Sequence Alignment and Dynamic Programming

Lecture 3

September 10, 2019

A Genome Sequence

```
GCGTCTGACGGCGCACCGTTCGCGCTGCCGGCACCCCGGGCTCCATAATGAAAATCATGT
TCAGTAAGCTACACTCTGCATATCGGGCTACCAACGAAATGGAGTATCGGTCATGATCTT
GCCAGCCGTGCCTAAAAGCTTGGCCGCAGGGCCGAGTATAATTGGTCGCGGTCGCCTCGA
AGTTAGCTTATGCAATGCAGGAGGTGGGGCAAAGTTCAGGCGGATCGGCCGATGGCGGGC
GTAGGTGAAGGAGACAGCGGAGGCGTGGAGCGTGATGACATTGGCATGGTGGCCGCTTCC
CCCGTCGCGTCTCGGGTAAATGGCAAGGTAGACGCTGACGTCGTCGGTCGATTTGCCACC
TGCTGCCGTGCCCTGGGCATCGCGGTTTACCAGCGTAAACGTCCGCCGGACCTGGCTGCC
GCCCGGTCTGGTTTCGCCGCGCTGACCCGCGTCGCCCATGACCAGTGCGACGCCTGGACC
GGGCTGGCCGCTGCCGGCGACCAGTCCATCGGGGTGCTGGAAGCCGCCTCGCGCACGGCG
ACCACGGCTGGTGTGTTGCAGCGGCAGGTGGAACTGGCCGATAACGCCTTGGGCTTCCTG
TACGACACCGGGCTGTACCTGCGTTTTCGTGCCACCGGACCTGACGATTTCCACCTCGCG
TATGCCGCTGCGTTGGCTTCGACGGCGGGCCGGAGGAGTTTGCCAAGGCCAATCACGTG
```

GGAGGTGGGGCAAAGTTCAGG

A fragment of a gene

A Genome Sequence

```
GCGTCTGACGGCGCACCGTTCGCGCTGCCGGCACCCCGGGCTCCATAATGAAAATCATGT
TCAGTAAGCTACACTCTGCATATCGGGCTACCAACGAAATGGAGTATCGGTCATGATCTT
GCCAGCCGTGCCTAAAAGCTTGGCCGCAGGGCCGAGTATAATTGGTCGCGGTCGCCTCGA
AGTTAGCTTATGCAATGCAGGAGGTGGGGCAAAGTTCAGGCGGATCGGCCGATGGCGGGC
GTAGGTGAAGGAGACAGCGGAGGCGTGGAGCGTGATGACATTGGCATGGTGGCCGCTTCC
CCCGTCGCGTCTCGGGTAAATGGCAAGGTAGACGCTGACGTCGTCGGTCGATTTGCCACC
TGCTGCCGTGCCCTGGGCATCGCGGTTTACCAGCGTAAACGTCCGCCGGACCTGGCTGCC
GCCCGGTCTGGTTTCGCCGCGCTGACCCGCGTCGCCCATGACCAGTGCGACGCCTGGACC
GGGCTGGCCGCTGCCGGCGACCAGTCCATCGGGGTGCTGGAAGCCGCCTCGCGCACGGCG
ACCACGGCTGGTGTGTTGCAGCGGCAGGTGGAACTGGCCGATAACGCCTTGGGCTTCCTG
TACGACACCGGGCTGTACCTGCGTTTTCGTGCCACCGGACCTGACGATTTCCACCTCGCG
TATGCCGCTGCGTTGGCTTCGACGGCGGGCCGGAGGAGTTTGCCAAGGCCAATCACGTG
```

GGAGGTGGGGCAAAGTTCAGG

A fragment of a gene

A Genome Sequence

GCGTCTGACGGCGCACCGTTCGCGCTGCCGGCACCCCGGGCTCCATAATGAAAATCATGT
TCAGTAAGCTACACTCTGCATATCGGGCTACCAACGAAATGGAGTATCGGTCATGATCTT
GCCAGCCGTGCCTAAAAGCTTGGCCGCAGGGCCGAGTATAATTGGTCGCGGTCGCCTCGA
AGTTAGCTTATGCAATGCA**GGAGGTGGGGCAAAGTTCAGG**CGGATCGGCCGATGGCGGGC
GTAGGTGAAGGAGACAGCGGAGGCGTGGAGCGTGATGACATTGGCATGGTGGCCGCTTCC
CCCGTCGCGTCTCGGGTAAATGGCAAGGTAGACGCTGACGTCGTCGGTCGATTTGCCACC
TGCTGCCGTGCCCTGGGCATCGCGGTTTACCAGCGTAAACGTCCGCCGGACCTGGCTGCC
GCCCGGTCTGGTTTCGCCGCGCTGACCCGCGTCGCCCATGACCAGTGCGACGCCTGGACC
GGGCTGGCCGCTGCCGGCGACCAGTCCATCGGGGTGCTGGAAGCCGCCTCGCGCACGGCG
ACCACGGCTGGTGTGTTGCAGCGGCAGGTGGAACTGGCCGATAACGCCTTGGGCTTCCTG
TACGACACCGGGCTGTACCTGCGTTTTCGTGCCACCGGACCTGACGATTTCCACCTCGCG
TATGCCGCTGCGTTGGCTTCGACGGCGGGCCGGAGGAGTTTGCCAAGGCCAATCACGTG

GGAGGTGGGGCAAAGTTCAGG

GGAGGAGGGGCAATTTTCAGG

GGAGGAGGG--AATTTTCCGC

Sequence Alignment

- Much of molecular biology concerns analyzing polymers
  - DNA
  - RNA
  - Proteins
- These sequences can be computationally represented by sequences of letters
- Sequence alignment is used in range of biological problems
  - Sequence Annotation
  - Homology detection
  - Function prediction
  - Structure prediction
  - Genome assembly
  - Phylogenetics
  - Genotyping

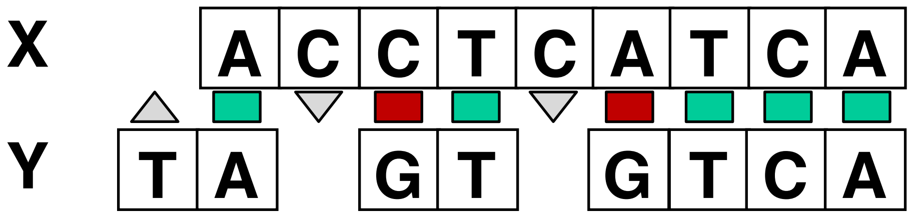
The Sequence Alignment Problem

- Given: Two Sequences

X | A | C | C | T | C | A | T | C | A

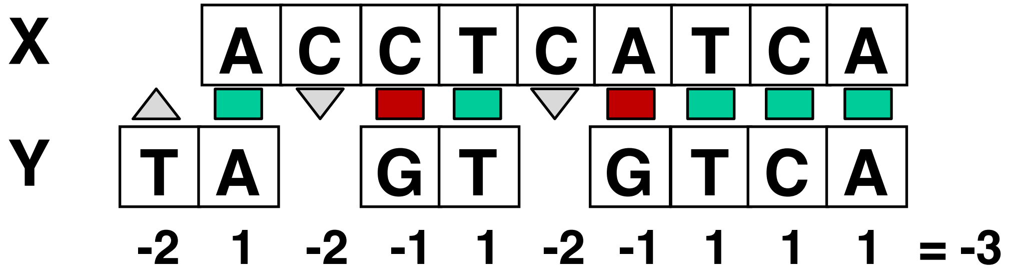
Y | T | A | G | T | G | T | C | A

- Output: An alignment of both



Is this a "good" alignment?

Scoring an Alignment



- Score each position independently

Match	Mismatch	Indel
1	-1	-2

- Additive scoring function: score of sequence is sum of score of each position

The Sequence Alignment Problem

X	A	C	G	T	C	A	T	C	A
Y	T	A	G	T	G	T	C	A	

- Given additive scoring function:
  - Reward of match
  - Cost of mismatch
  - Cost of indel
- Need algorithm for inferring best alignment

What if we just search all possible alignments of two sequences?

Can We Simply Enumerate All Alignments?

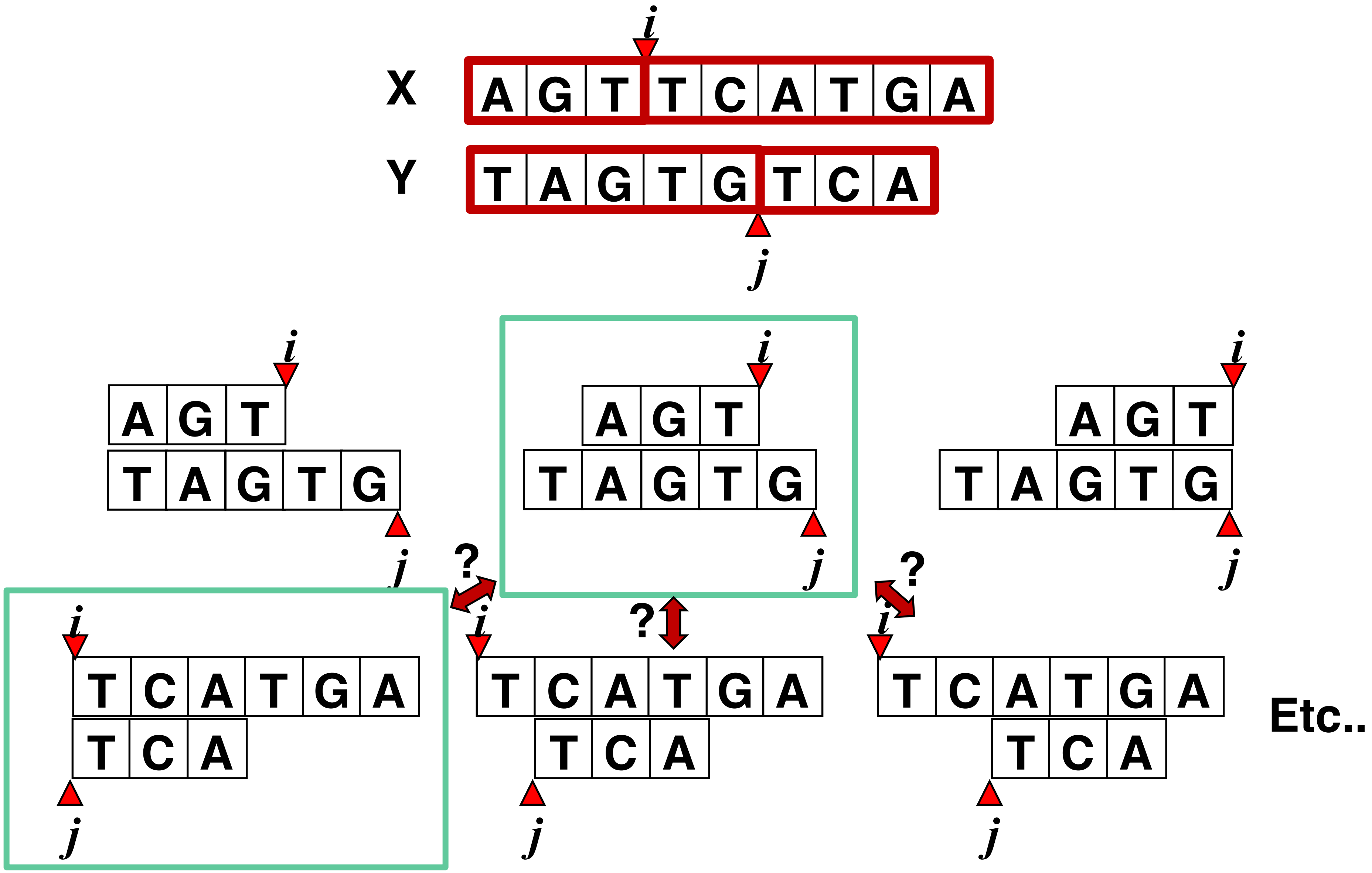
- Ways to align two sequences of length m, n

$$\binom{n+m}{m} = \frac{(m+n)!}{(m!)^2} \approx \frac{2^{m+n}}{\sqrt{\pi \cdot m}}$$

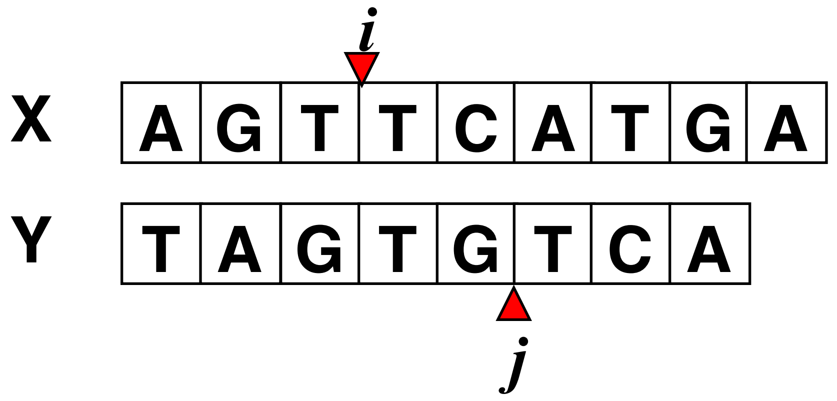
- For two sequences of length n

n	Enumeration
10	184,756
20	1.40E+11
100	9.00E+58

Key Insight: Optimal Substructure



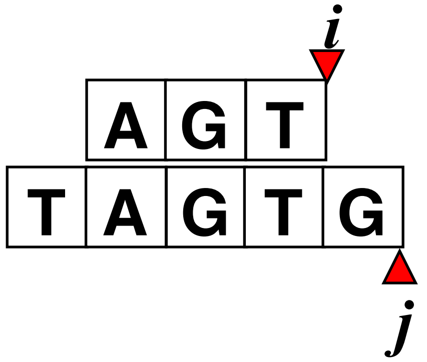
Key Insight: Optimal Substructure

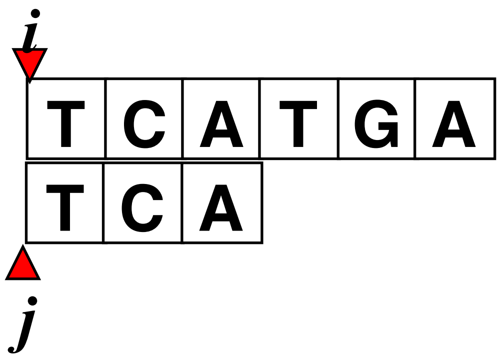


For any pair of indices *(i, j)*, the best alignment of (X,Y) is:

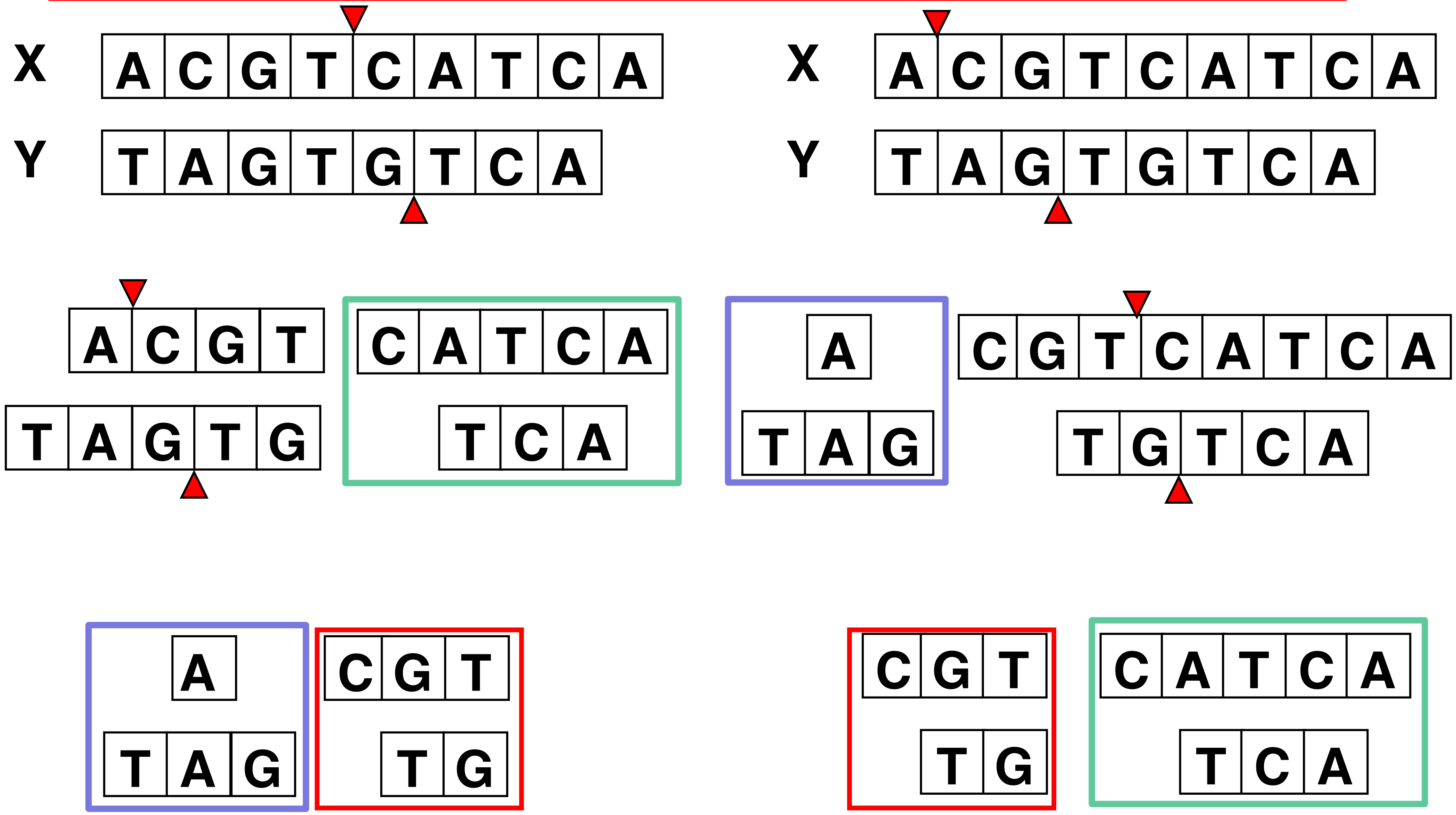
Best alignment of X[1..i] and Y[1..j]

AND Best alignment of X[i..n] and Y[j..m]





Key insight: re-use computation



Identical sub-problems! We can reuse our work!

Using Insights for Sequence Alignment

Global Alignment

Needleman-Wunsch Algorithm

X	A	G	T	
Y	A	A	G	C

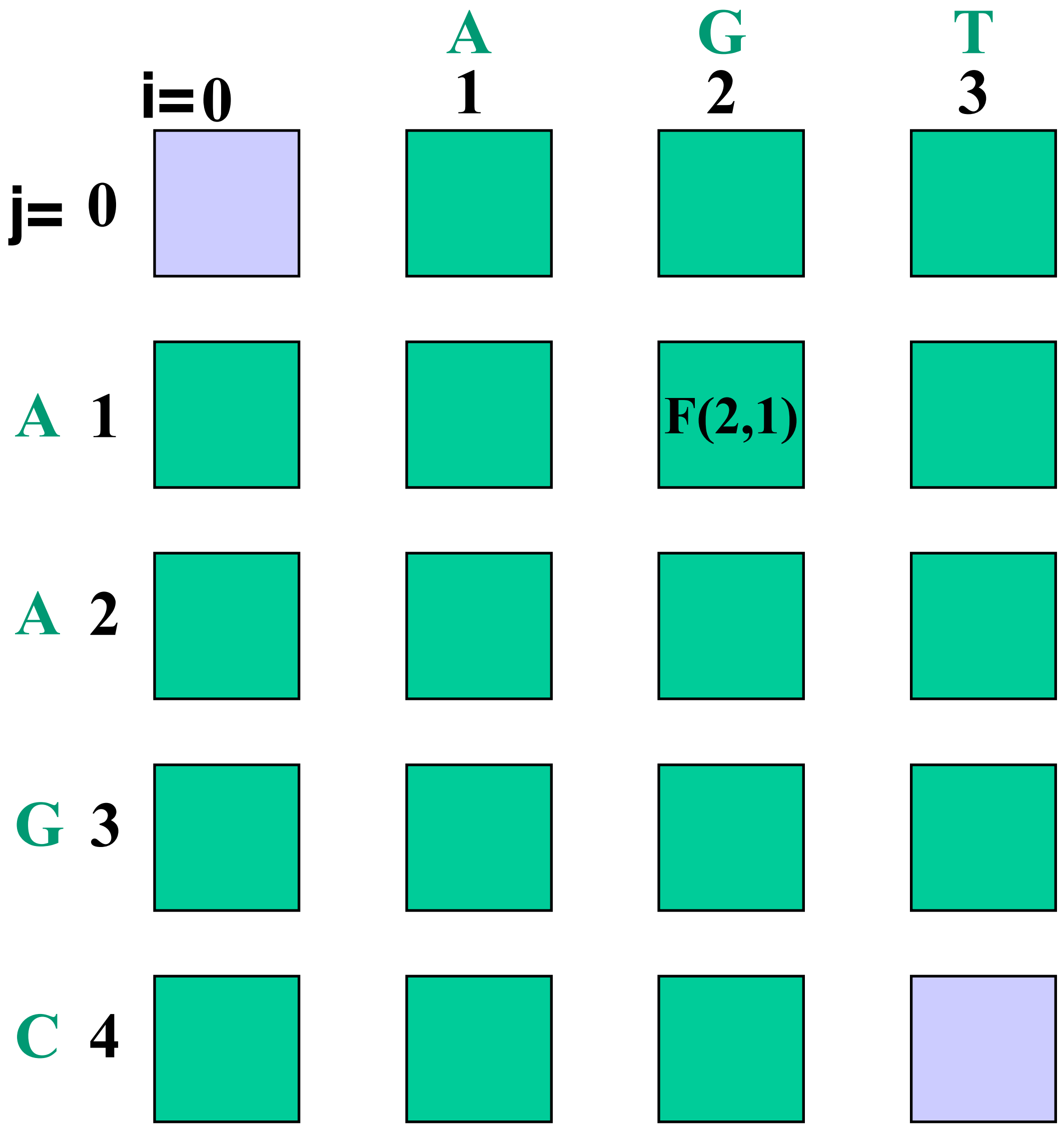
Scoring

Gap = -2

Mismatch = -1 *

Match = 1

Score Matrix – F(i,j)

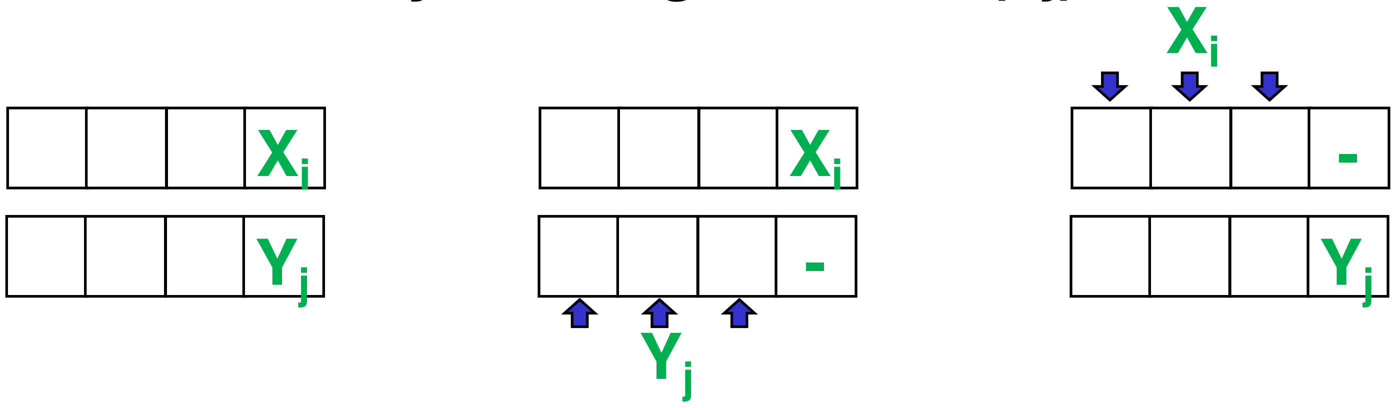


F(i,j)

Score of best alignment between $X_1,\ldots,X_i$ and $Y_1,\ldots,Y_j$

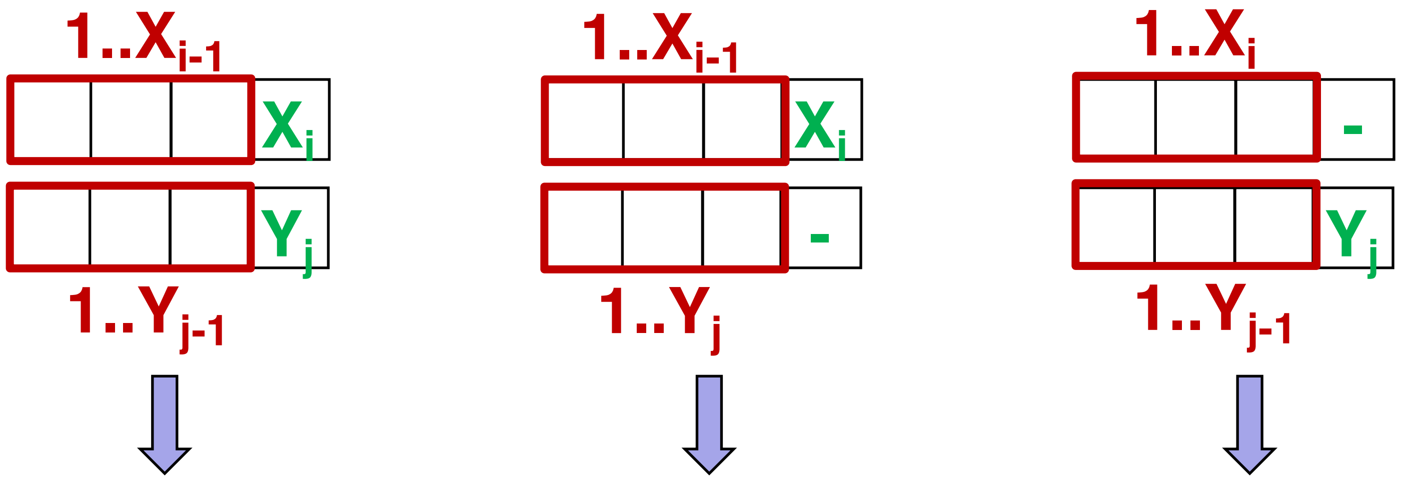
Getting to (i,j)

Three ways an alignment of (i,j) can end



Getting to (i,j)

Three ways an alignment of (i,j) can end



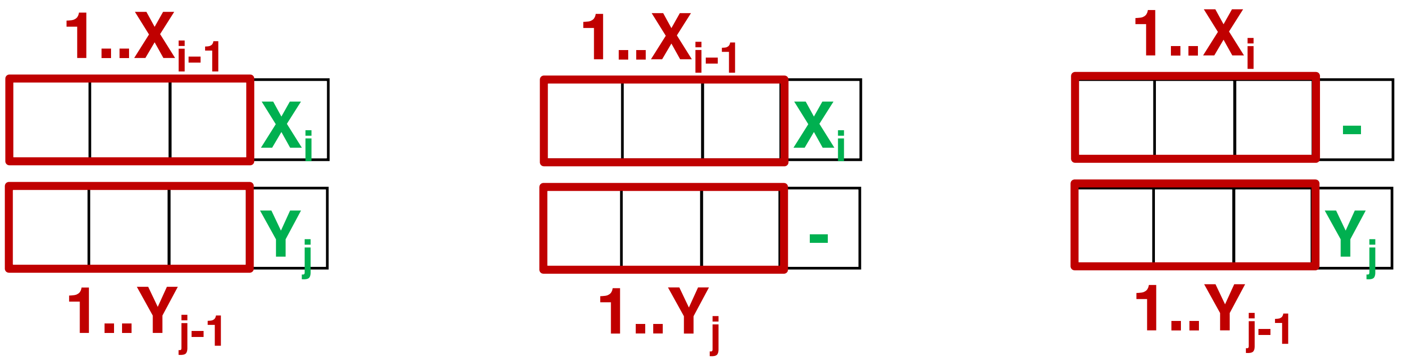
Extend (i-1,j-1) with match or mismatch

Extend (i-1,j) with gap matching X_i

Extend (i,j-1) with gap matching Y_j

Getting to (i,j) – Scoring

What is the score of the best alignment in each scenario?



$f(i,j)=F(i-1,j-1)\underline{+}1$

Best alignment score up to (i-1,j-1)

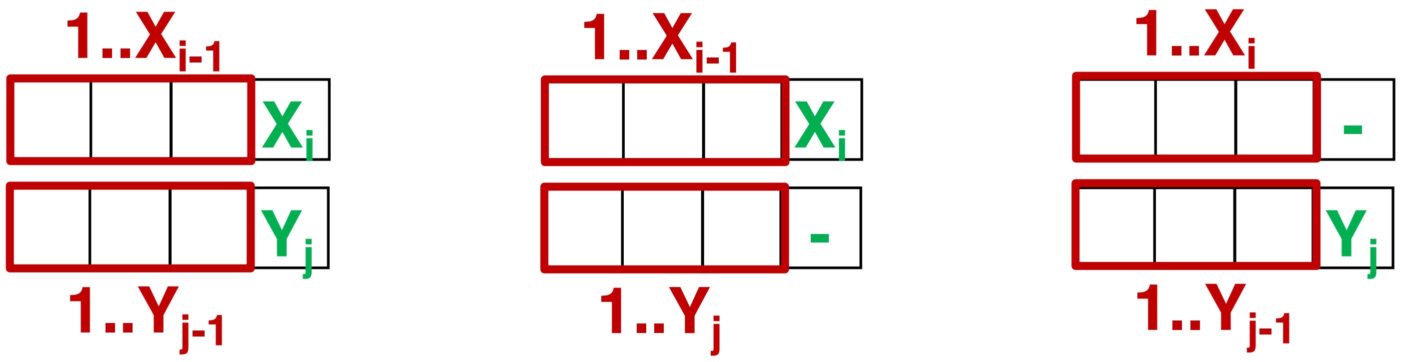
Change in score due to match or mismatch

Optimality Substructure

We do not need the score of *every* alignment (X_{i-1},Y_{j-1}), just the *best*

Getting to (i,j) – Scoring

What is the score of the best alignment in each scenario?



$$f(i,j)=F(i-1,j-1)\pm 1$$

$$f(i,j)=F(i-1,j)-2$$

$$f(i,j)=F(i,j-1)-2$$

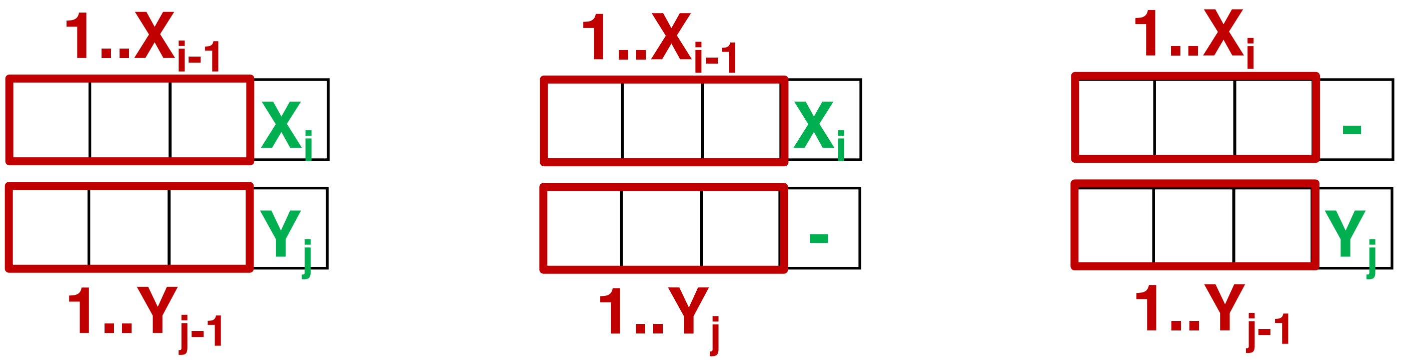
Best alignment score up to (i-1,j-1) — **Change in score due to match or mismatch**

Best alignment score up to (i-1,j) — **Change in score due to gap**

Best alignment score up to (i,j-1) — **Change in score due to gap**

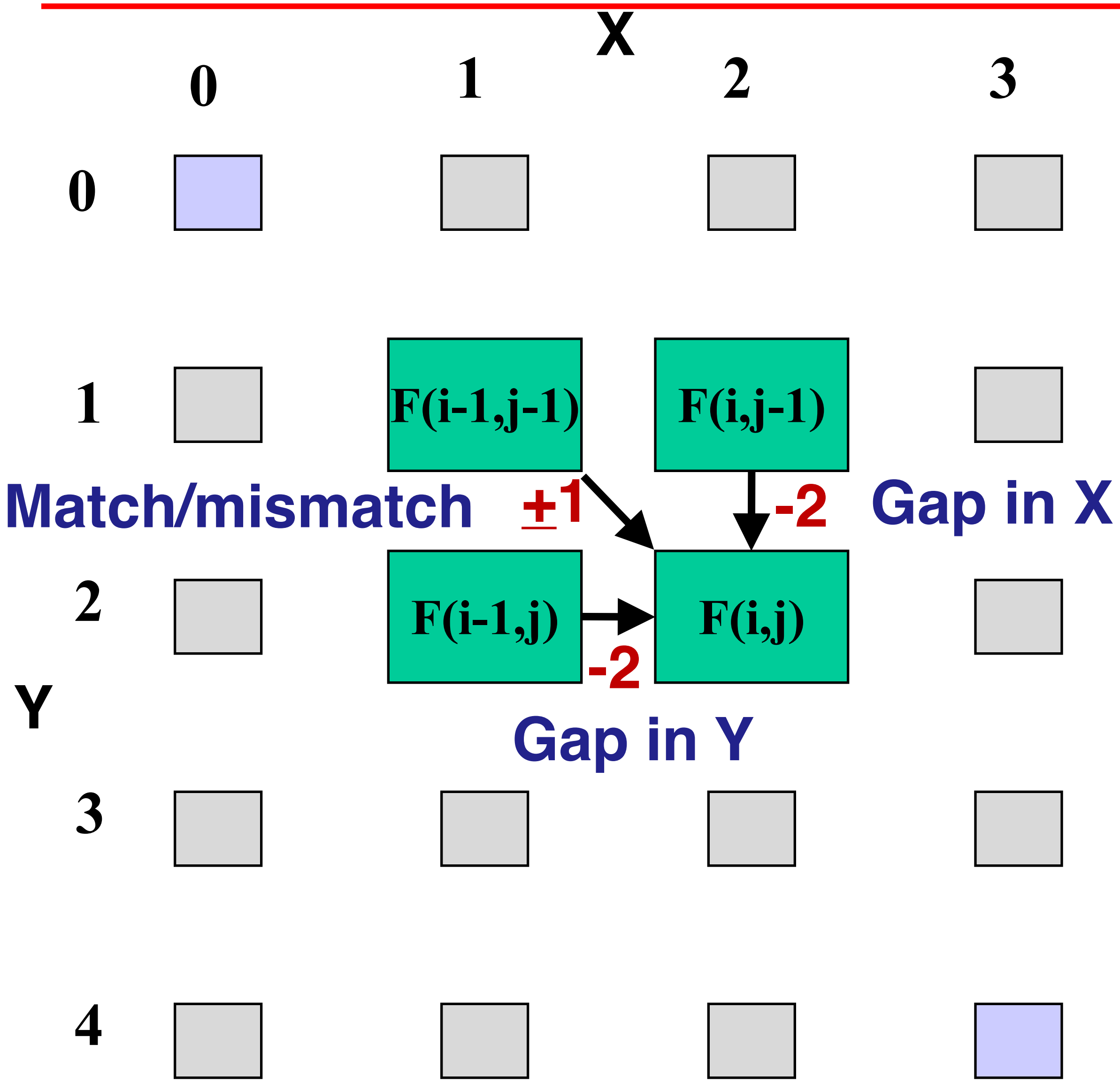
Getting to (i,j) – Scoring F(i,j)

The score of the *best* alignment of (i,j)



$$F(i,j) = \max \begin{cases} F(i-1, j-1) \pm 1 \\ F(i-1, j) - 2 \\ F(i, j-1) - 2 \end{cases}$$

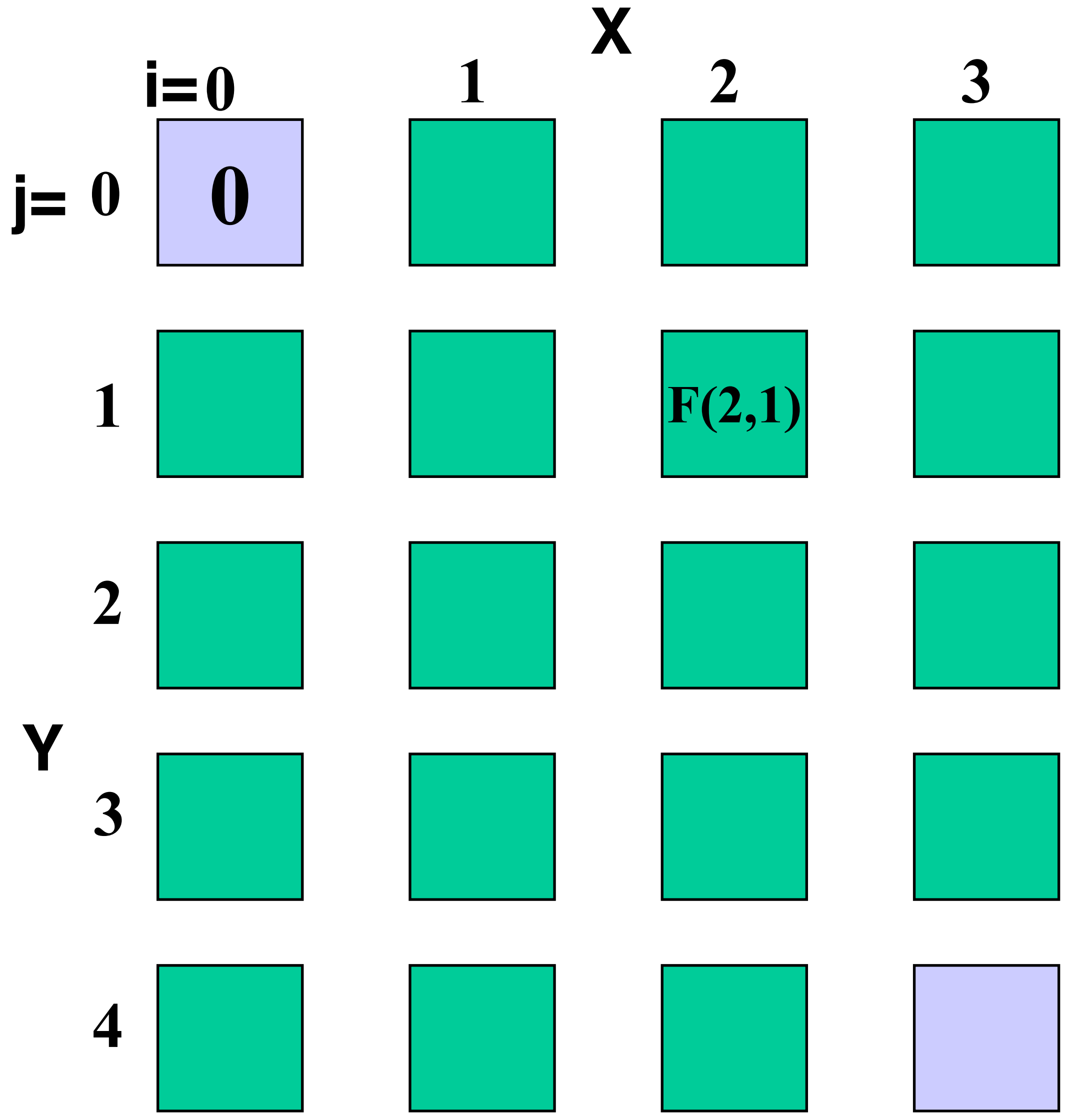
Score Matrix – F(i,j)



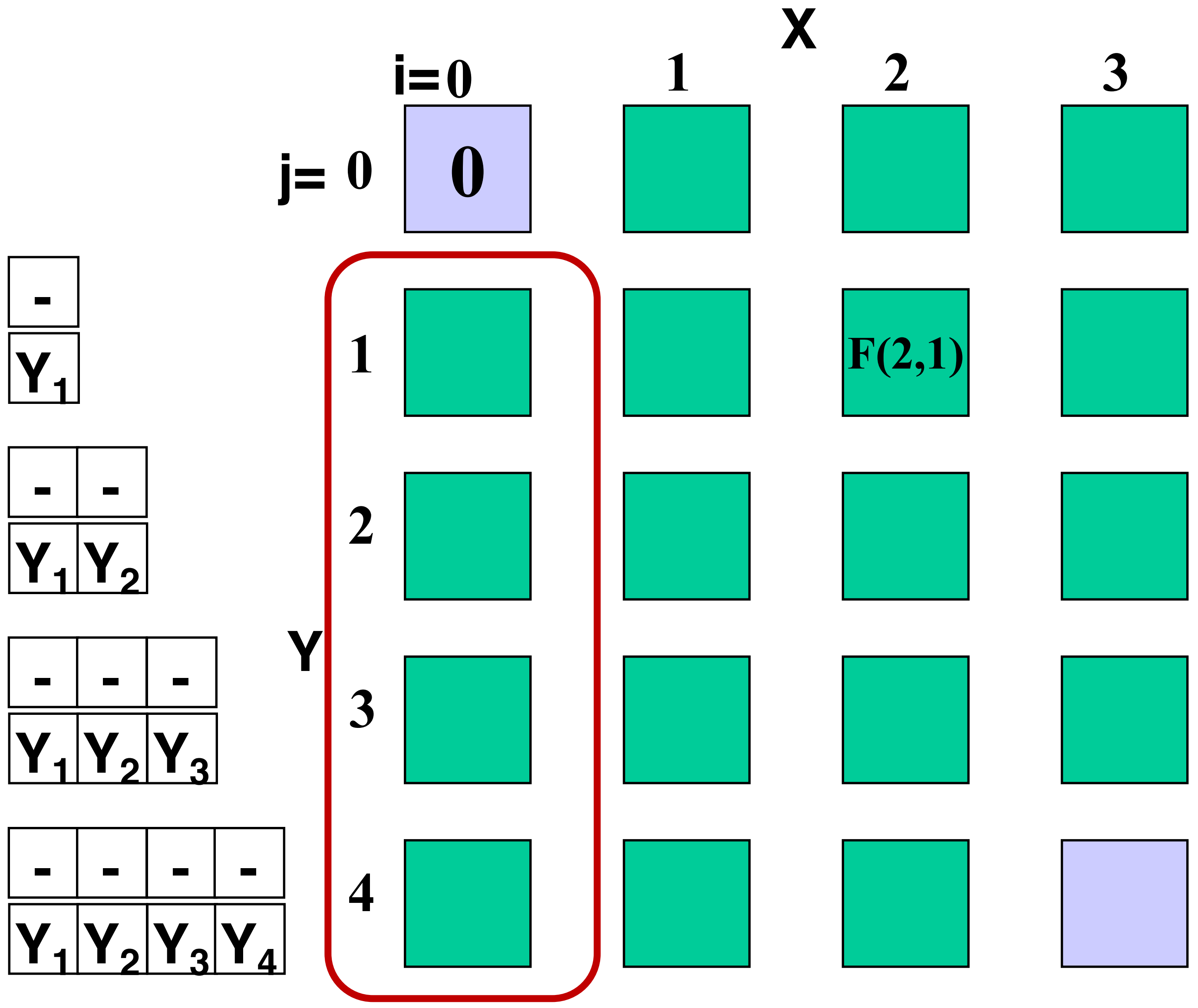
$$F(i,j) = \max \begin{cases} F(i-1,j-1) \pm 1 \\ F(i-1,j) - 2 \\ F(i,j-1) - 2 \end{cases}$$

Score Matrix – Boundaries

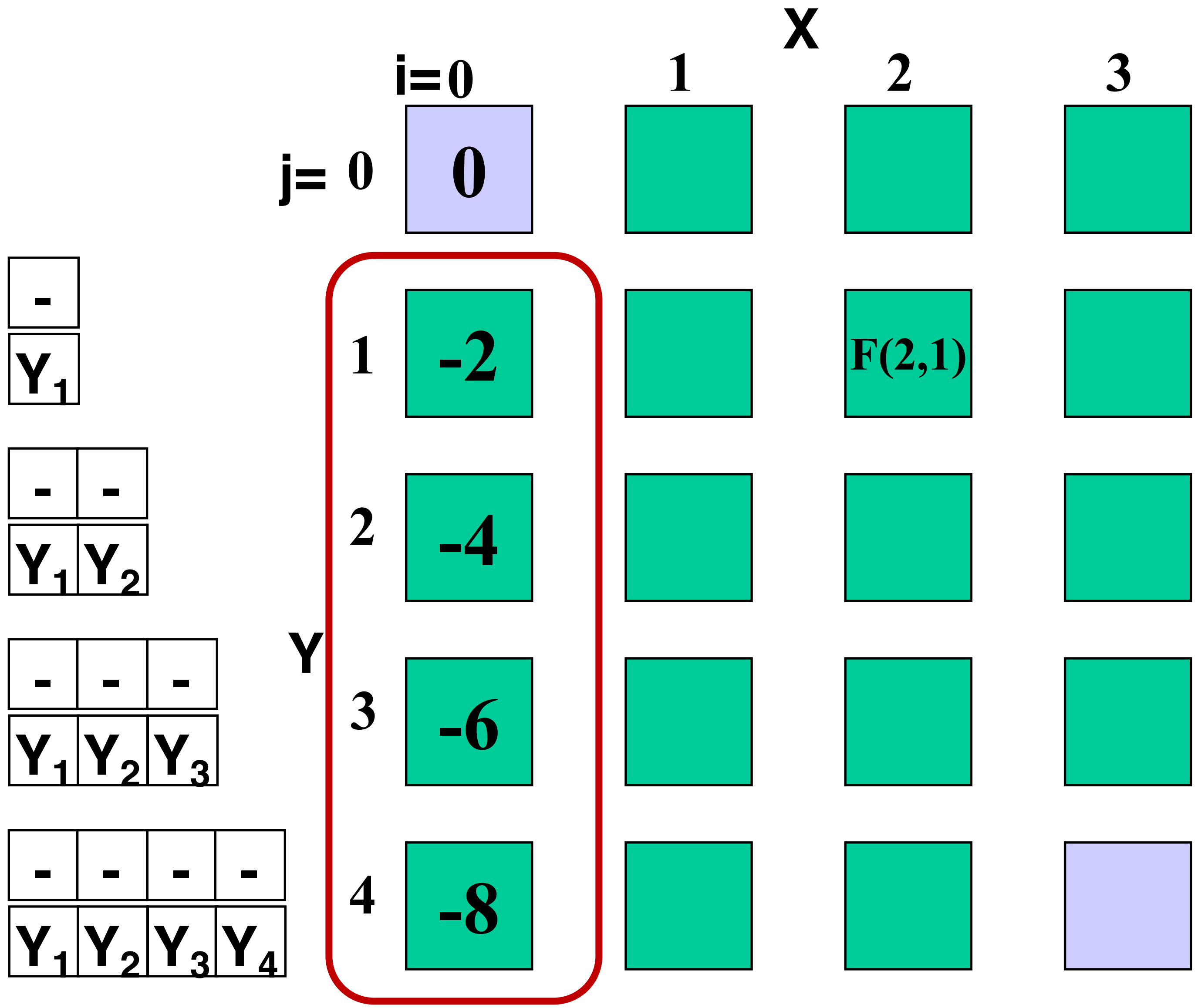
Initialize F(0,0)=0



Score Matrix – Boundaries



Score Matrix – Boundaries



Score Matrix – Boundaries

X

Y	i=0	1	2	3
j= 0	0	-2	-4	-6
1	-2		F(2,1)	
2	-4			
3	-6			
4	-8			

Score Matrix – Boundaries

$$\begin{array}{cc|cccc} & & & & X & \\ & & i=0 & 1 & 2 & 3 \\ \hline & j=0 & 0 & -2 & -4 & -6 \\ & 1 & -2 & & F(2,1) & \\ & 2 & -4 & & & \\ Y & 3 & -6 & & & \\ & 4 & -8 & & & \end{array}$$

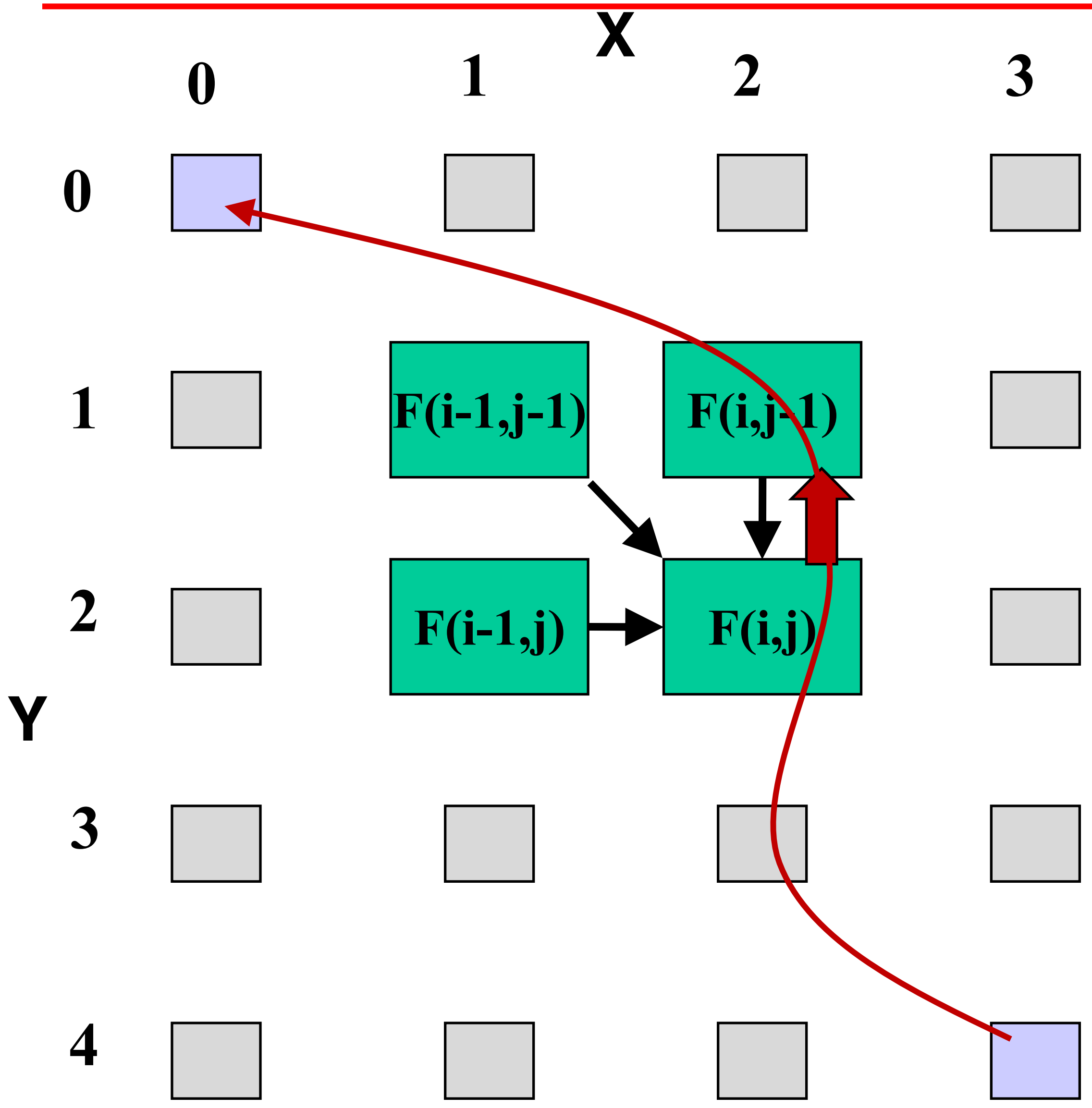
Score Matrix – Boundaries

$$\begin{array}{cccccc}
 & & & X & & \\
 & & i=0 & 1 & 2 & 3 \\
 & j=0 & 0 & -2 & -4 & -6 \\
 & 1 & -2 & & F(2,1) & \\
 & 2 & -4 & & & \\
Y & 3 & -6 & & & \\
 & 4 & -8 & & & \boxed{F(N,M)}
\end{array}$$

F(N,M)
Score **of best global alignment**

But how do we get the actual alignment?

Tracebacks

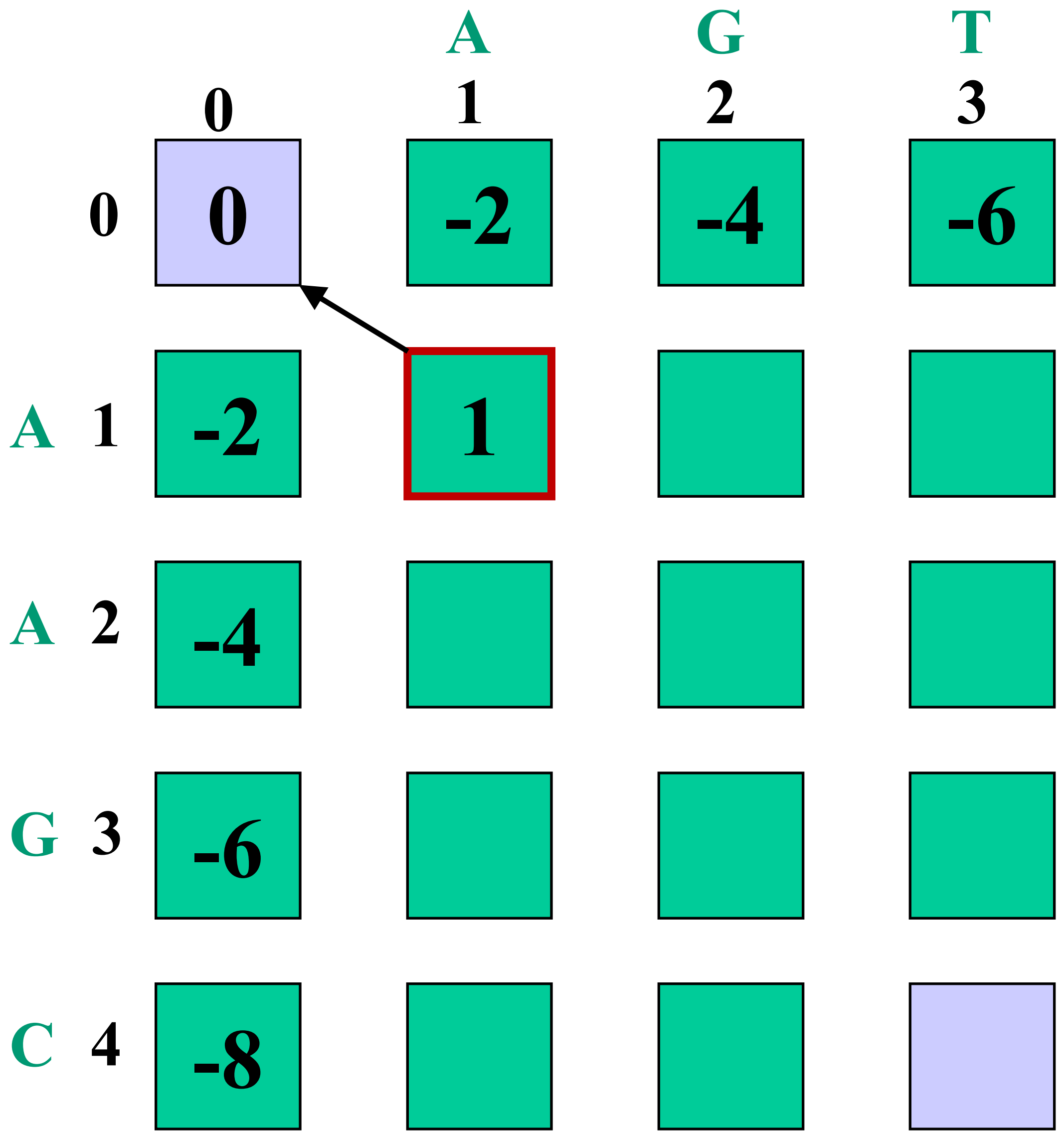


To calculate F(i,j) we “came from” one of

F(i-1,j-1)
F(i,j-1)
F(j-1,i)

Traceback:
Store a pointer from F(i,j) to the cell used

Filling in an Example



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

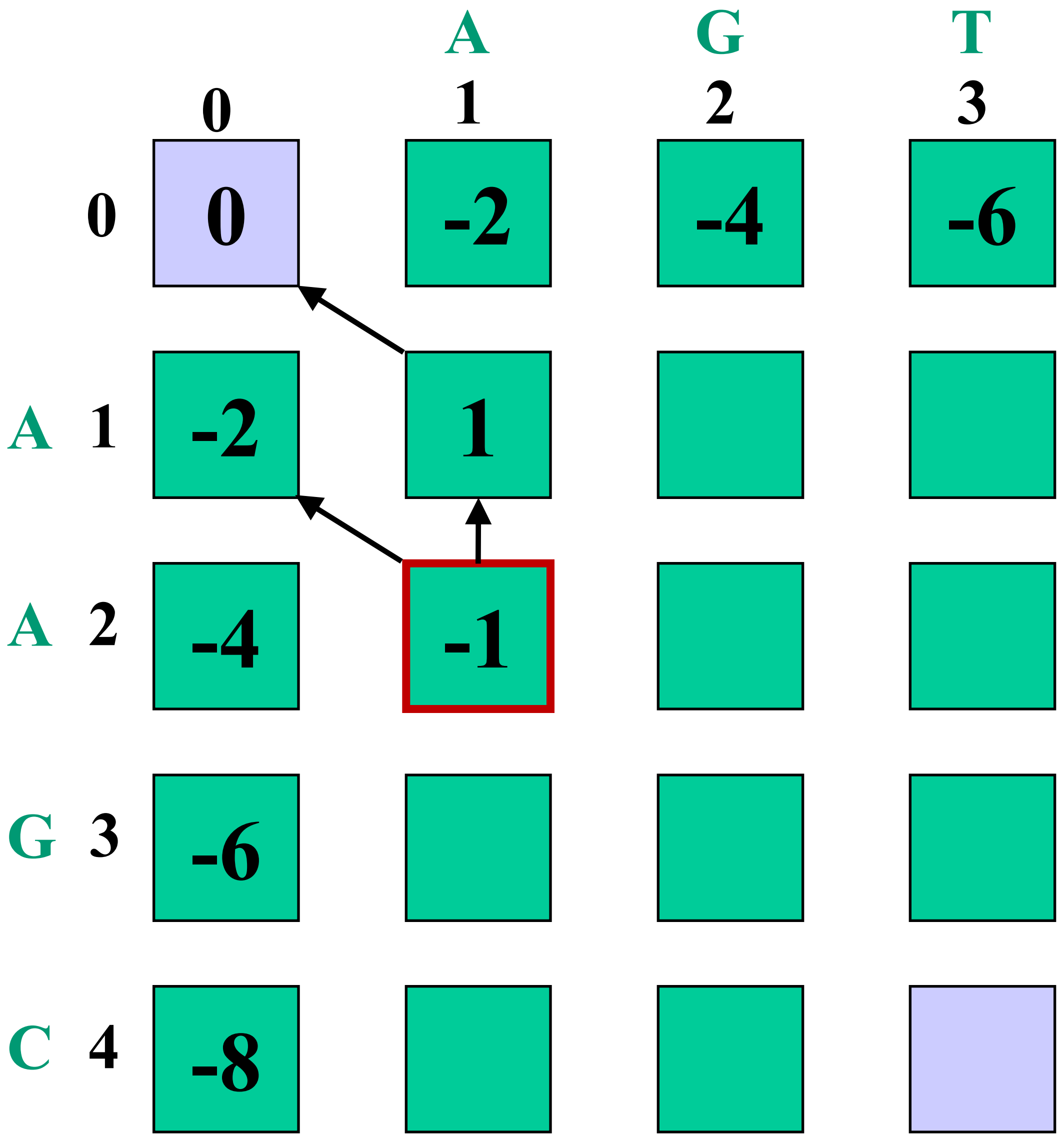
Scoring

Gap = -2

Mismatch = -1

Match = 1

Filling in an Example



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

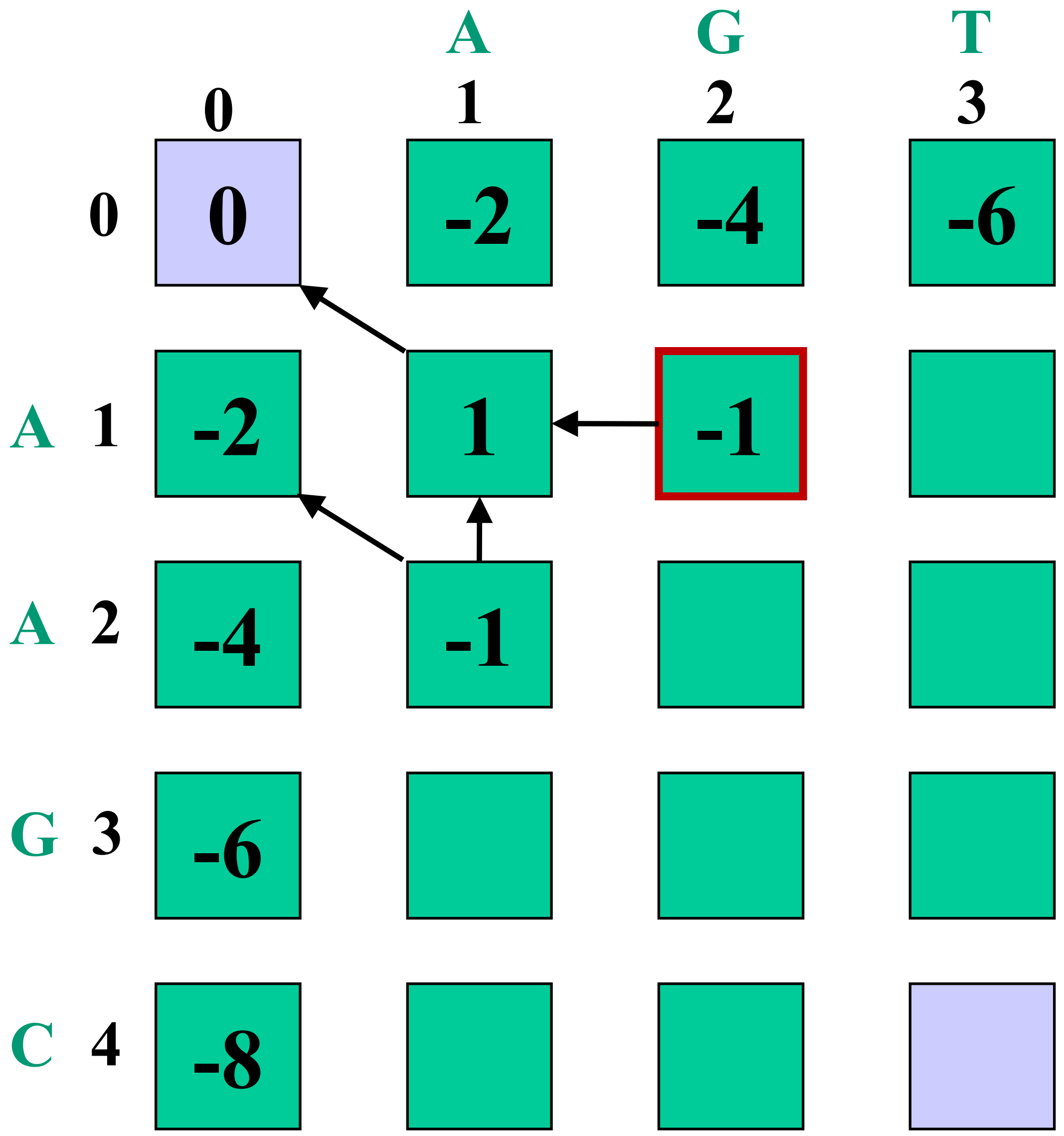
Scoring

Gap = -2

Mismatch = -1

Match = 1

Filling in an Example



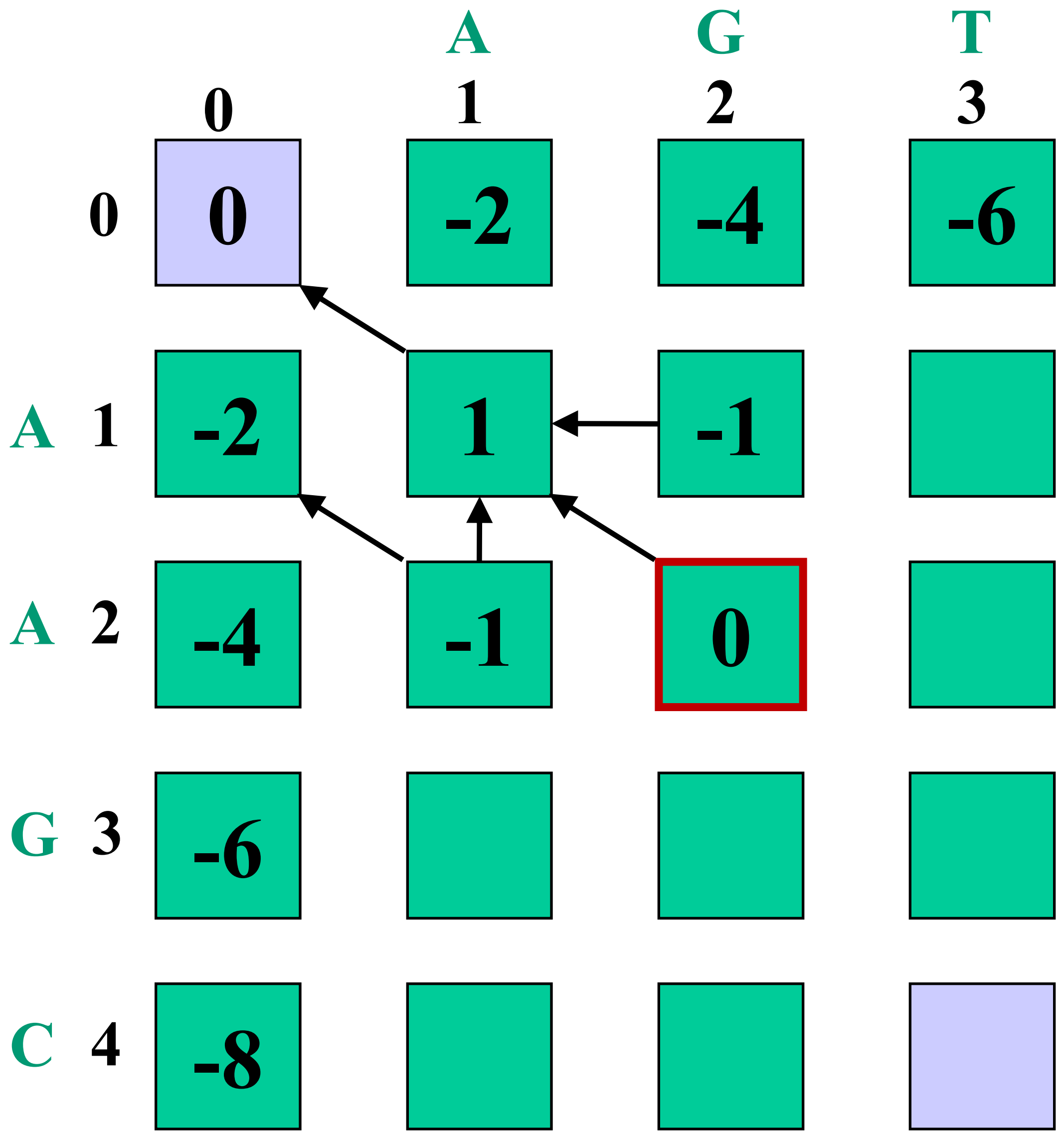
Scoring

Gap = -2

Mismatch = -1

Match = 1

Filling in an Example



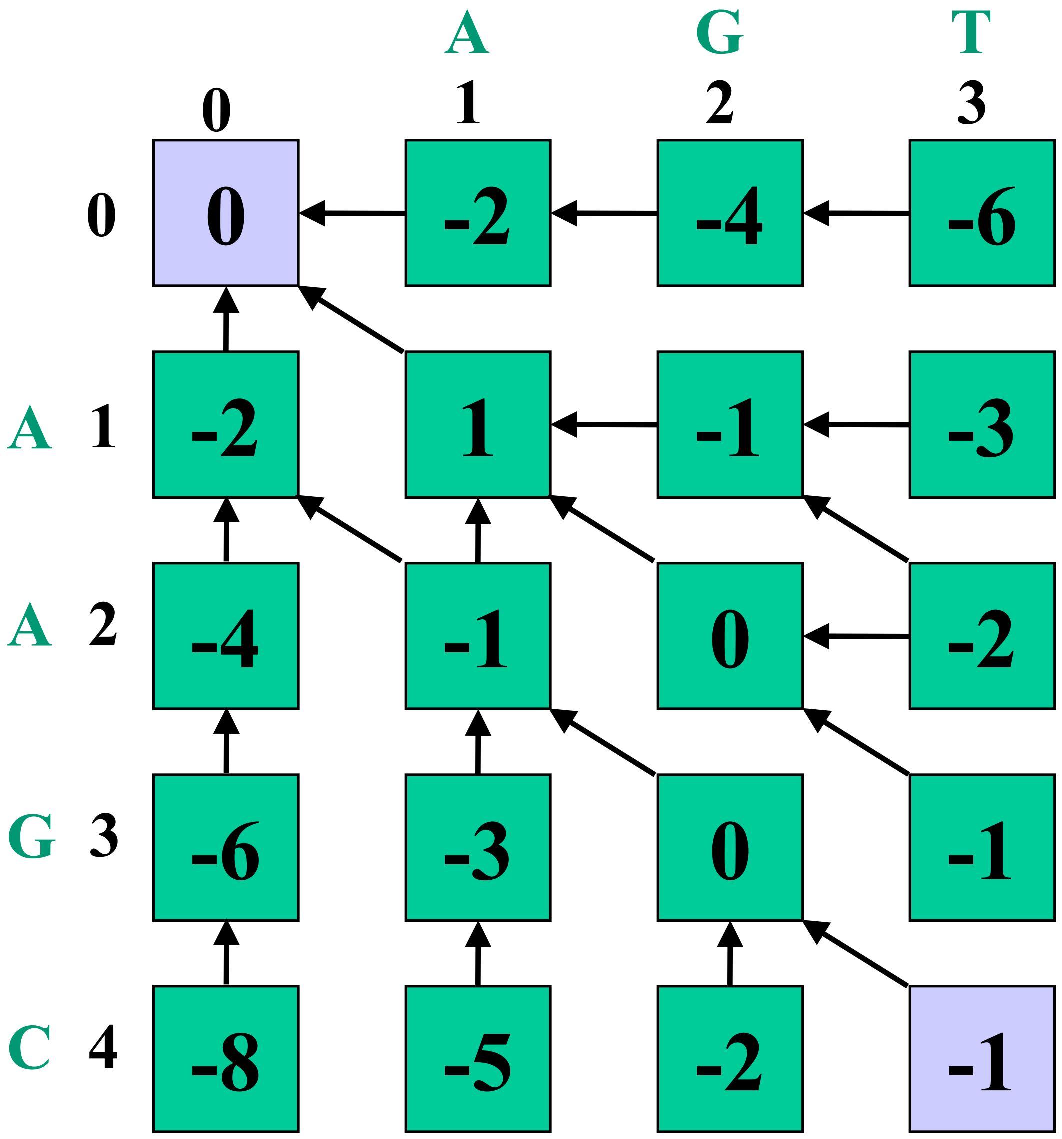
Scoring

Gap = -2

Mismatch = -1

Match = 1

The Full Matrix



$$\begin{tikzcd}
 & & & A & G & T \\
 & & 0 & 1 & 2 & 3 \\
 & 0 & 0 & -2 \arrow[l] & -4 \arrow[l] & -6 \arrow[l] \\
A & 1 & -2 \arrow[u] & 1 \arrow[ul] & -1 \arrow[l] & -3 \arrow[l] \\
A & 2 & -4 \arrow[u] & -1 \arrow[ul] \arrow[u] & 0 \arrow[ul] & -2 \arrow[ul] \arrow[l] \\
G & 3 & -6 \arrow[u] & -3 \arrow[u] & 0 \arrow[ul] & -1 \arrow[ul] \\
C & 4 & -8 \arrow[u] & -5 \arrow[u] & -2 \arrow[u] & -1 \arrow[ul]
\end{tikzcd}$$

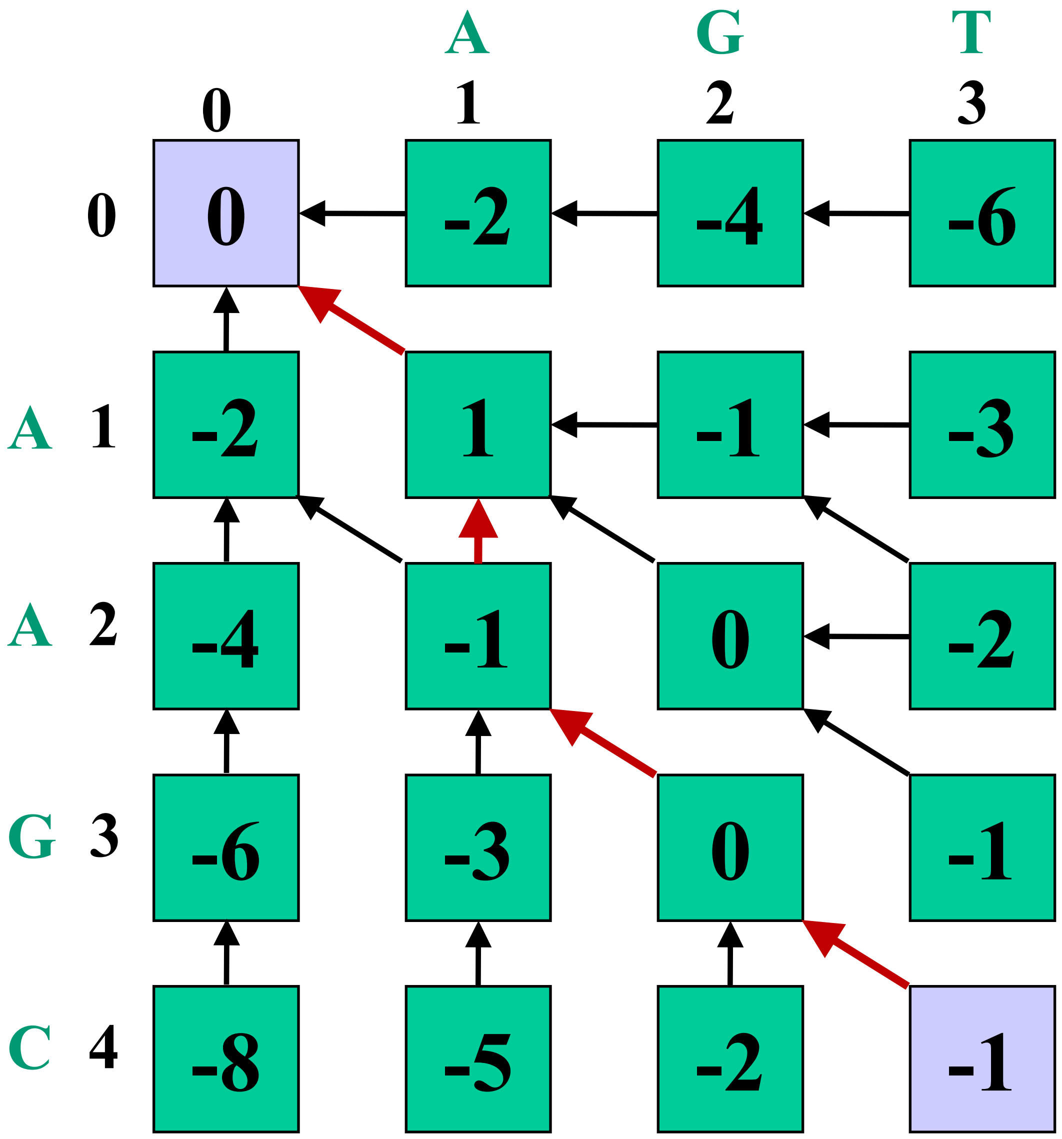
Scoring

Gap = -2

Mismatch = -1

Match = 1

The Full Matrix - Traceback



Scoring

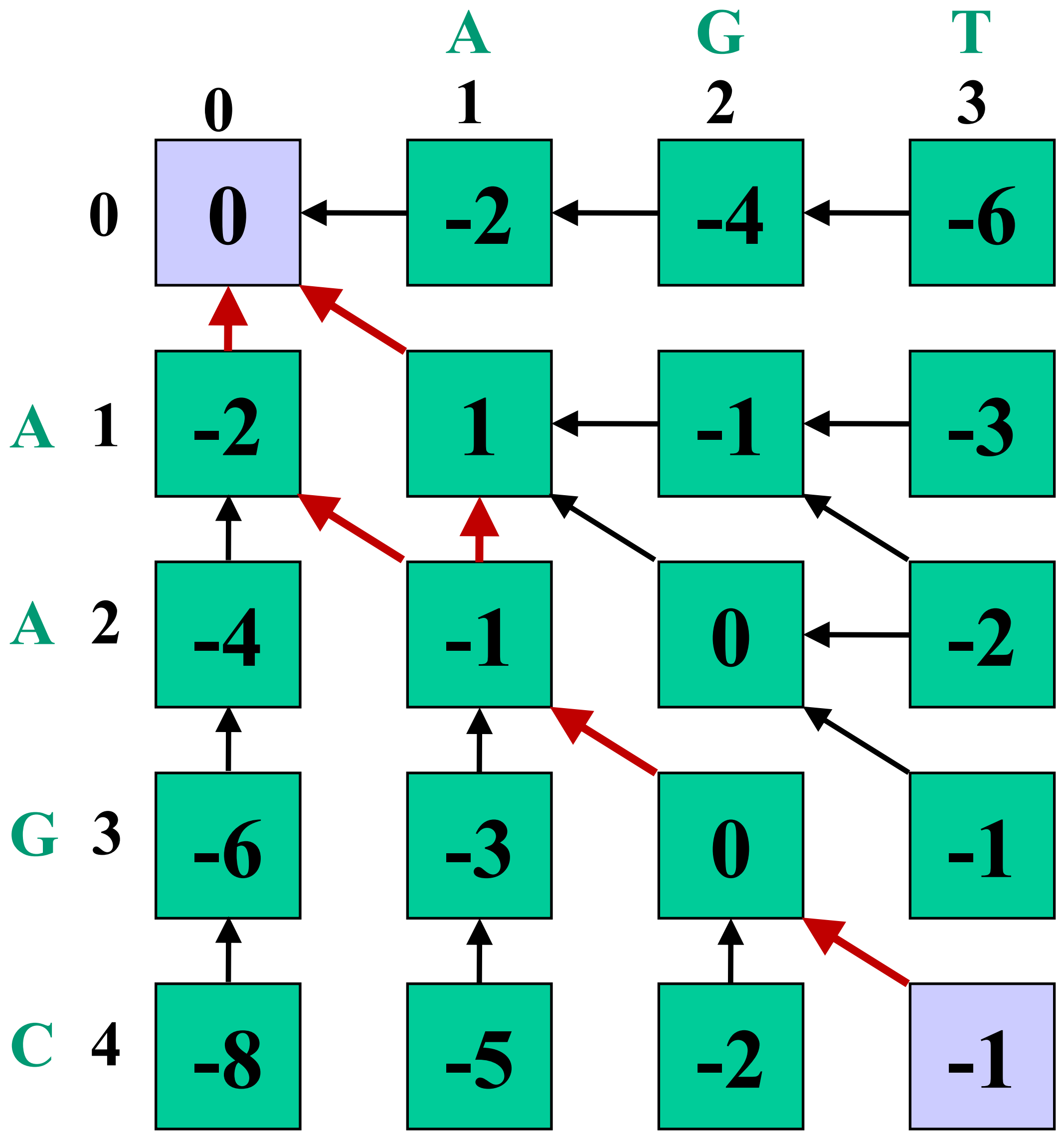
Gap = -2

Mismatch = -1

Match = 1

A		G	T
A	A	G	C

The Full Matrix - Traceback



Scoring

Gap = -2

Mismatch = -1

Match = 1

	A	G	T
A	A	G	C

Needleman-Wunsch

Optimal Global Alignment of Two Sequences

- Very simple computationally
  - Just fill in the table to get the best score
  - Use traceback to get the best alignment(s)
- Ensures that there is no duplicated work
  - Only need to compute each sub-alignment once!
  - Every subpart that is calculated will be needed
- *Effectively* searches all possible alignments!
  - How does it do this???

Needleman, Saul B. & Wunsch, Christian D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". Journal of Molecular Biology. 48 (3): 443–53.

Running Time Analysis

- We have to store (n+1)x(m+1) numbers
- Each number requires constant time to compute (3 sums and a max)
- So, O(NM) or $O(N^2)$ for sequences of same length

n	Enumeration	DP
10	184,756	100
20	1.40E+11	400
100	9.00E+58	10,000

Dynamic Programming

Needleman-Wunsch Algorithm is a type of dynamic programming algorithm

- Optimality substructure
  - Larger problems can be solved with the optimal solutions for subproblems
- Overlapping subproblems
  - The same subproblem needs solution many times

Dynamic Programming in Practice

- Setting up dynamic programming
  1. Identify "states" (e.g. i,j) and "decisions" (match, gap, etc…)
  2. Recursion formula: larger problems = F(subparts)
  3. Traversal order: sub-results ready when you need them
     - Computation order matters! (bottom-up, but not always obvious)
  4. Remember choices: typically F() includes min() or max()
     - Need representation for storing pointers

- Then start computing
  1. Systematically fill in table of results, find optimal score
  2. Trace-back from optimal score, find optimal solution

Exercise for You

- Often gaps come in bunches
- The probability of getting 10 gaps in a row is higher than 10 different 1 gap events
- Solution: Affine Gap Penalty
- g(n) is penalty for gap of length n:

$$\mathbf{g(n)= (-\underset{\text{Gap-open penalty (d)}}{\underset{\nearrow}{d}}) - (n-1)\ \underset{\text{Gap-extension penalty (e)}}{\underset{\nearrow}{e}}}$$

What is the new update rule for F(i,j)?