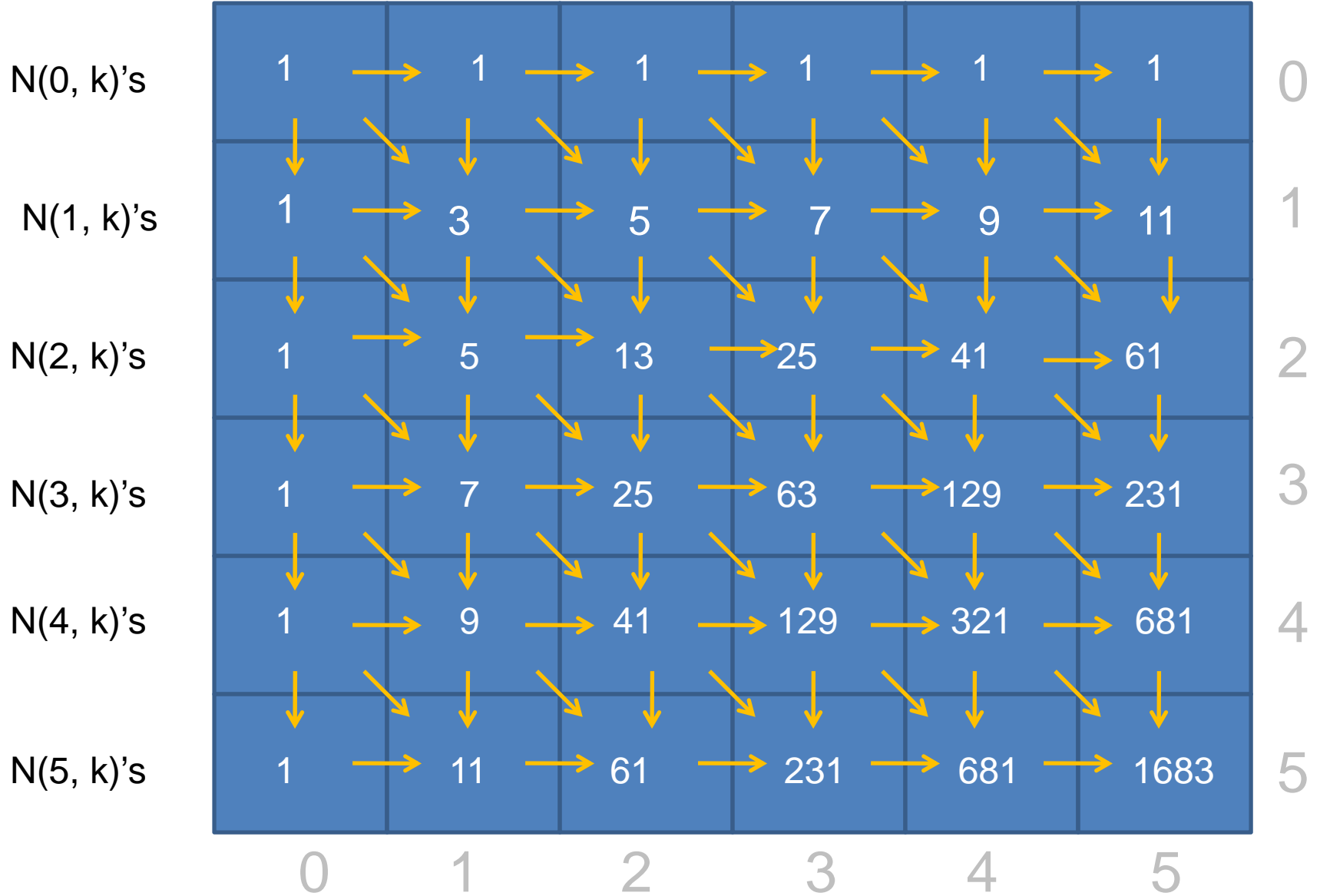
### Theorem:

$$N(0, k) = 1; \quad N(j, 0) = 1;$$

$$N(j, k) = N(j-1, k) + N(j, k-1) + N(j-1, k-1);$$

$N(0, k) = 1$ ;  $N(j, 0) = 1$ ;  
 $N(j, k) = N(j-1, k) + N(j, k-1) + N(j, k)$ ;

$N(k, k)$  grows exponentially!  
 $N(8, 8) = 265,729$

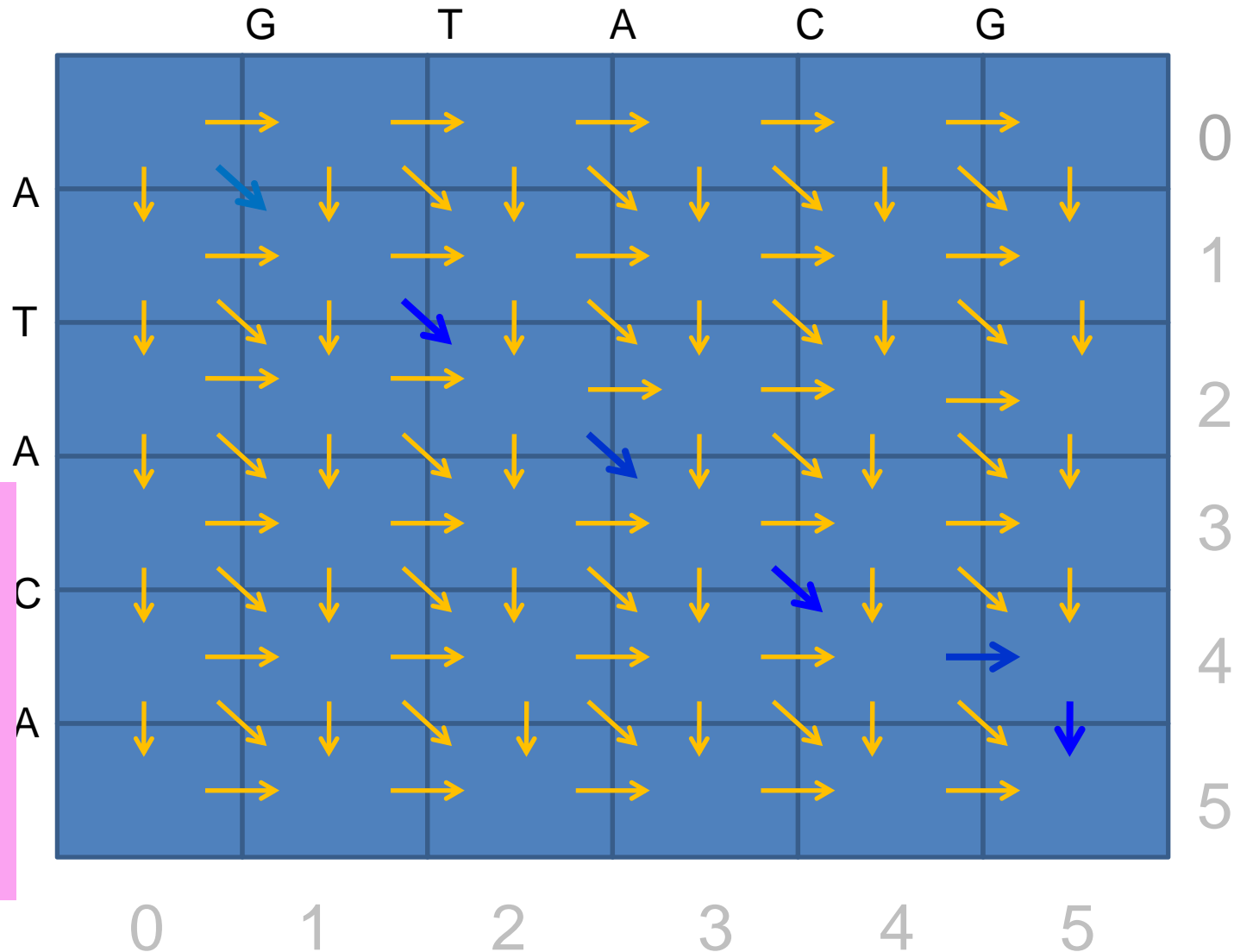


# 2.1 Alignment using Dynamic Programming

Each column of an alignment corresponds to an edge of the corresponding path.

Assign the column score to an edge as weight.

Find an optimal alignment is to find a path with the largest weight.



# Consider two sequences

$$X = x_1 x_2 \dots x_j, \quad Y = y_1 y_2 \dots y_k$$

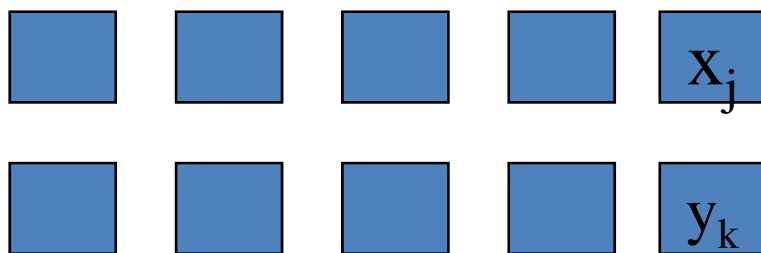
Write their t-char prefixes as

$$X[1, \dots, t] = x_1 x_2 \dots x_t \quad Y[1, \dots, t] = y_1 y_2 \dots y_t$$

Let  $S(j, k)$  be the score of optimally aligning  $X[1, \dots, j]$  and  $Y[1, \dots, k]$ .

Consider one of the best alignments of  $X[1, \dots, j]$  and  $Y[1, \dots, k]$ .

Case 1: The alignment corresponds to a path whose last edge is diagonal.

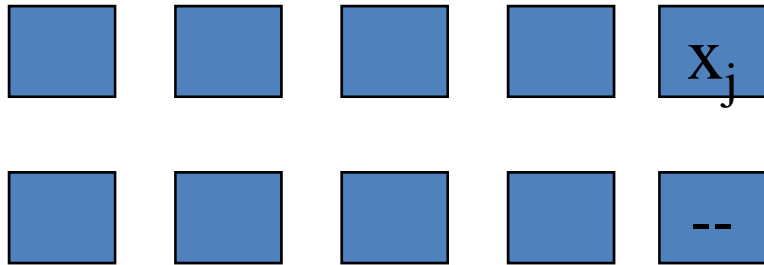


A best alignment of  $X[1, \dots, j-1]$  and  $Y[1, \dots, k-1]$

$$S(j, k) = S(j-1, k-1) + s(x_j, y_k)$$

$S(j, k)$  = optimal score of aligning  $X[1, \dots, j]$  and  $Y[1, \dots, k]$ .

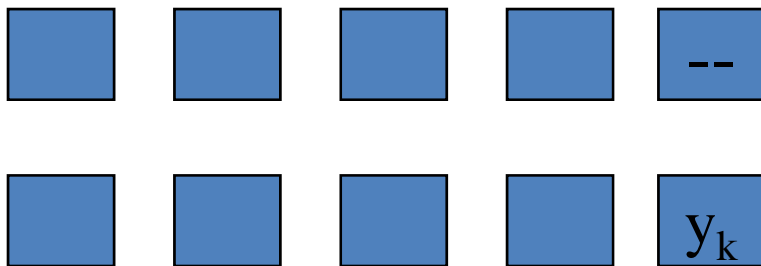
**Case 2:** Correspond to a path whose last edge is vertical



$$S(j, k) = S(j-1, k) + s(x_j, -)$$

An optimal alignment of  $X[1, \dots, j-1]$  and  $Y[1, \dots, k]$

**Case 3:** Correspond to a path whose last edge is horizontal



$$S(j, k) = S(j, k-1) + s(-, y_k)$$

An optimal alignment of  $X[1, \dots, j]$  and  $Y[1, \dots, k-1]$

$S(j, k)$  = the score of optimally aligning  $x[1, \dots, j]$  and  $y[1, \dots, k]$ .

$$S(j, k) = \max \begin{cases} S(j-1, k) + s(x_j, -); \\ S(j, k-1) + s(-, y_k); \\ S(j-1, k-1) + s(x_j, y_k). \end{cases}$$

If we know the optimal alignments for  $X[1, \dots, j-1]$  and  $Y[1, \dots, k]$ ,  $X[1, \dots, j]$  and  $Y[1, \dots, k-1]$ ,  $X[1, \dots, j-1]$  and  $Y[1, \dots, k-1]$ , we can find an optimal alignments for  $X[1, \dots, j]$  and  $Y[1, \dots, k]$ .

$S(0, k)$  = the score of optimally aligning an empty  $X$  to  $Y$   
 $= s(-, y_1) + s(-, y_2) + s(-, y_3) + \dots + s(-, y_k)$ ;

$S(j, 0)$  = the score of optimally aligning  **$X$  to the empty  $Y$**   
 $= s(x_1, -) + s(x_2, -) + s(x_3, -) + \dots + s(x_j, -)$ ;

# Tabular Computation of Pairwise Alignment

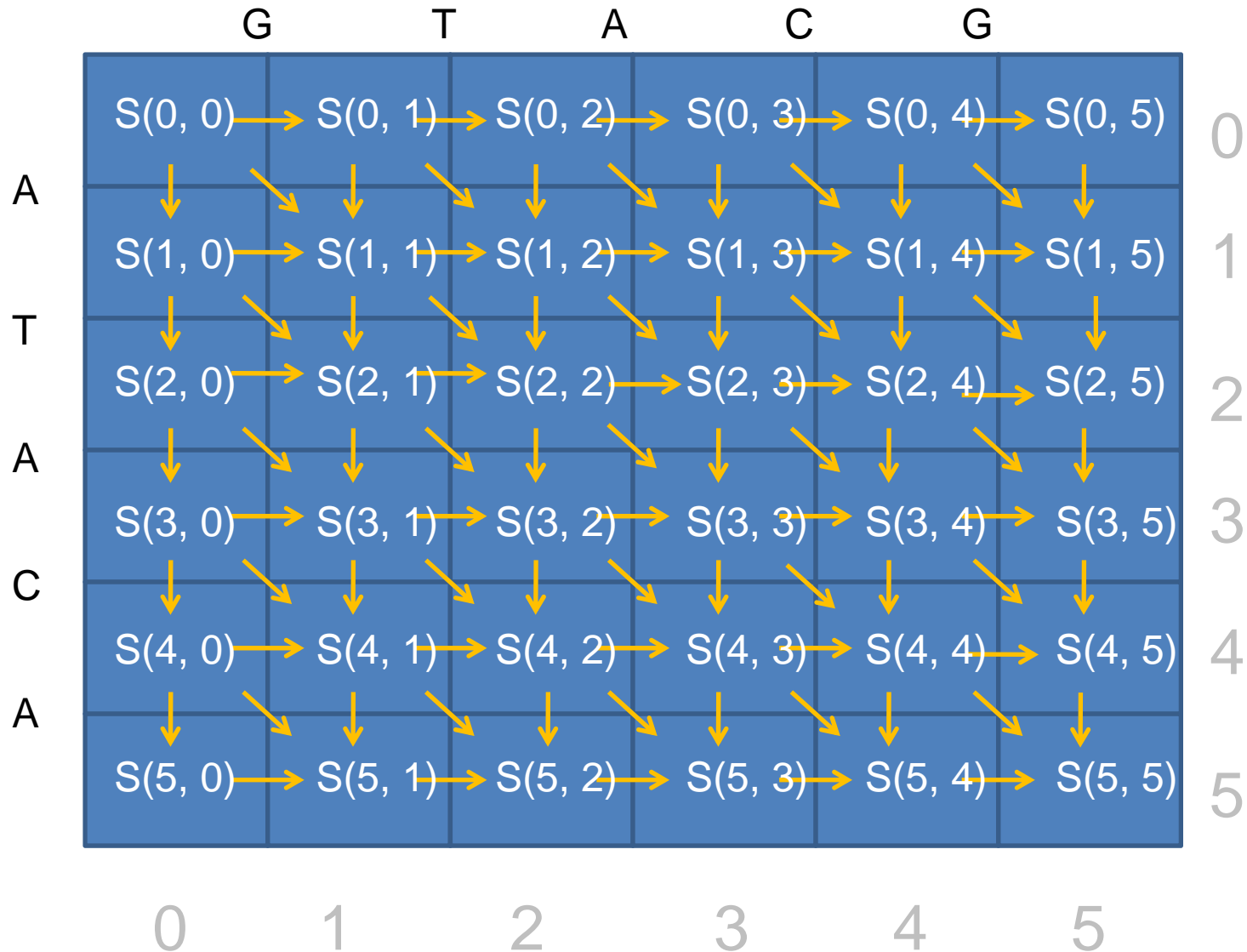
To align an  $j$ -character sequence  $X$  and a  $k$ -character sequence  $Y$ , we form an  $(j+1)$  by  $(k+1)$  matrix  $F$  in which  $S(j', k')$  is the score of optimally aligning  $X[1, \dots, j']$  and  $Y[1, \dots, k']$  and computed by the formula one by one from left to right, top to bottom.

	A	G	G	C	G	T
A						
G						
T						
G						

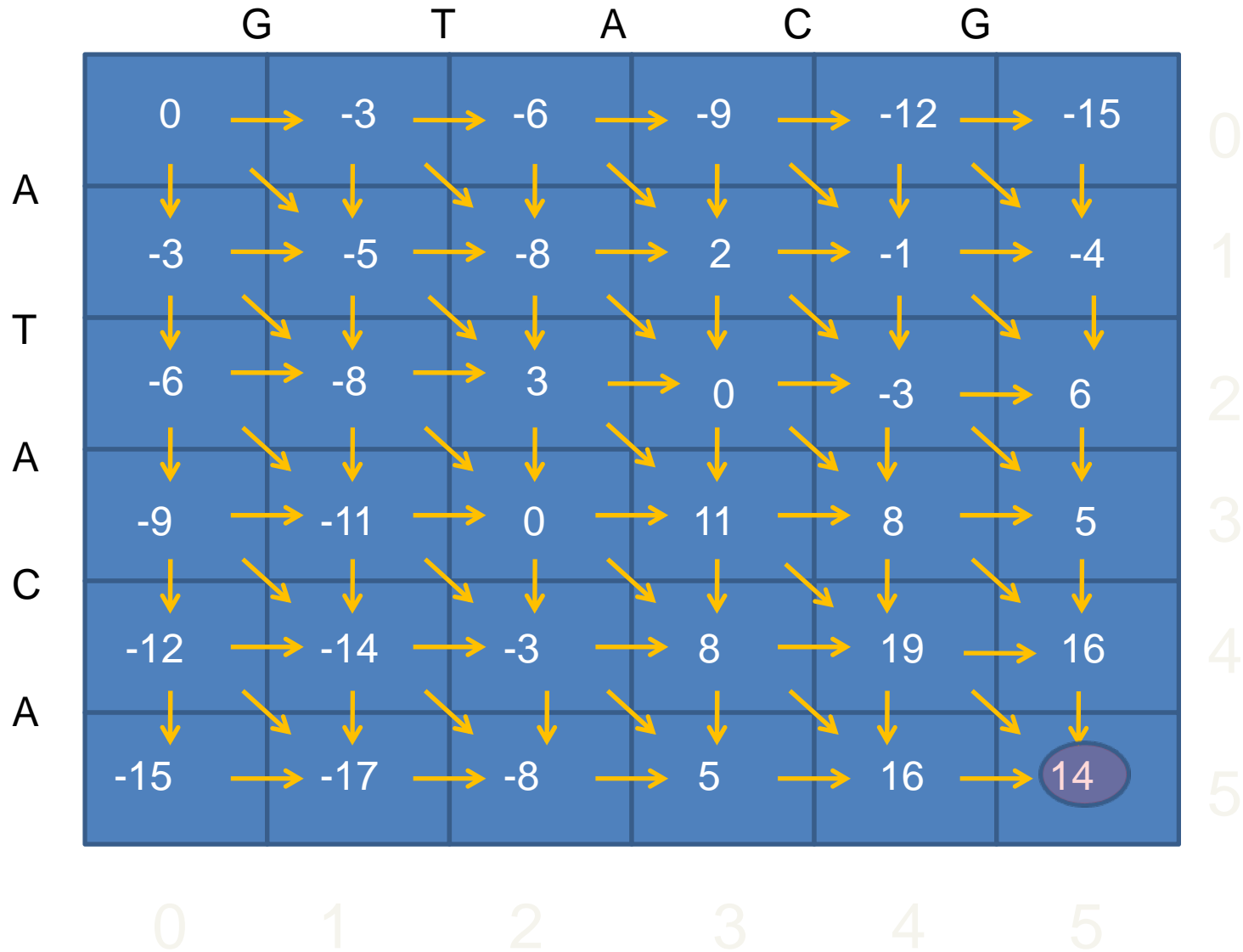
The score of the best align. of AGT and AG

The score of the best align. of X and Y

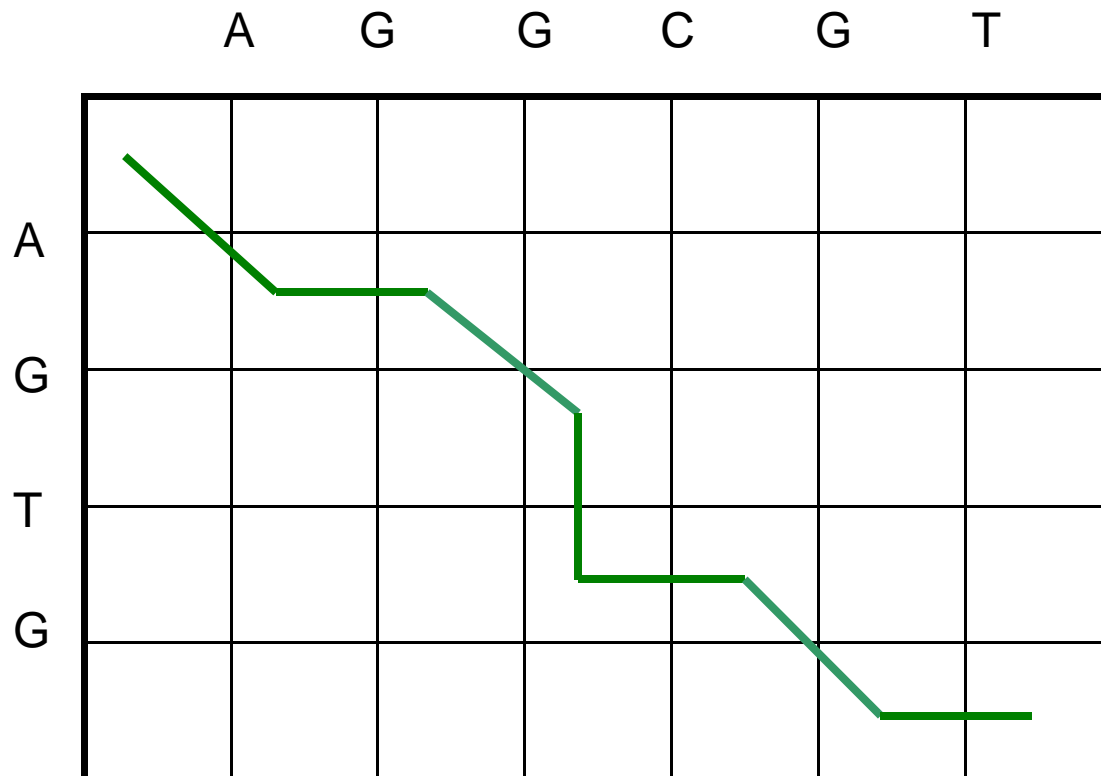
# Alignment Example with Scoring Scheme



# Alignment Example with Scoring Scheme: Match: 8, mismatch -5, indel column: -3

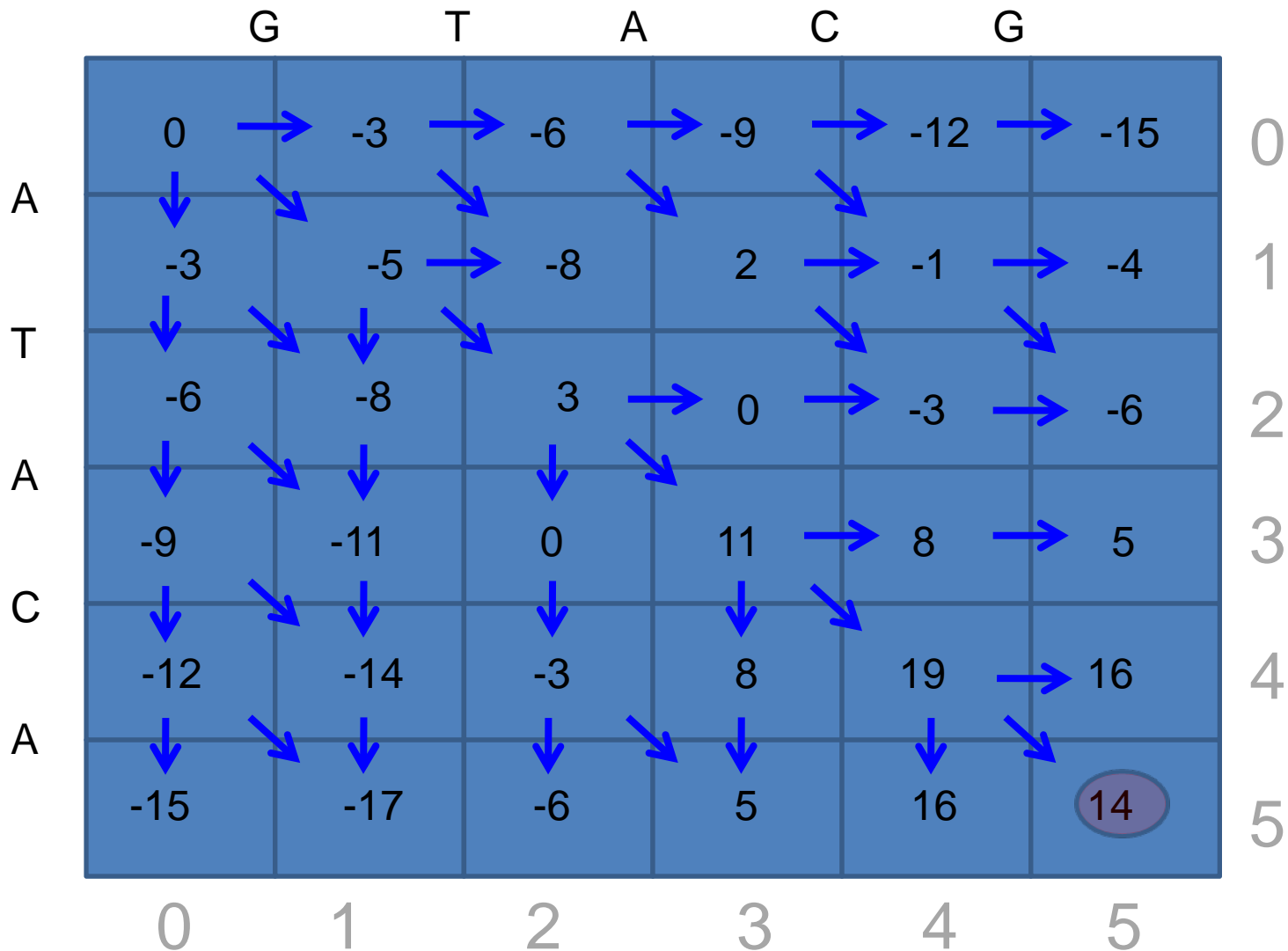


An optimal alignment can be found using information about how  $S(j, k)$  are calculated.

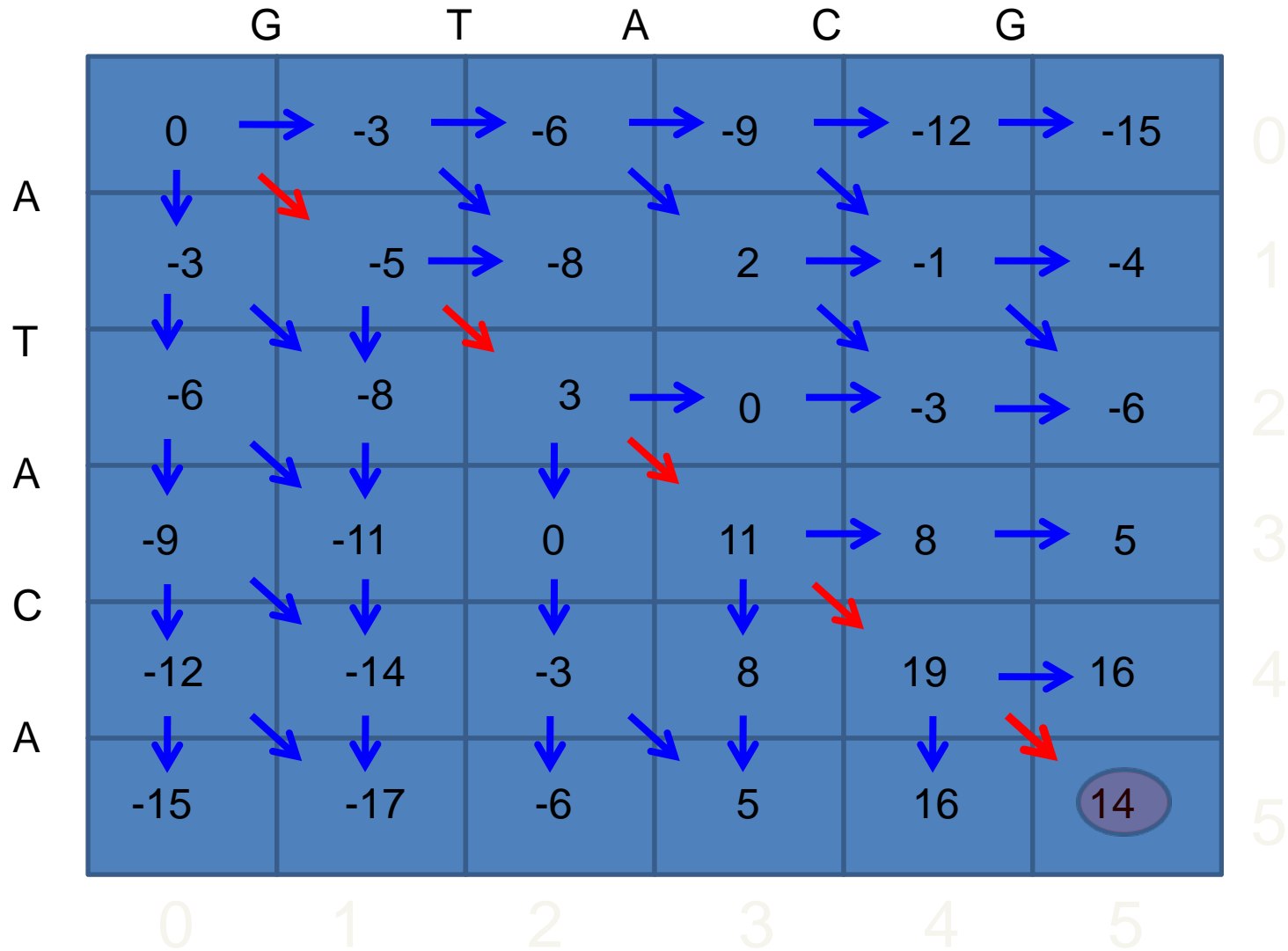


$a \rightarrow b$  indicates that  $b$  can be calculated from  $a$ .

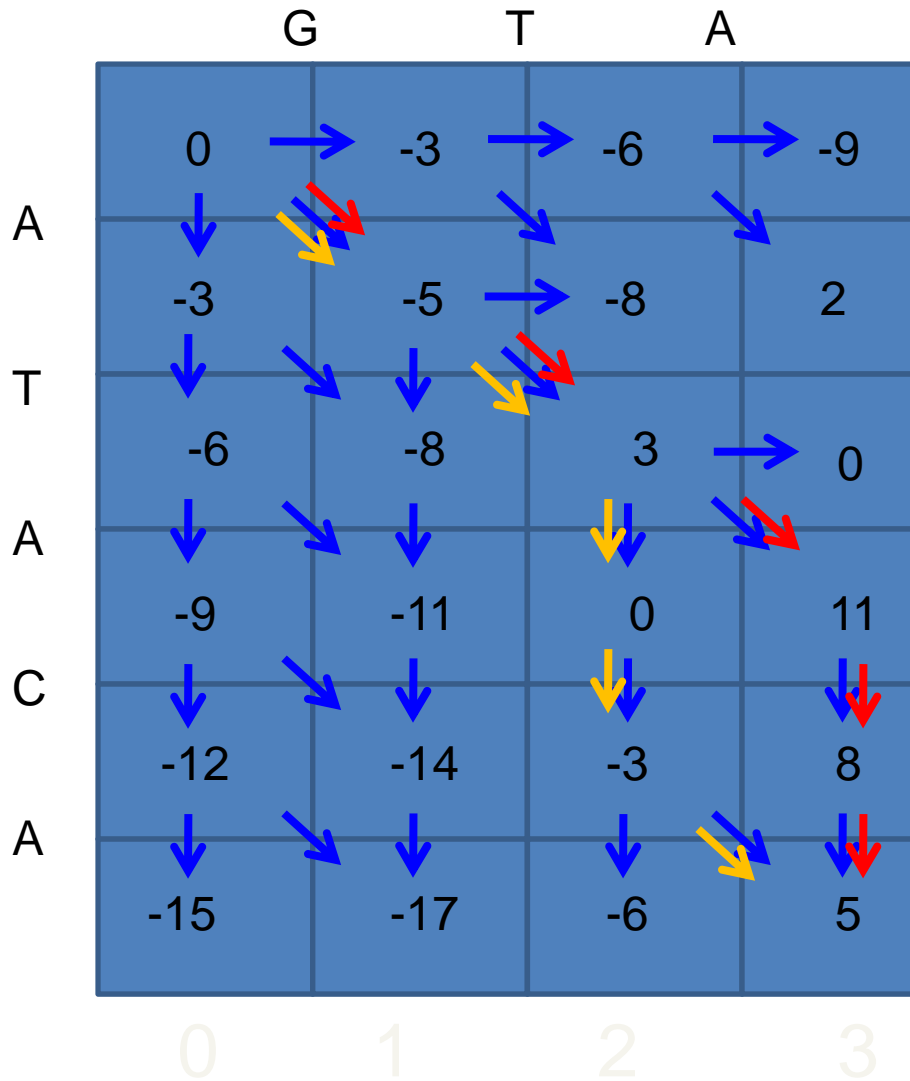
Any non-decreasing path from **the first cell** to **the last cell** gives an optimal alignment.



Backtracking from the right-most cell at the bottom to the left-most cell on the top obtain the optimal path. **It is not unique unlike this example in general.**



An example where there are more than one optimal alignment



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

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

# Summary

---

To align a  $j$ -char sequence and a  $k$ -char sequence, we use a  $(j+1) \times (k+1)$  table in which each entry is an alignment score. We compute the entry at  $(j', k')$  from three entries at  $(j'-1, k')$ ,  $(j', k'-1)$ ,  $(j'-1, k'-1)$  in 5 operations using a recurrence formula. Hence, **the alignment score can be computed in  $5(j+1)(k+1)$  operations.**

$$S(j', k') = \max \begin{cases} S(j'-1, k') + s(x_{j'}, -); \\ S(j', k'-1) + s(-, y_{k'}); \\ S(j'-1, k'-1) + s(x_{j'}, y_{k'}). \end{cases}$$

# Summary (con't)

To output an optimal alignment, we trackback the score computation from the the last cell to the first cell. For this purpose, we need to **know** which of entries  $(j'-1, k')$ ,  $(j', k'-1)$  and  $(j'-1, k'-1)$  gives the entry at  $(j', k')$  using three extra pointers. Overall, this also takes about  $3(j+1)(k+1)$  operations.

**Theorem:** Aligning two sequences of length  $j$  and  $k$  respectively takes quadratic time  $O(jk)$ .

Remark: Our algorithm uses  $O(jk)$  cells. But it is possible to use only  $O(j+k)$  cells

# Dynamic Programming Technique

---

Solve an instance of a problem by taking advantage of the computed solutions to the small parts of the instance.

The term was first used by Bellman in 1940s

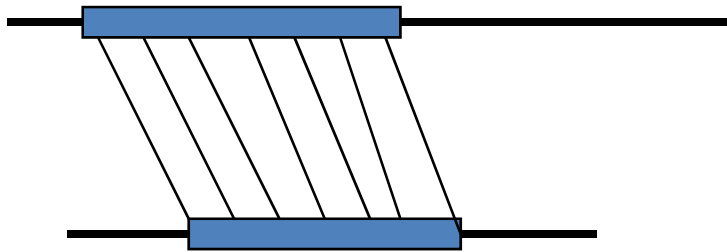


Programming is a synonym for optimization.

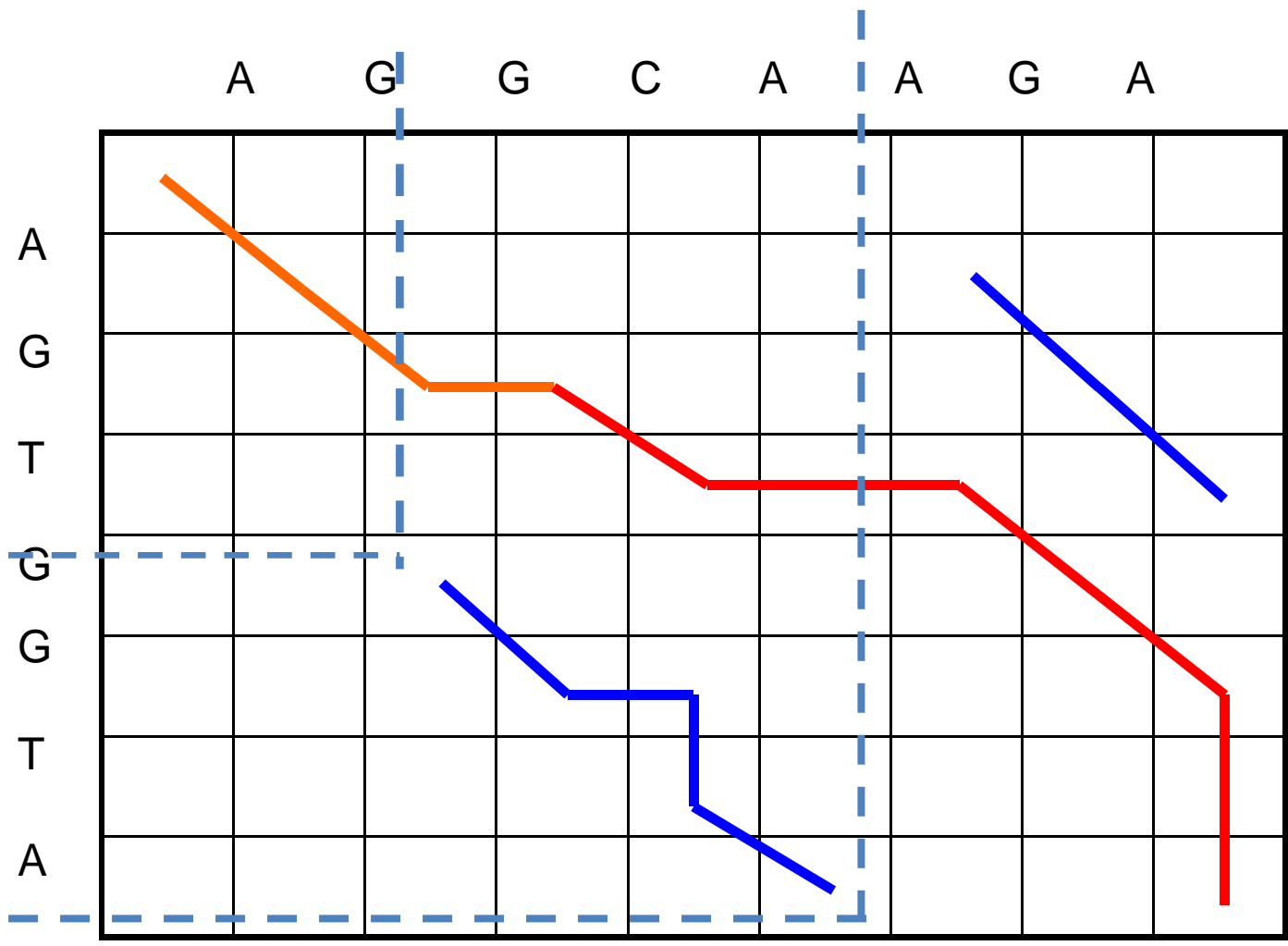
**Richard Ernest Bellman (1920–1984)**

### 3. Locally Aligning Sequences

---



Given two sequences  $X$  and  $Y$ , together with a scoring scheme, find the best alignment of a segment of  $X$  and a segment of  $Y$ .



# 3. Locally Aligning Sequences

Input: Two sequences of length  $m$  and  $n$ , and a scoring scheme  $s( , )$ .

**Step 1:**  $F(0, k')=0$ ;  $F(j', 0)=0$ ; and for  $j' > 0$  and  $k' > 0$ , compute  $F(j', k')$  by

$$F(j', k') = \max \begin{cases} 0 \\ F(j', k'-1) + s(x_{j'}, -) \\ F(j'-1, k') + s(-, y_{k'}) \\ F(j'-1, k'-1) + s(x_{j'}, y_{k'}) \end{cases}$$

**Step 2:** Identify the largest entry  $F(m, n)$  in the table and backtrack from the cell  $(m, n)$  to a cell  $(m', n')$  such that  $F(m', n')=0$  to obtain an optimal local alignment.

Match: 8

Mismatch: -5

Gap symbol: -3

# local alignment

	C	G	G	A	T	C	A	T	
C	0	0	0	0	0	0	0	0	
T	0	8	5	2	0	0	8	5	2
T	0	5	3	0	0	8	5	3	13
A	0	2	0	0	0	8	5	2	11
A	0	0	0	0	8	5	3	?	
C	0				0				
A	0								

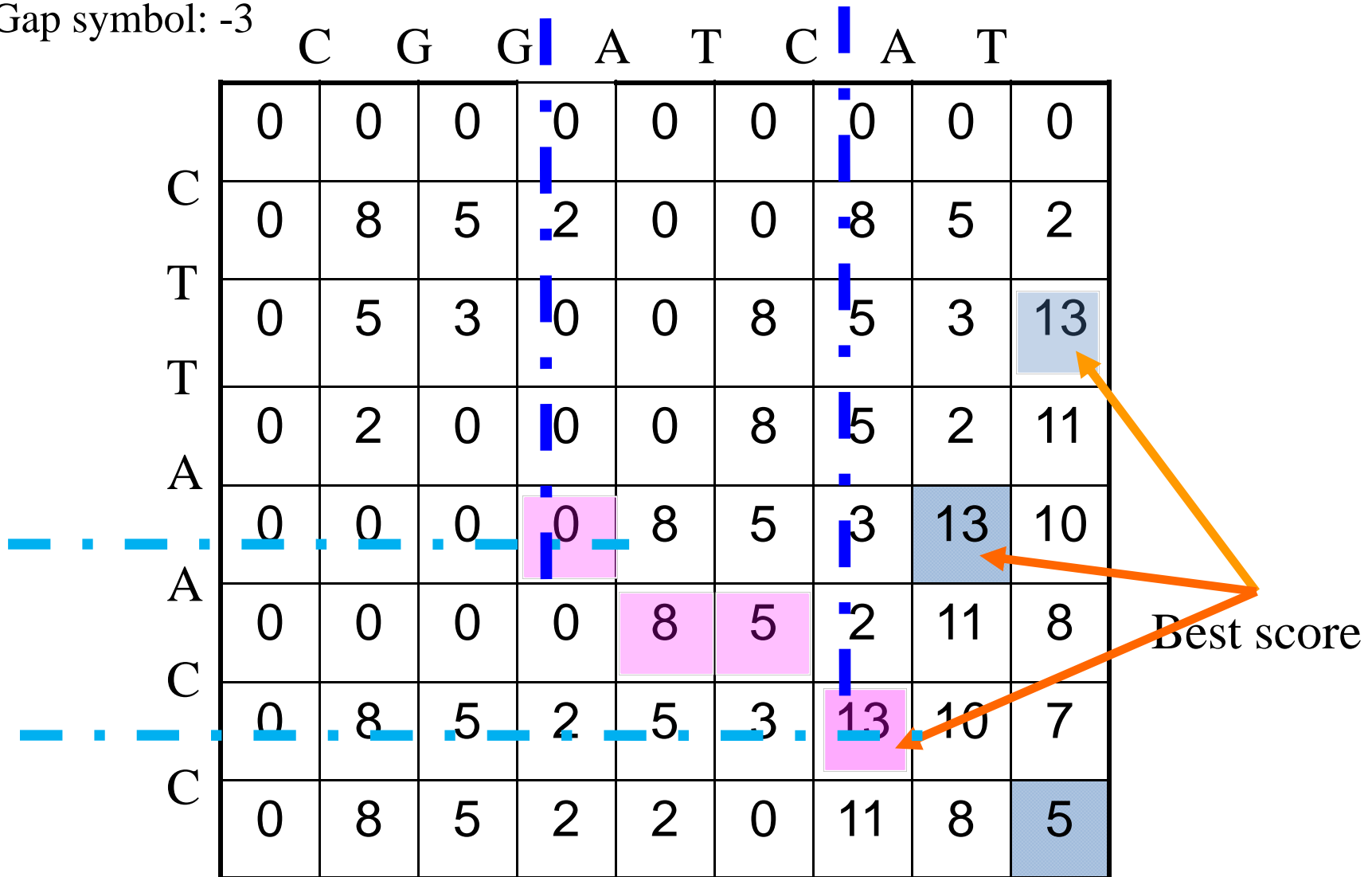
$F(j, k) = \max \left\{ \begin{array}{l} F(j, k-1) + w(x_i, -) \\ F(j-1, k) + w(-, y_j) \\ F(j-1, k-1) + w(x_i, y_j) \end{array} \right.$

Match: 8

Mismatch: -5

Gap symbol: -3

# Example of a local alignment



# 4. Dot Plot Matrix

A dot matrix allows the users to visualize quickly the similar regions between two sequences.

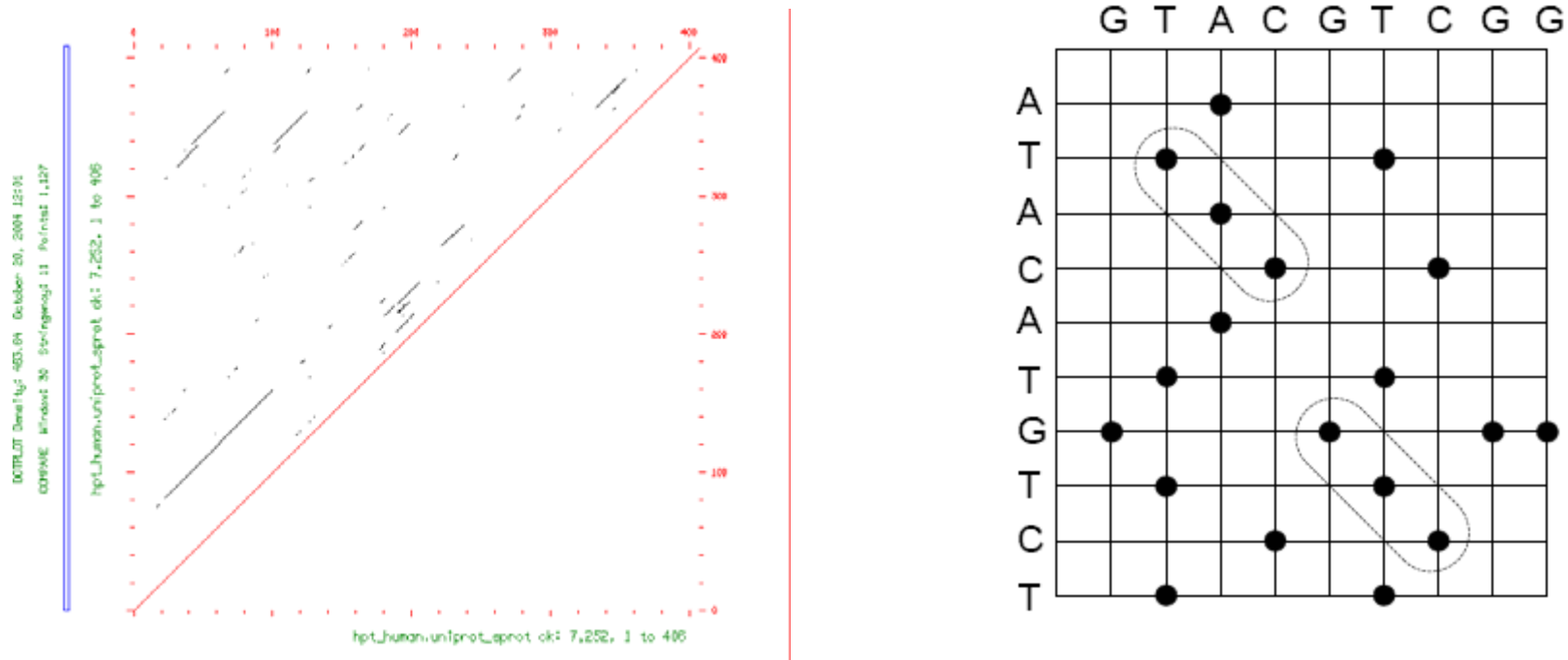


Fig. 3.1 A dot matrix of the two sequences ATACATGTCT and GTACGTCGG.

As a method, it is efficient and intuitive.

# Notes on Lecture

Time: 1:20 minutes.

Transition from Number of alignments to alignment confused students.

Need to work the examples in details

Insert the constant gap model in the last topics.