Optimal Sequence Alig-ment

Plan 9 from outer space
https://commons.wikimedia.org/w/index.php?curid=1793456

Overview

- The alignment problem
- The dynamic programming solution
- Pairwise alignment: exact global and local solutions
- Multiple alignment and the cost of perfection

Recap: protein scoring

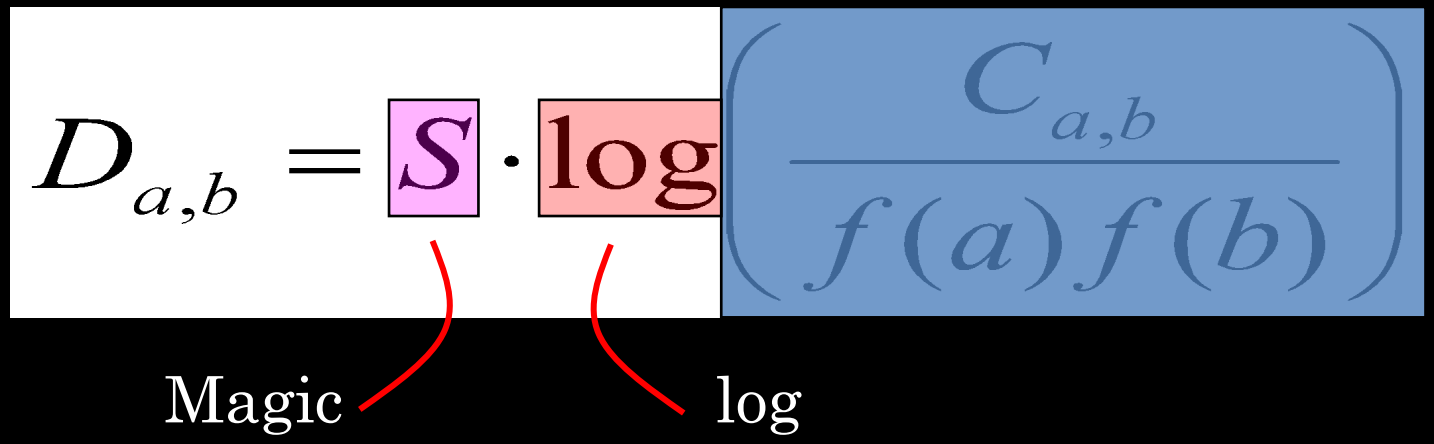
$$\left(\frac{C_{a,b}}{f(a)f(b)} \right)$$

C matrix – scaled frequencies of change from amino acid *a* to amino acid *b* (based on observed changes in some set)

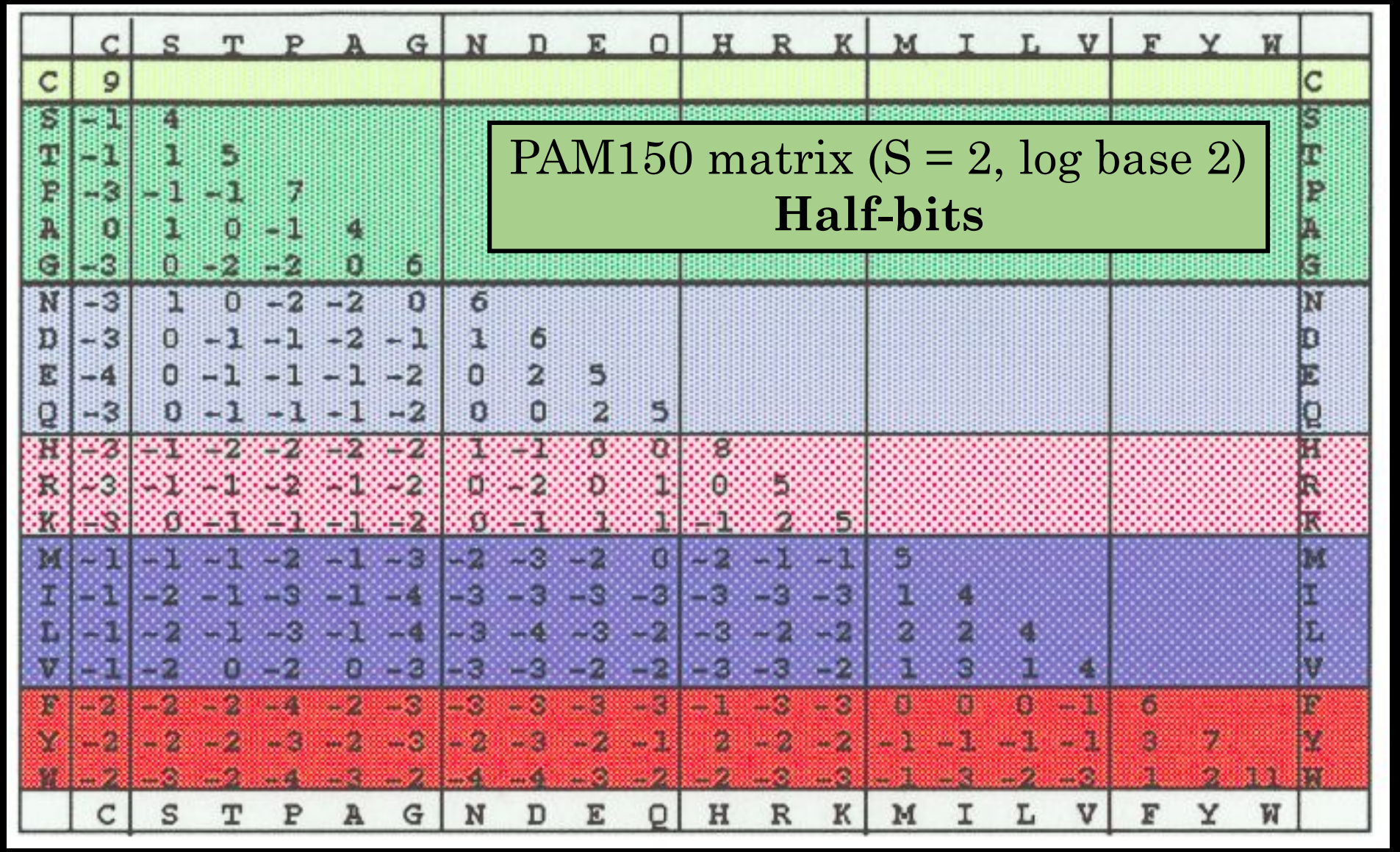
Expectation based solely on frequencies of amino acids (changes not favoured / disfavoured)

Better than random: ratio > 1
Random: ratio = 1
Worse than random: ratio < 1

log transformation



Better than random: $D_{a,b} > 0$
Random: $D_{a,b} = 0$
Worse than random: $D_{a,b} < 0$



https://bioinformaticamente.com/2020/10/22/alignment-algorithms/

DNA matrix

Something like this usually works:

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

Or this:

	A	G	C	T
A	1	0.5	-1	-1
G	0.5	1	-1	-1
C	-1	-1	1	0.5
T	-1	-1	0.5	1

Back to the alignment problem

Given a scoring scheme…

and a set of homologous sequences S…

introduce gaps if necessary to generate an alignment that optimizes the score

So let's make some alignments!

Sequence S_1: length m
Sequence S_2: length n

So let's make some alignments!

Sequence S_1: length m

Sequence S_2: length n

In total, there are $\binom{n+m}{m}$ possible alignments of these sequences

n = m = 2:

4!/2!2! = 6 possibilities

AB-- --CD	AB- -CD	AB- C-D	AB CD	A-B -CD	-AB CD-

n = m = 10: 184,756 possible alignments

Alignment of 2 sequences, each 100 amino acids in length:

$= 9.05485147 \times 10^{58}$ possibilities

Brute force is *not* going to work here…

Scaling of algorithms: Big-O Notation

What rate do resources (time, memory) increase as the input increases?

Asymptotic: upper bound on growth as input tends to infinity

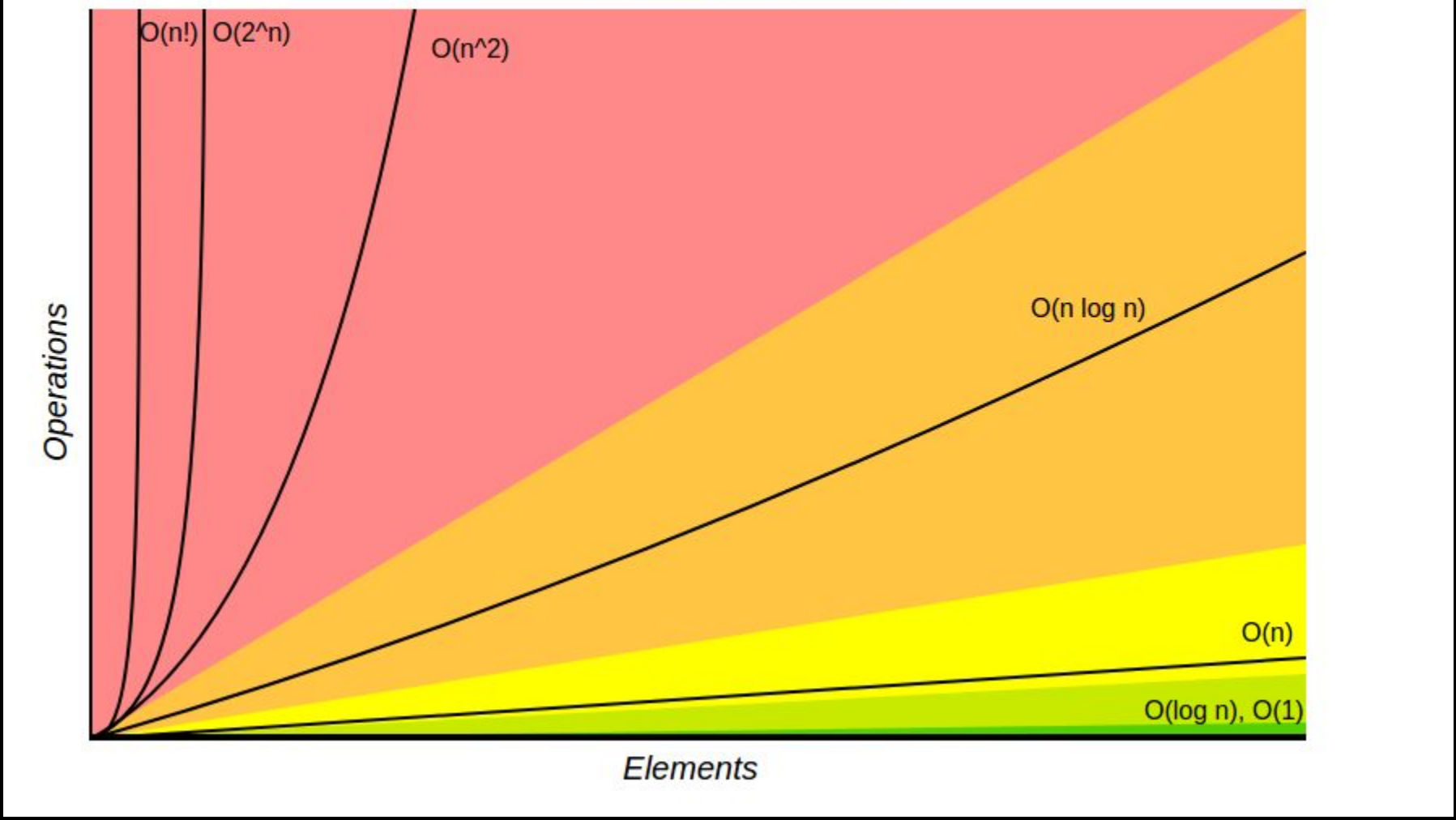
Keep only dominant term:

$3n^2 + 7n + 42$ is $O(n^2)$

Brute-force sequence alignment:

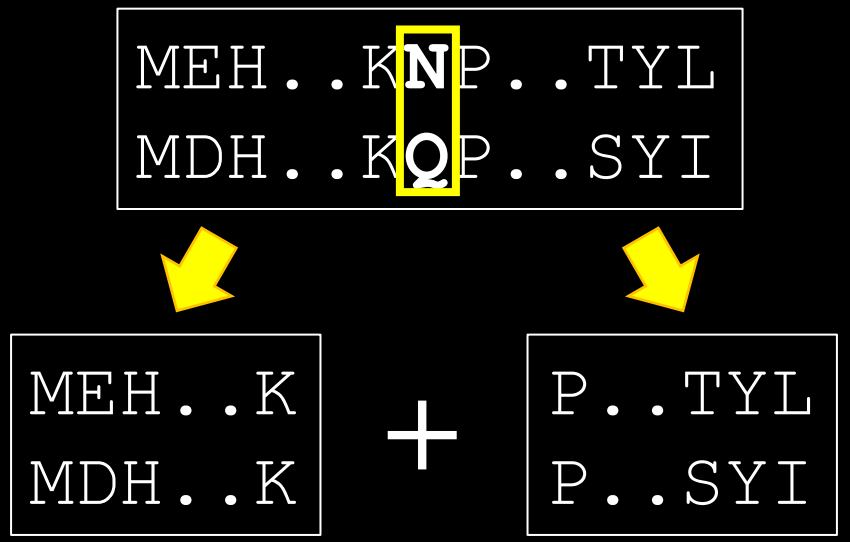
2n choose n ~ $O(4^n)$

Big O complexity



The Key to Alignment

If we were given a **point X** within an optimal alignment of S_1 and S_2, we could **split on X** and solve each problem independently



But we **don't know** any X, so divide and conquer isn't going to work

However…

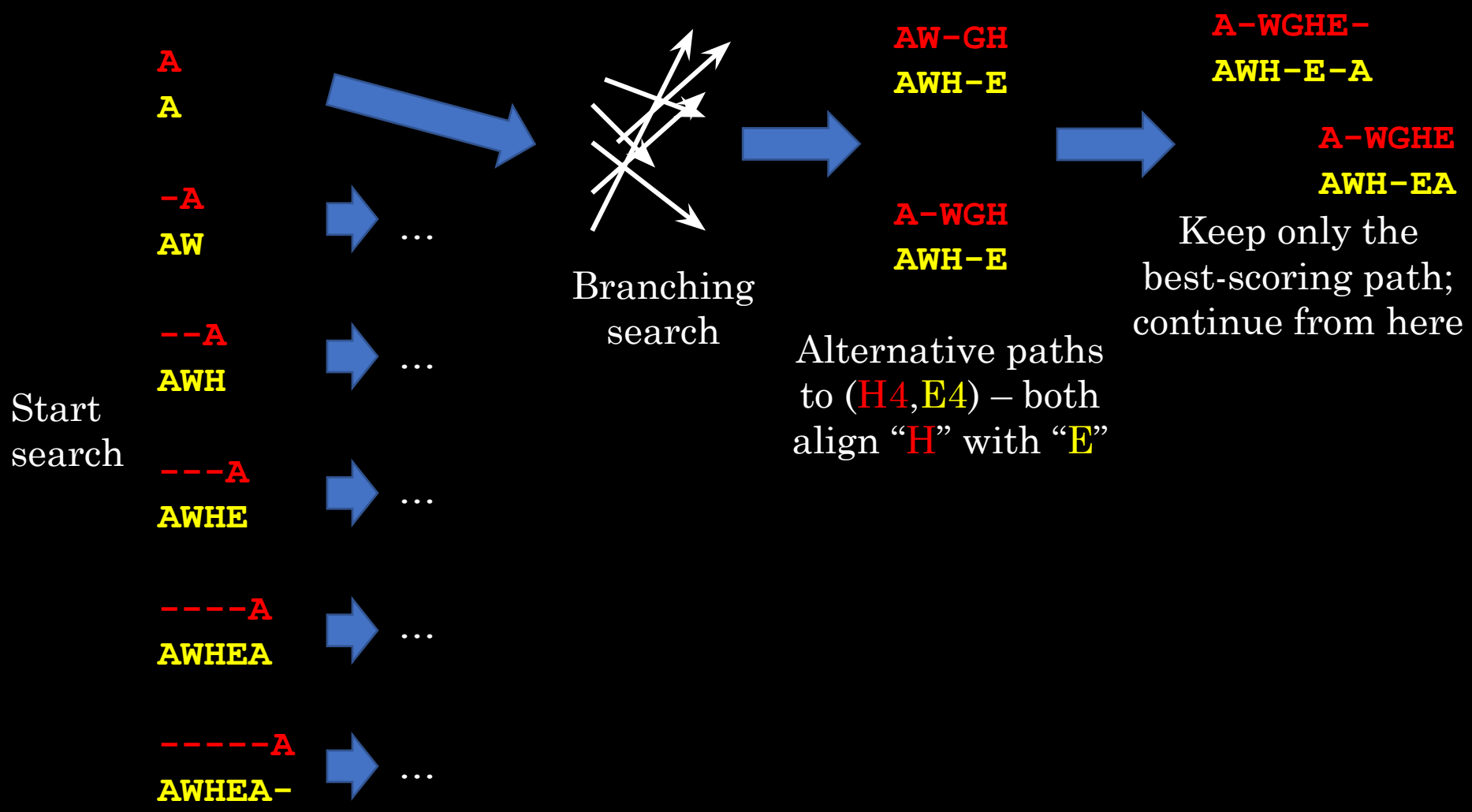
In searching for the best alignment:

- Start at the beginning of the sequences and consider every possible X

- BUT -

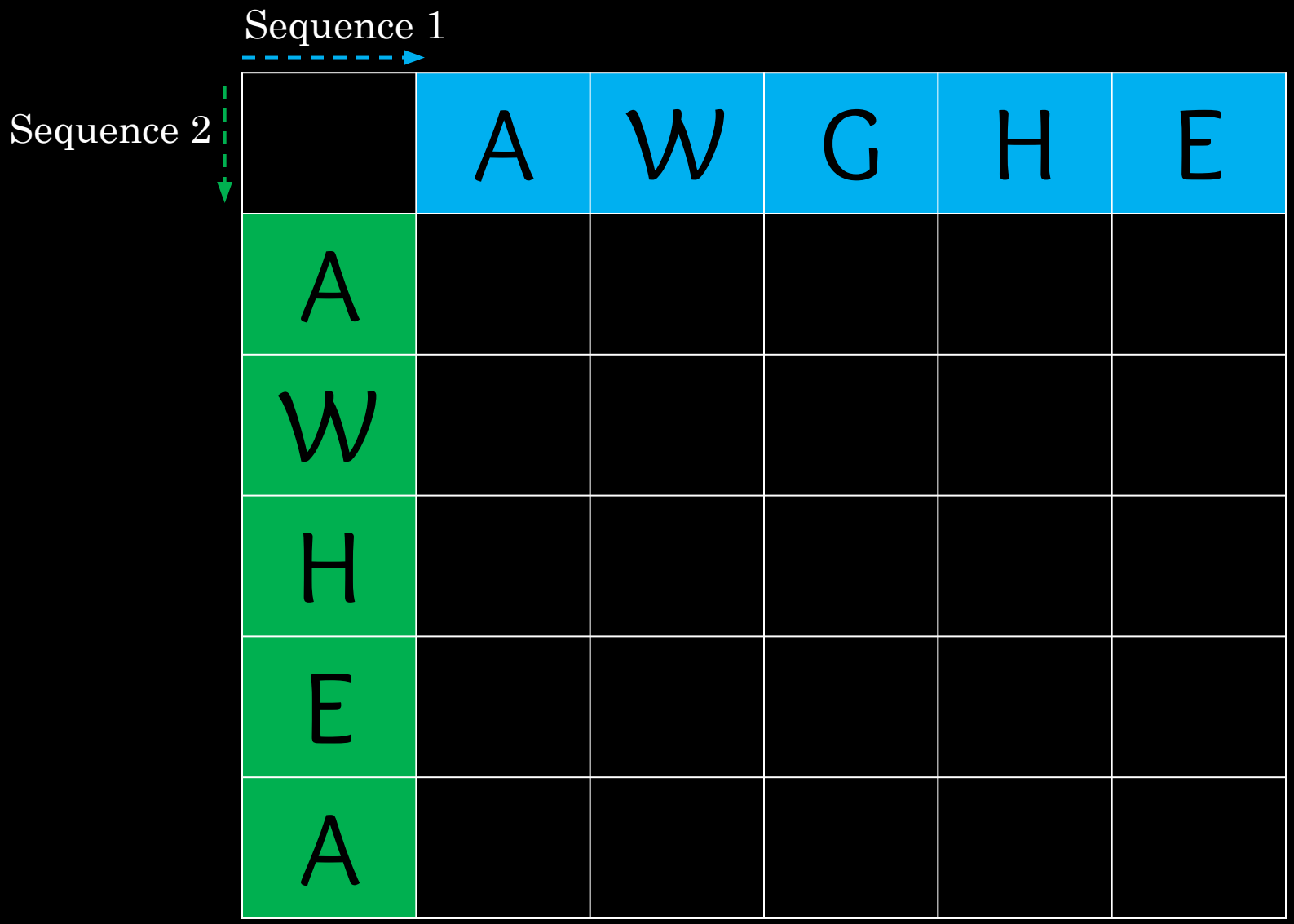
- Store only the best path (series of matches and gaps) that leads us to X

Consider an alignment search for AWGHE vs AWHEA:

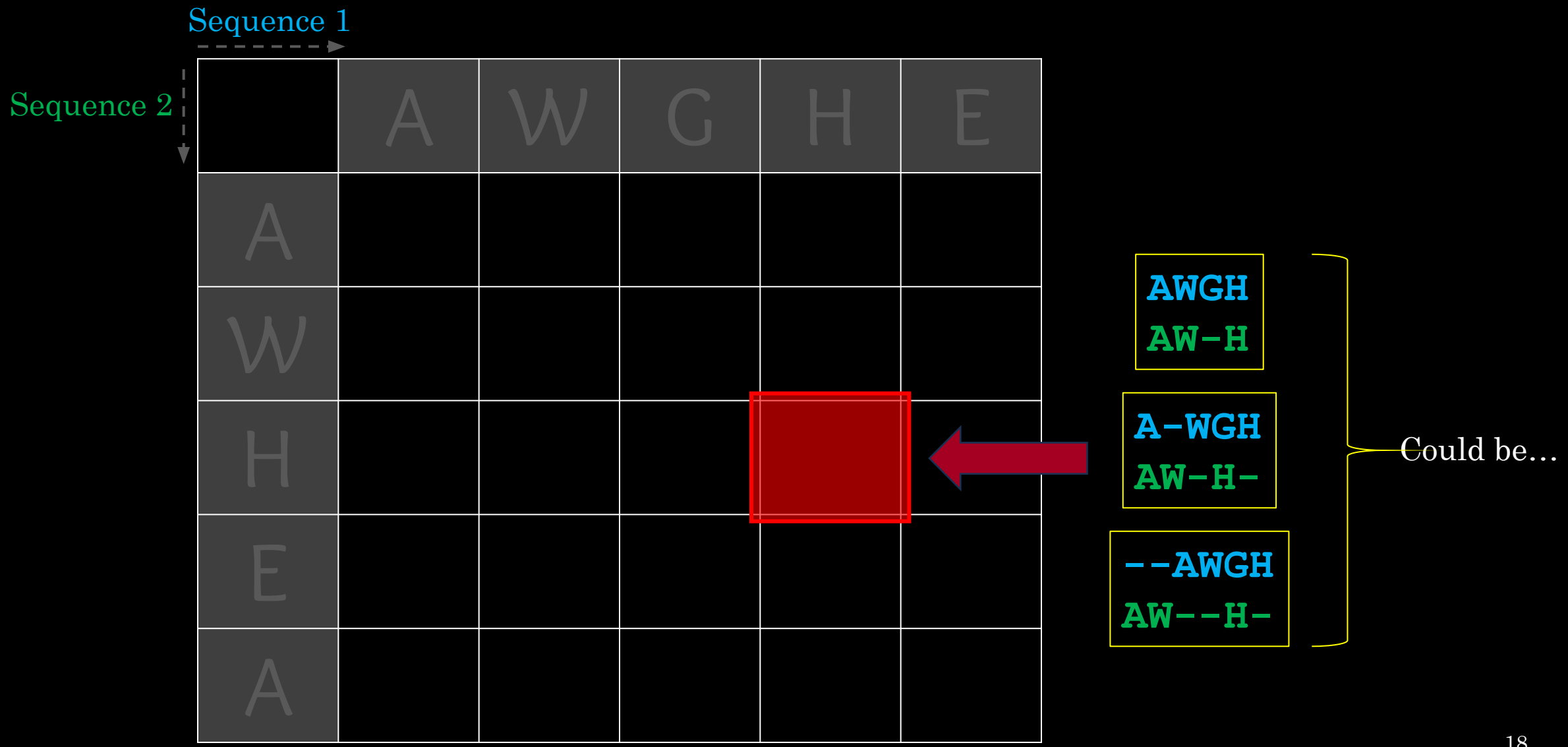


= Dynamic Programming

Consider an alignment of AWGHE vs AWHEA:



Each cell in the grid represents a point in the alignment where the corresponding residues have been added to the alignment



Determine the best score for every possible X from the two sequences

AWGHE vs. AWHEA	A	W	G	H	E
A	Best →(A,A)				
W					
H				Best → (H,H)	
E					
A					Best → (A,E)

Filling the matrix

We need our substitution matrix S and gap penalty scheme G

(we'll start with a linear gap penalty G = -gd)

For each possible X, consider the three immediate precursors

Upper left-hand corner: set to 0

S = **PAM250**
g = **5**

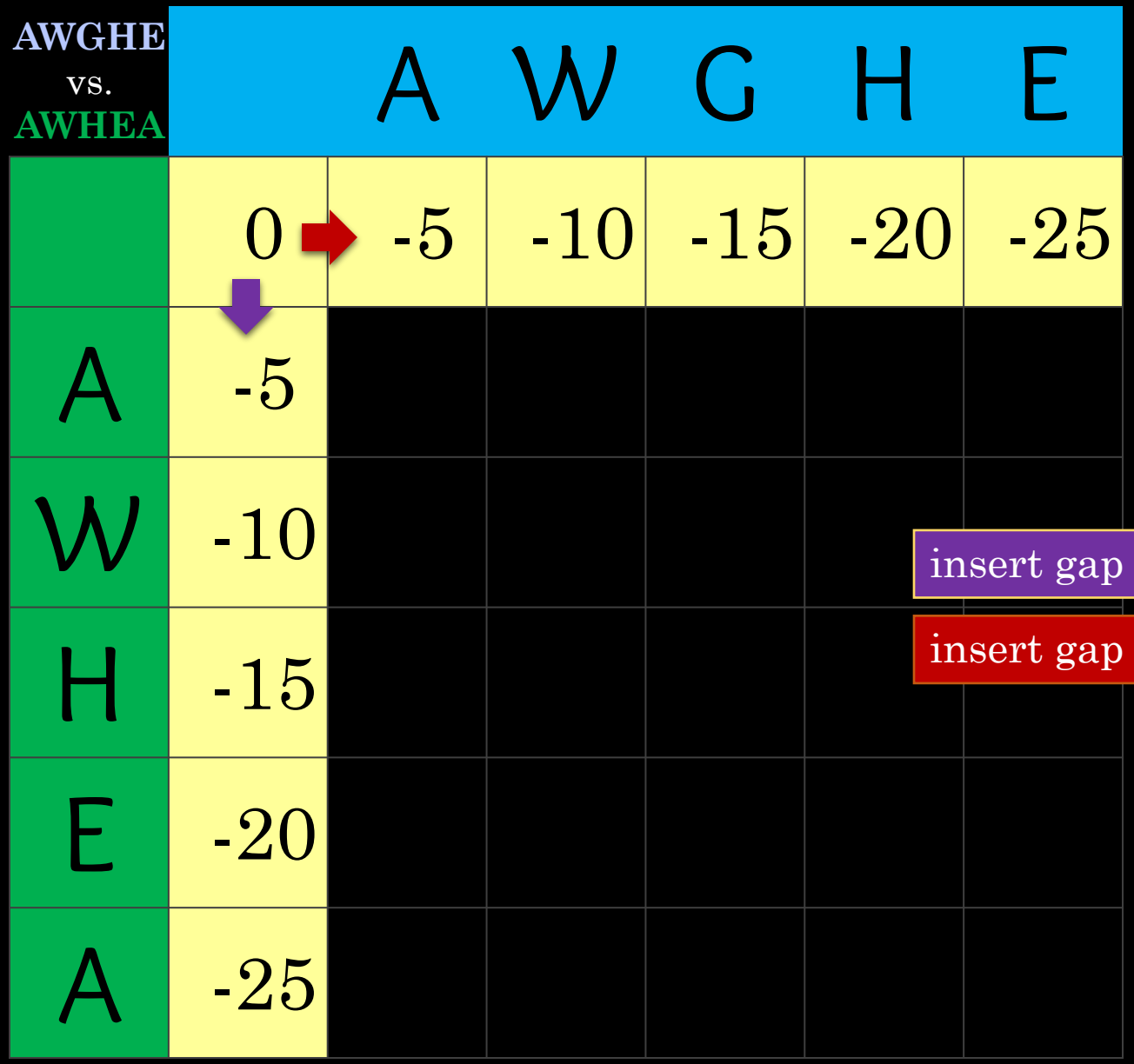
AWGHE vs. AWHEA		A	W	G	H	E
	0					
A						
W						
H						
E						
A						

First row / columns: runs of initial gaps

S = **PAM250**
g = **5**

-----AWGHE
AWHEA-----

AWGHE-----
-----AWHEA



Filling (A,A): What is the best path to get there?

S = **PAM250**
g = **5**

AWGHE vs. AWHEA		A	W	G	H	E
	0	-5	-10	-15	-20	-25
A	-5					
W	-10					
H	-15					
E	-20					
A	-25					

insert gap in AWGHE

insert gap in AWHEA

match

–	A
A	

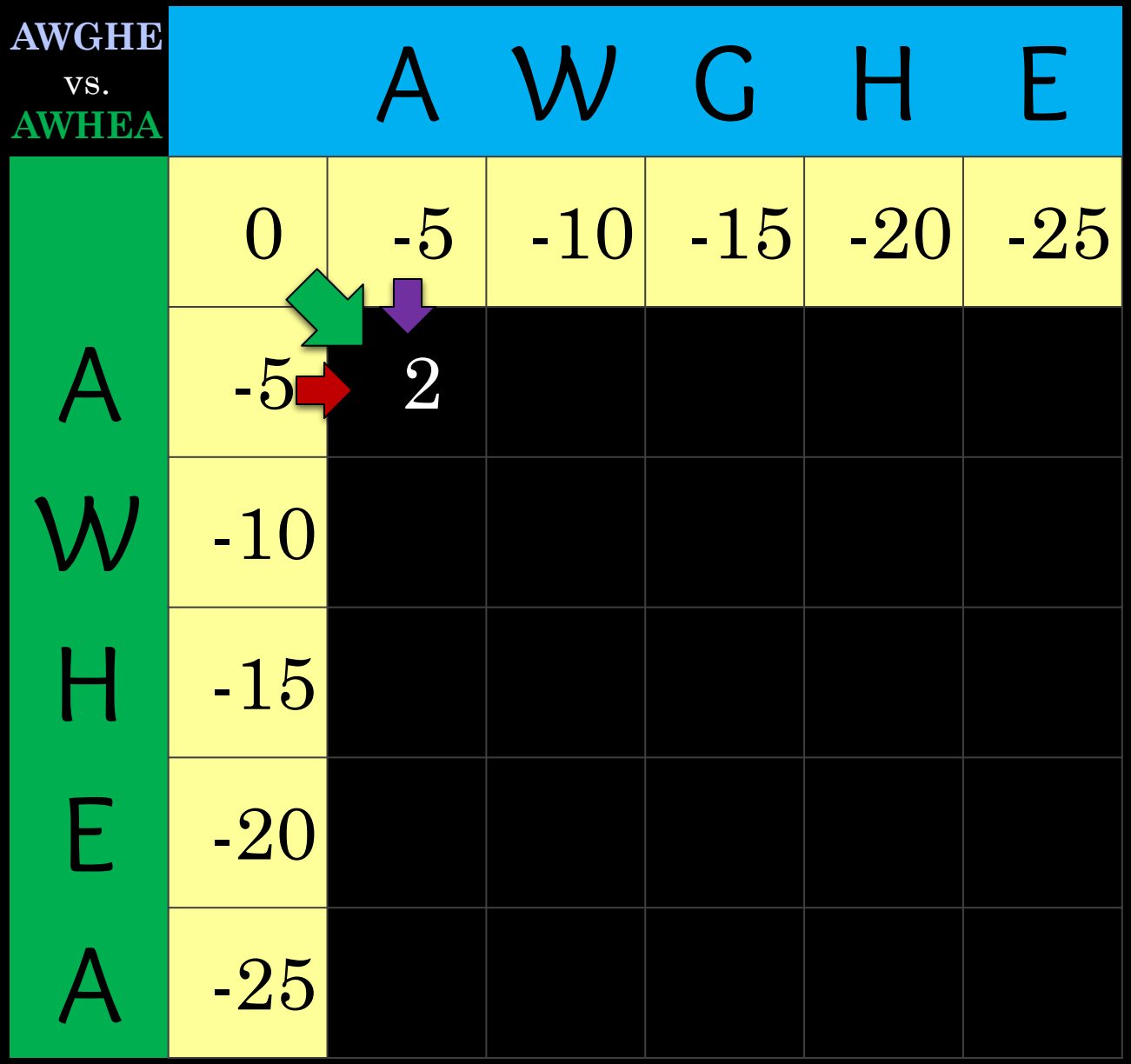
A	
–	A

A
A

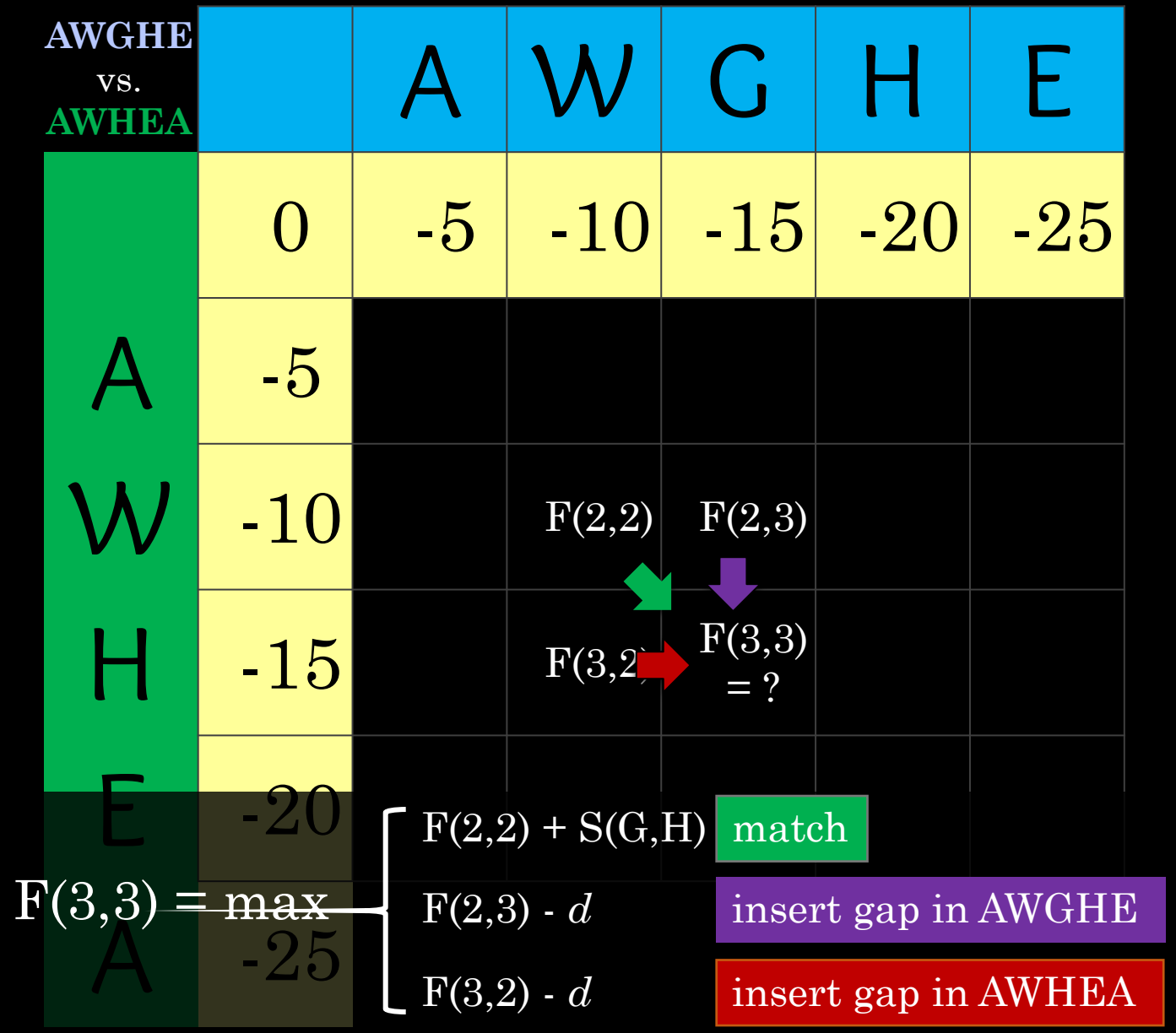
Choosing the best path to (A,A)

S(A,A) = 2

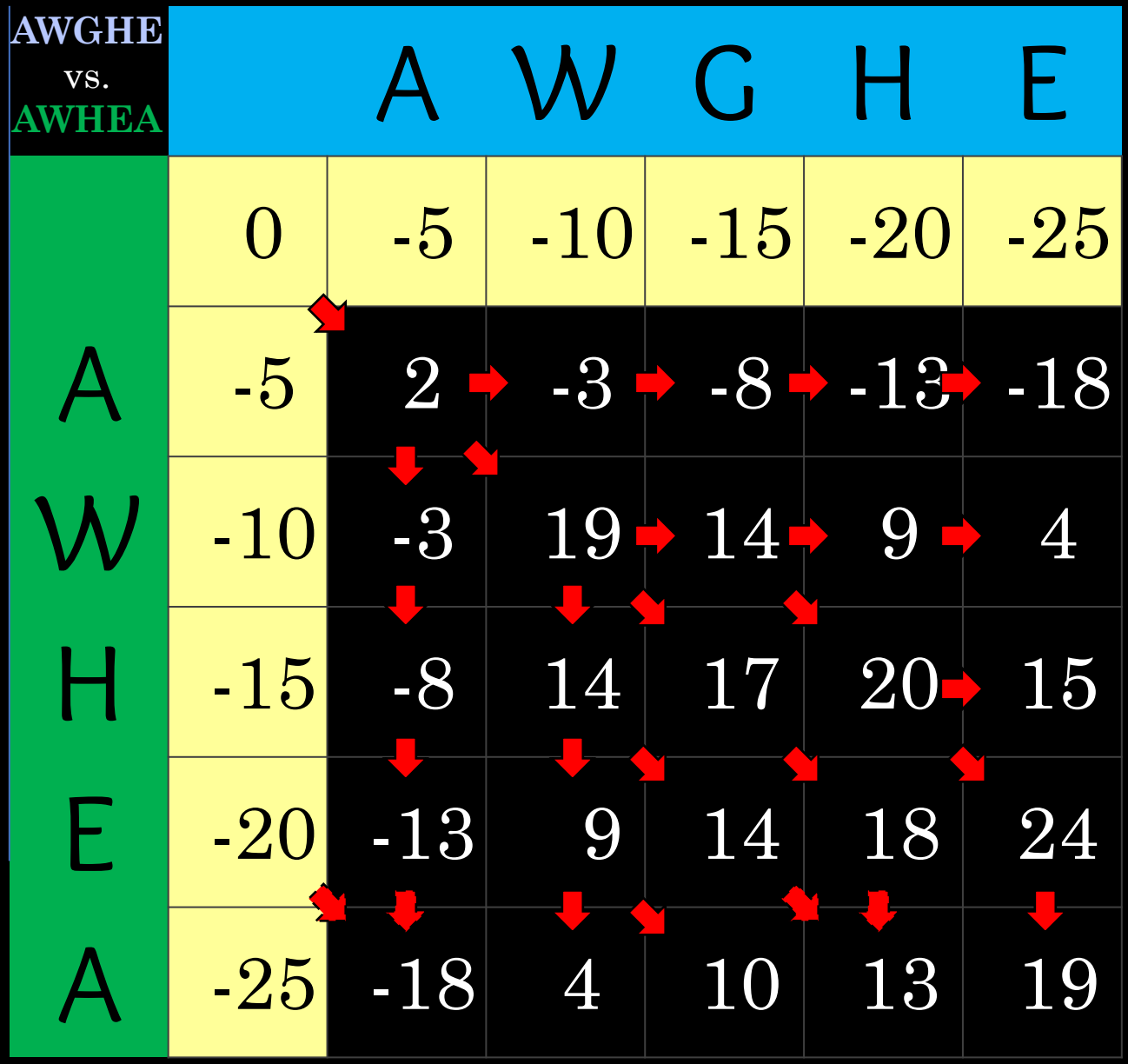
Therefore:
Insert -10
Insert -10
Match 2



General form: best path to any matrix cell



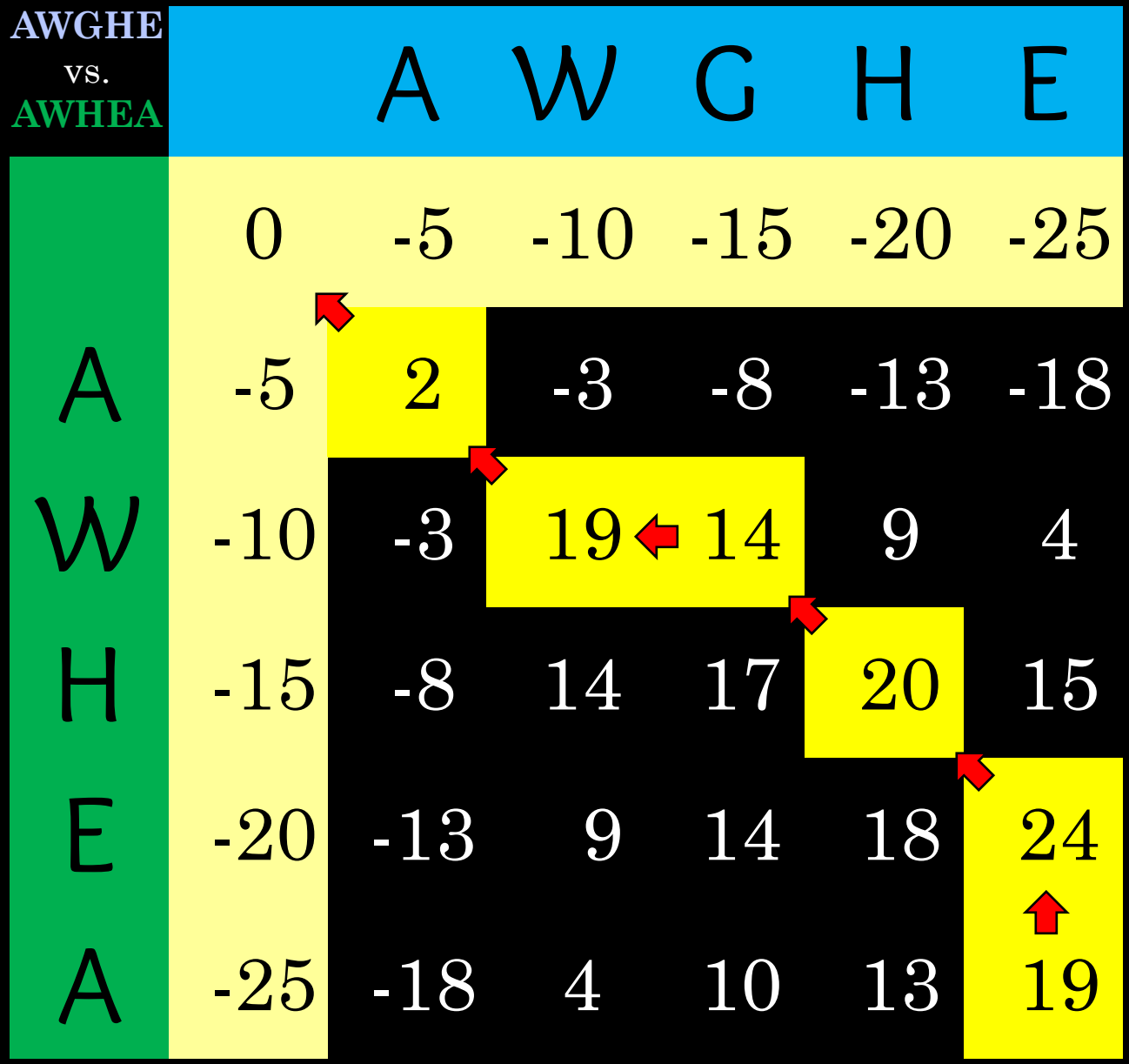
Remember paths INTO (not out of) each cell



Global Exact Alignment: Needleman-Wunsch

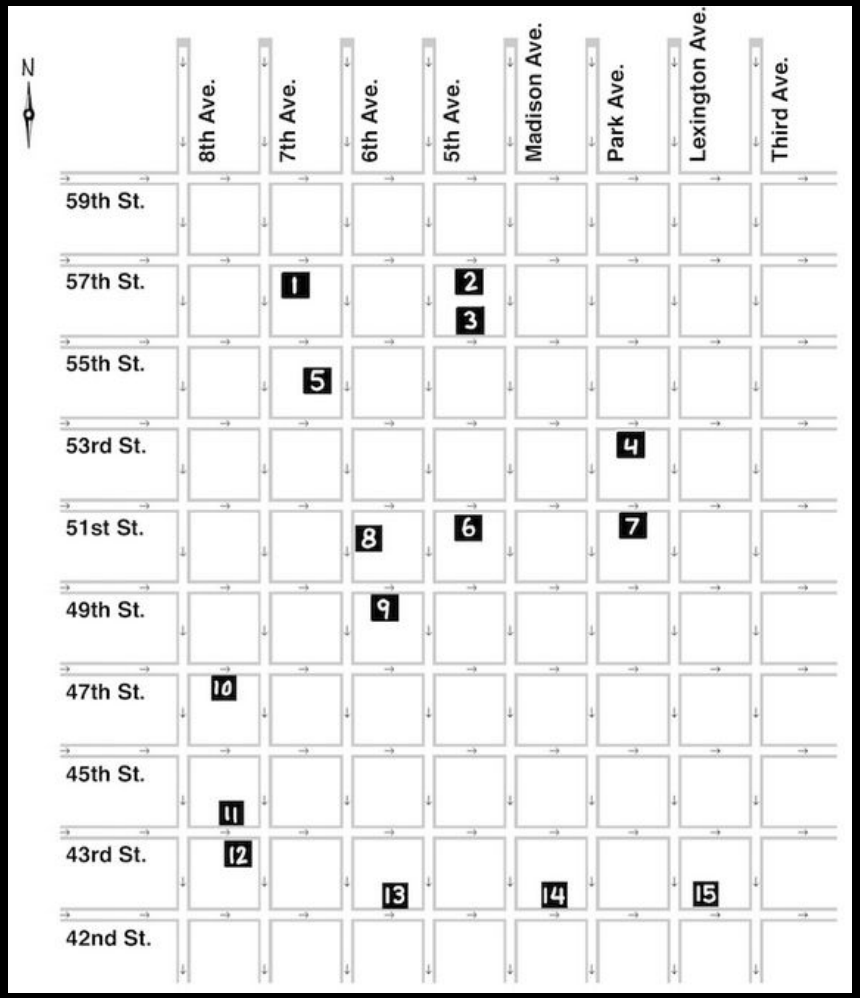
Since we have retained the best path to each F(x,y) in the matrix, we can trace back from the endpoint F(m,n) to the origin and retrieve the optimal alignment path

Traceback for the optimal alignment



AWGHE-
AW-HEA

Same problem as finding longest path through directed acyclic graph



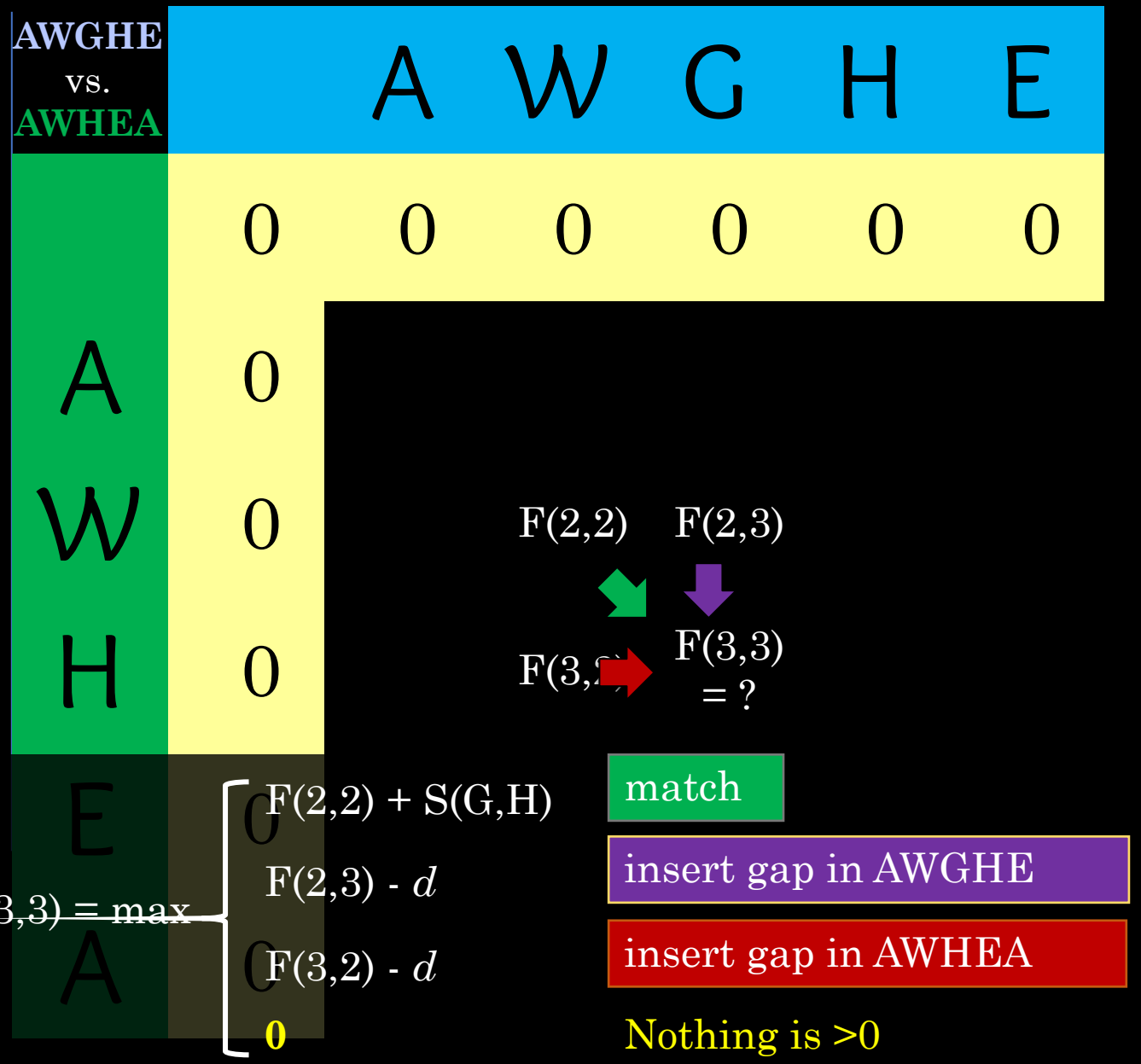
https://www.bioinformaticsalgorithms.org/bioinformatics-chapter-5

Local Exact Alignment: Smith-Waterman

- Only return ‘good’ sub-alignments of the whole problem

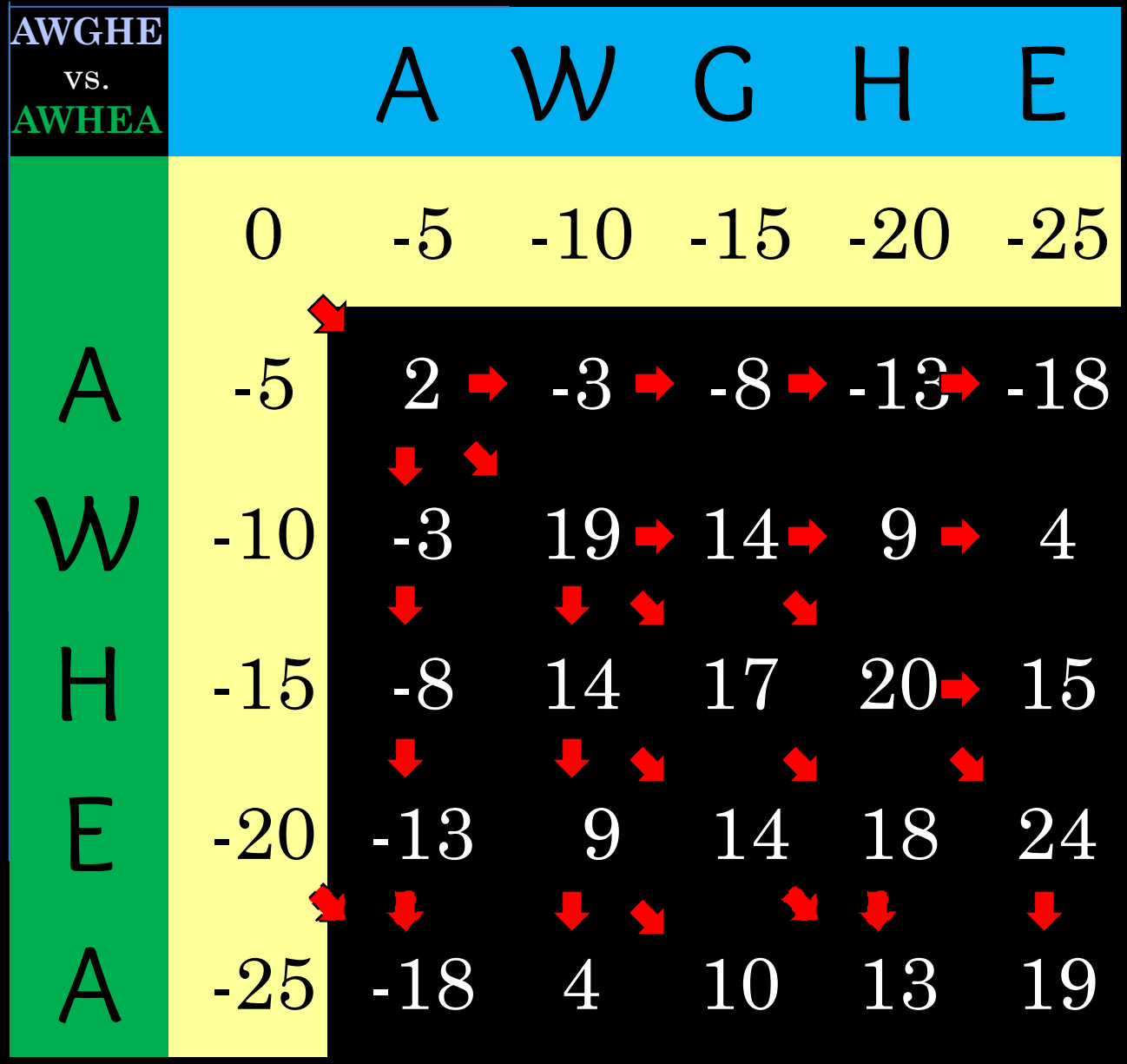
- Useful, for instance, when

One more rule for local alignment



The Needleman-Wunsch Matrix, Again

AWGHE-
AW-HEA



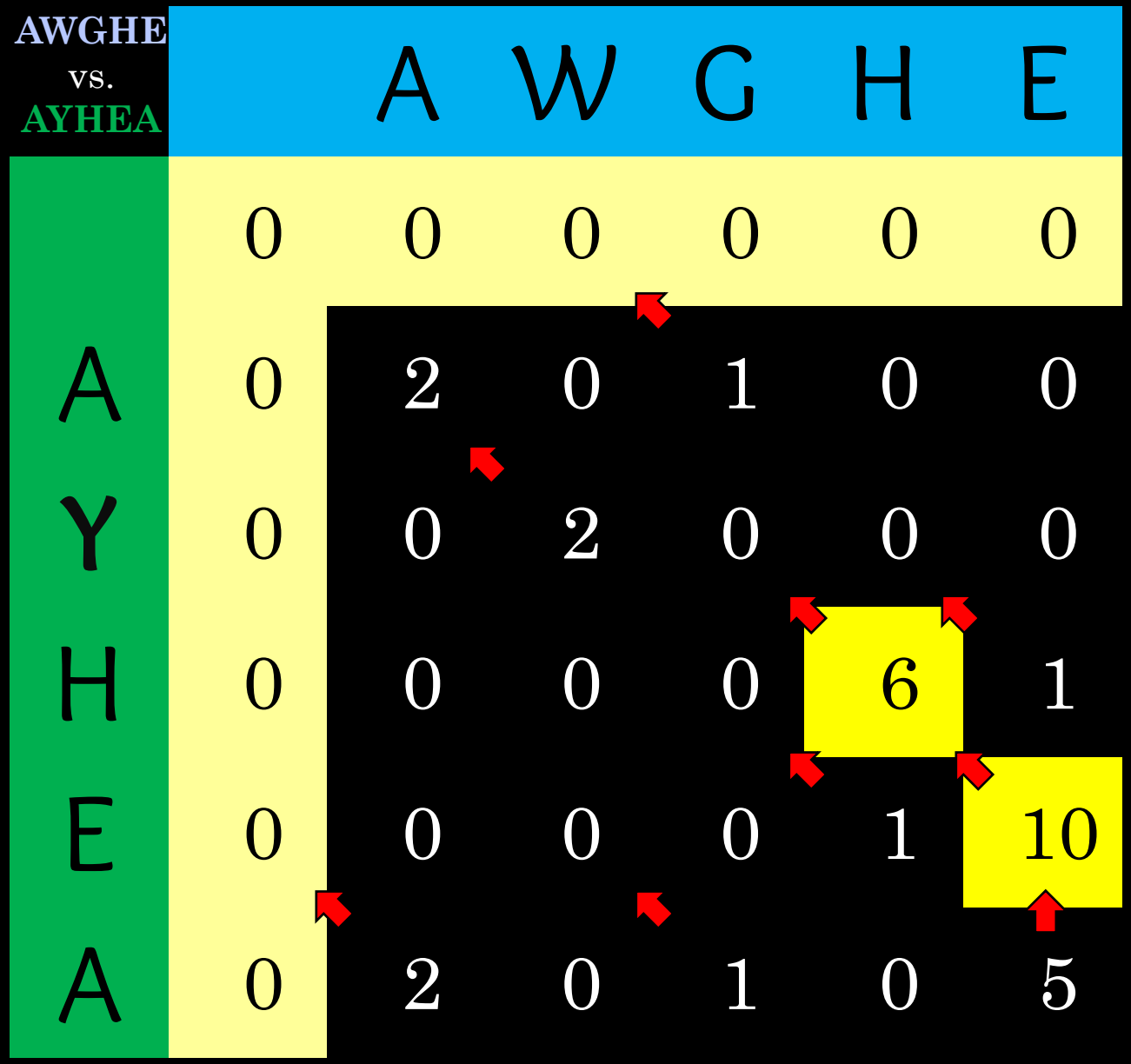
The Smith-Waterman matrix

Slightly modified (non-trivial) S-W example

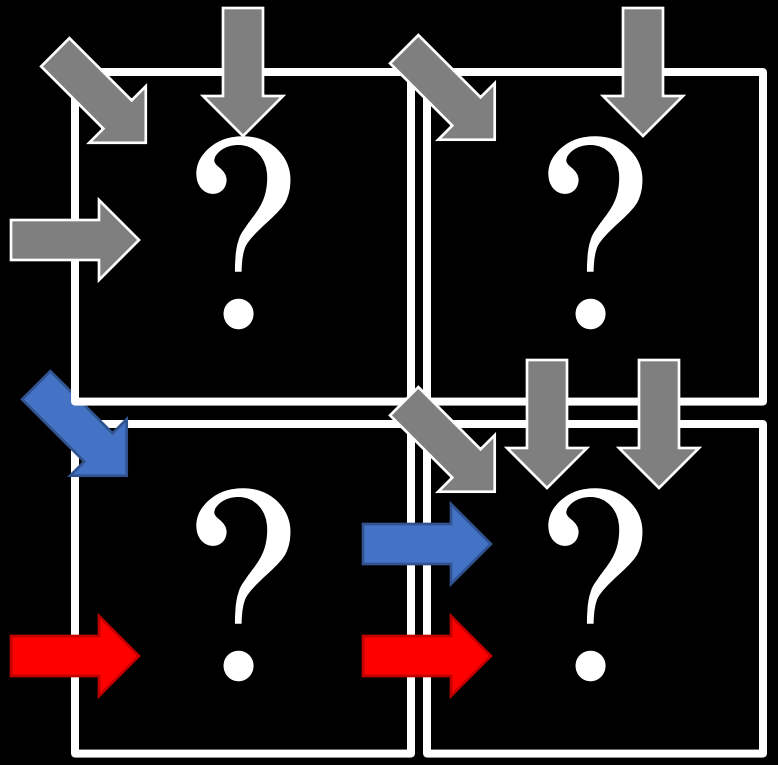
Find the **largest** value in the matrix, and trace back from there to 0

HE

HE



Affine Gap Penalties



Opening a new gap
(cost = d)

Extending a gap
(cost = e)

A horizontal move now has two possible costs; we need to consider both alternatives

(and therefore store the best scores for each box given horizontal, vertical, or diagonal entry)

Significance of S-W Alignments

Permutation test: Randomize the alignment n times, compute mean and standard deviation

Compute Z-score for each replicate:

Mean of replicates

True score

$$Z(A,B) = \frac{S(A,B) - \tilde{m}}{\tilde{\sigma}}$$

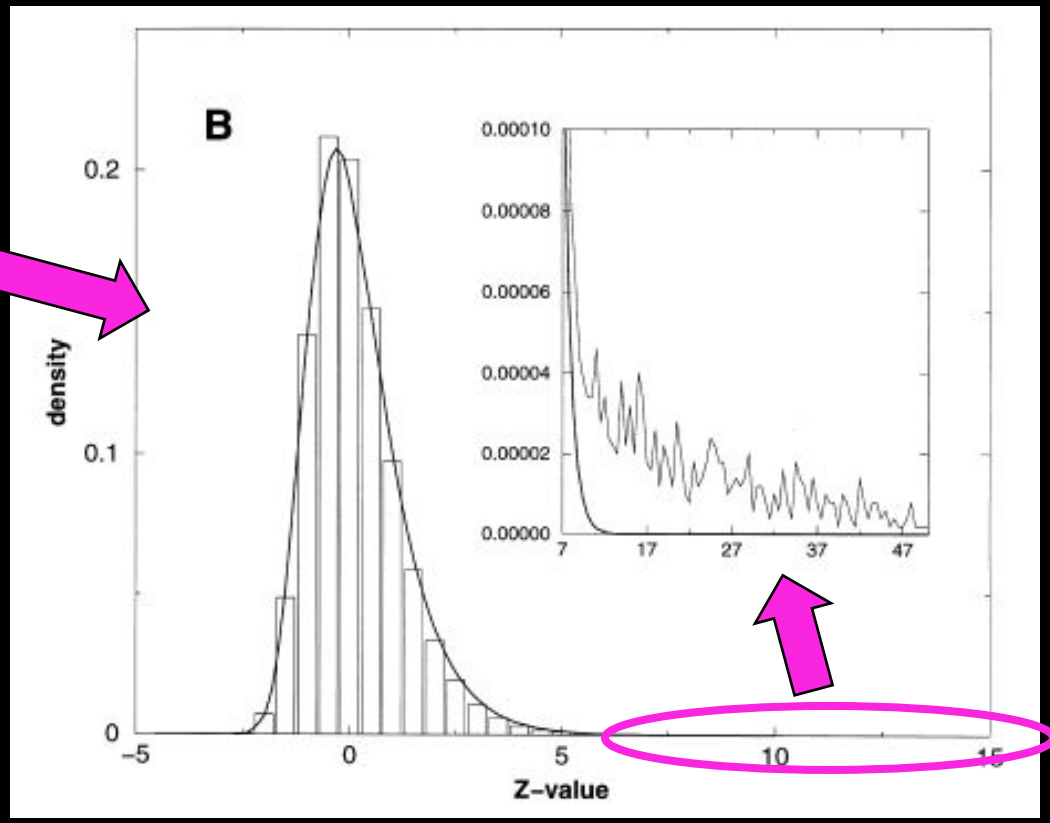
Standard deviation of replicates

Comet et al., *Computers and Chemistry* (1999)

Significance of S-W Alignments

$$Z(A,B) = \frac{S(A,B) - \tilde{m}}{\tilde{\sigma}}$$

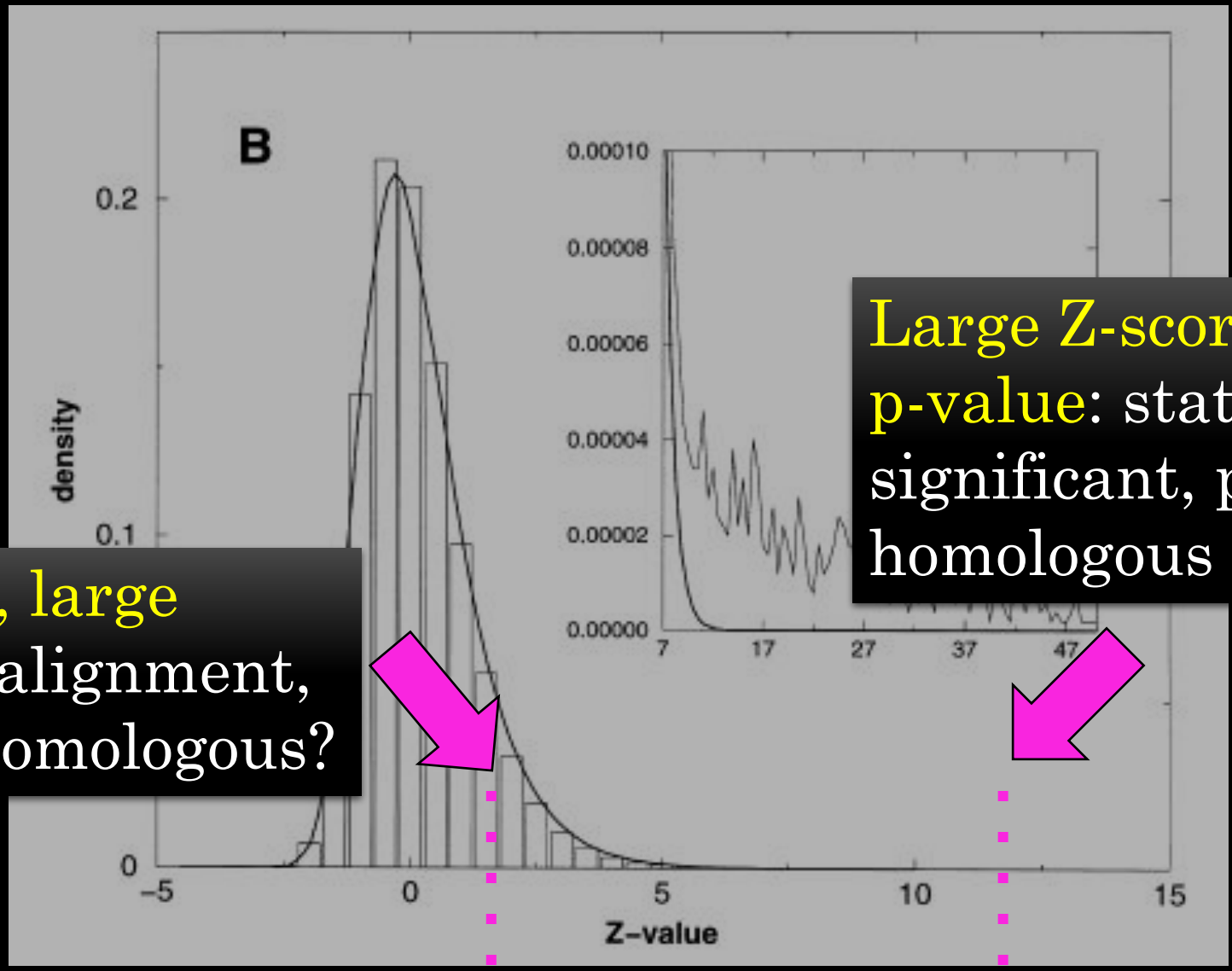
Curve = null model of Z-score fit to Gumbel extreme value distribution



Statistically "significant" alignments (small p-value)

Comet et al., *Computers and Chemistry* (1999)

Alignment Significance



Small Z-score, large p-value: poor alignment, possibly not homologous?

Large Z-score, small p-value: statistically significant, probably homologous

Comet et al., *Computers and Chemistry* (1999)

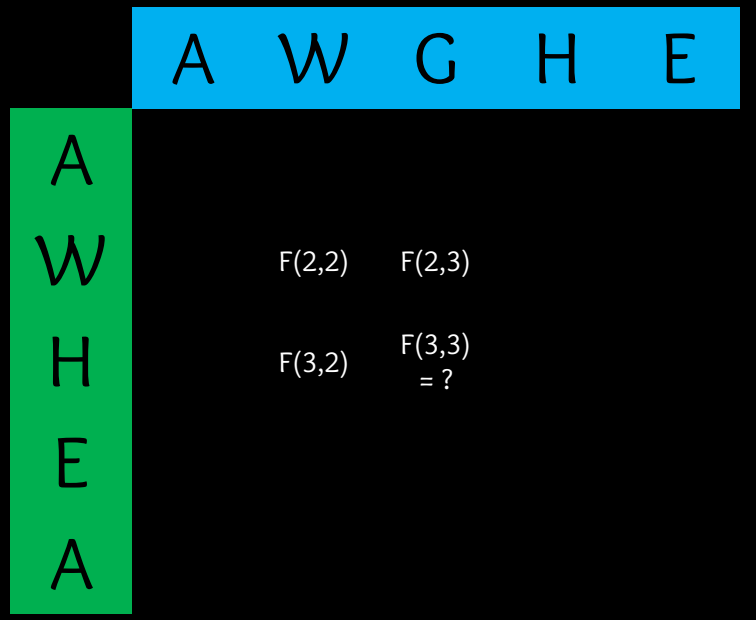
Alignment Complexity

- For each possible matching of a residue from sequence S_1 with a residue from S_2 , we need to carry out a constant number of computations and comparisons

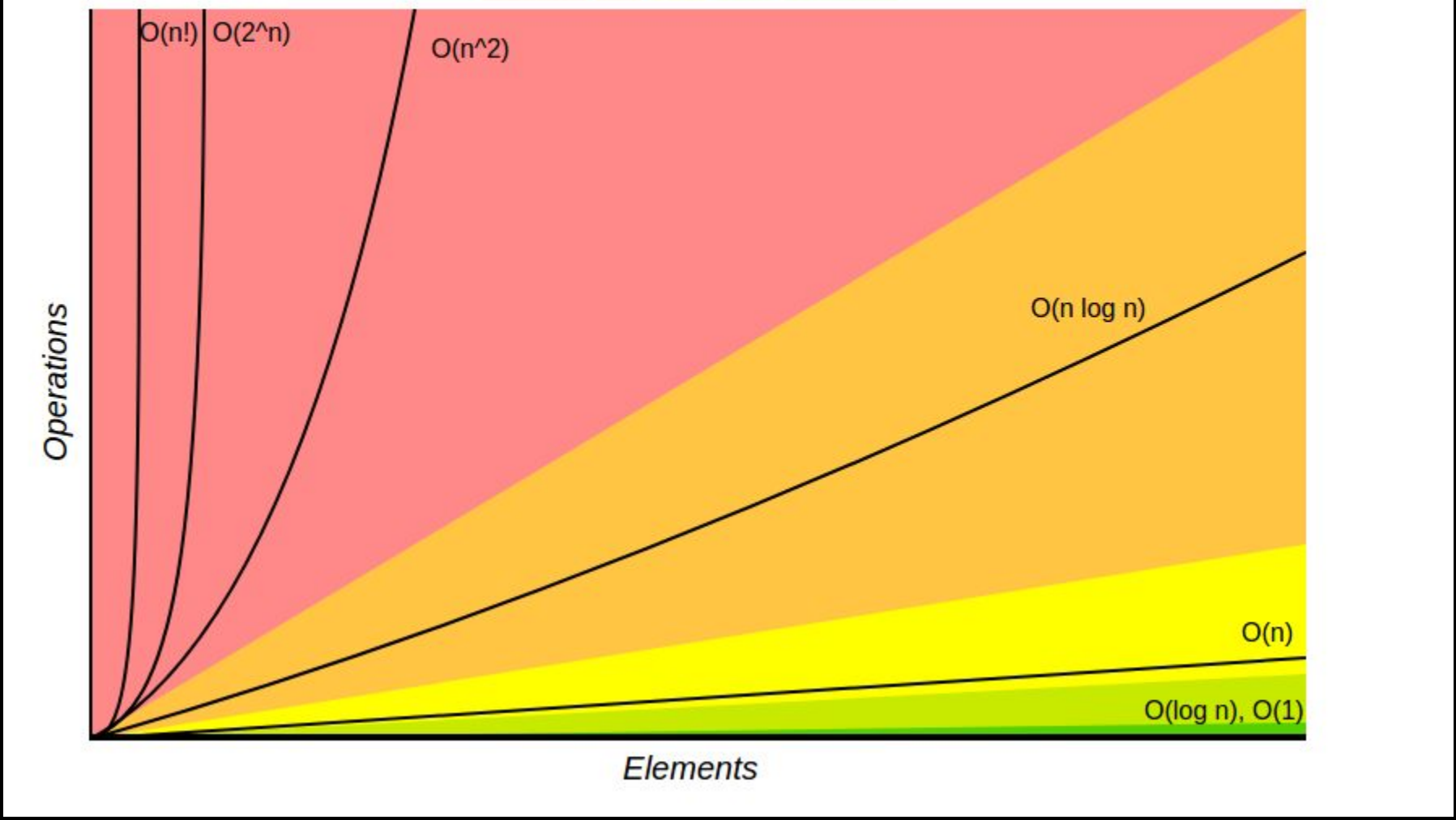
- Total = 3 x m x n = O(mn)

- ~ O(n^2) if we assume $m \cong n$

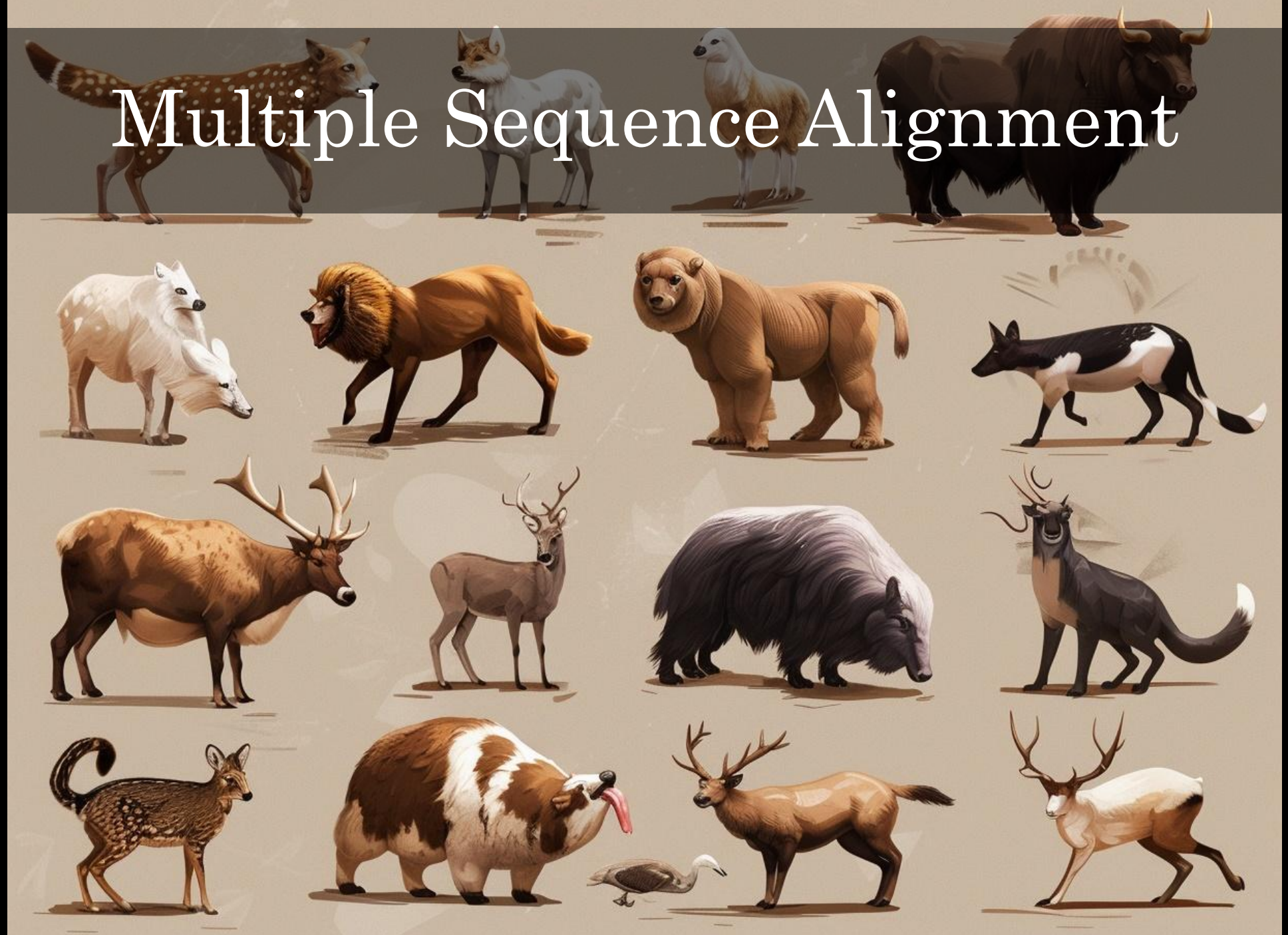
- Quadratic scaling!



Big O complexity

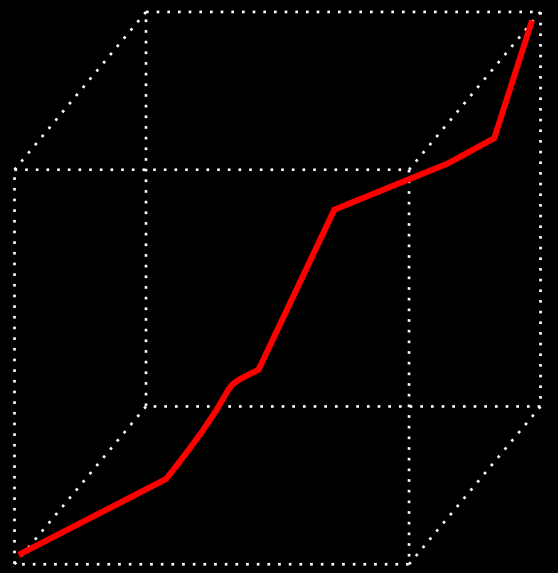


Multiple Sequence Alignment



Multiple Sequence Alignment

- Dynamic programming on k sequences, each of length n requires construction of a k-dimensional matrix with n^k entries



- = $O(n^k)$

- Therefore exponential in the number of sequences!

Scoring MSAs

- In pairwise alignment, we are optimizing the score between two sequences

- When aligning 3 or more sequences, instead optimize the sum of pairs score:

```
Sequence 1   N
Sequence 2   Q
Sequence 3   Q
Sequence 4   D
```

SP(N,Q,Q,D) =

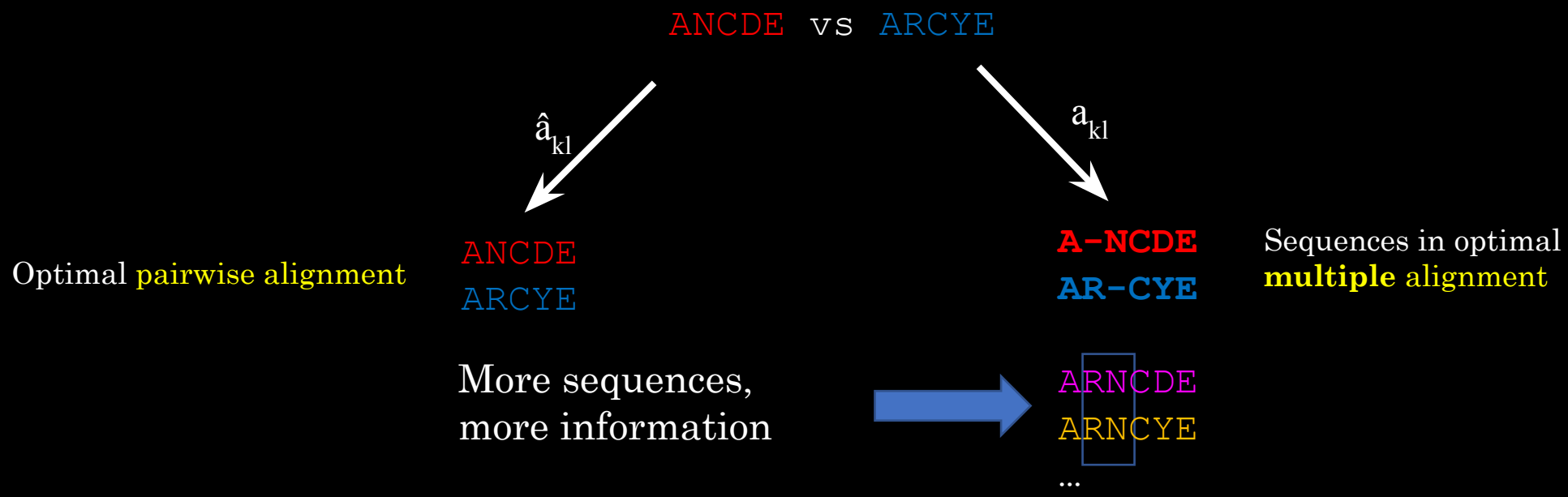
S(N,Q)
+ S(N,Q)
+ S(N,D)
+ S(Q,Q)
+ S(Q,D)
+ S(Q,D)

A Key Principle of MSAs

- Aligning everything at once using exact DP = n^k: BAD
- Aligning pairs of sequences using exact DP, then doing something with this information = $\binom{k}{2}(n^2)$: acceptable

An Important Observation

The best alignment between a pair of sequences may not appear in the optimal multiple alignment



And

- The score of the optimal multiple alignment $S(a)$ can be no greater than the sum of optimal pairwise alignments $S(\hat{a}^{kl})$

$$\sum_{k<l} S(a^{kl}) \leq \sum_{k<l} S(\hat{a}^{kl})$$

- In general, the multiple alignment score will be less than the sum of all pairwise alignments

But how much less???

MSA (Carrillo and Lipman, 1988)

- If we can establish a lower bound σ on the multiple alignment score, then we constrain each S(a^{kl}):

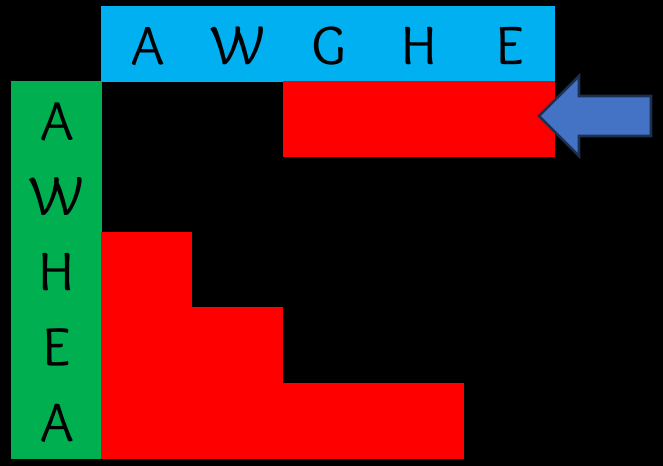
$$S(\hat{a}^{kl}) - S(a^{kl}) \leq \sum_{k'<l'} S(\hat{a}^{k'l'}) - \sigma$$

Remember: sum of all best possible pairwise alignments!

σ high: $S(a^{kl})$ must be close to $S(\hat{a}^{kl})$
Sets a bound on "how much less"

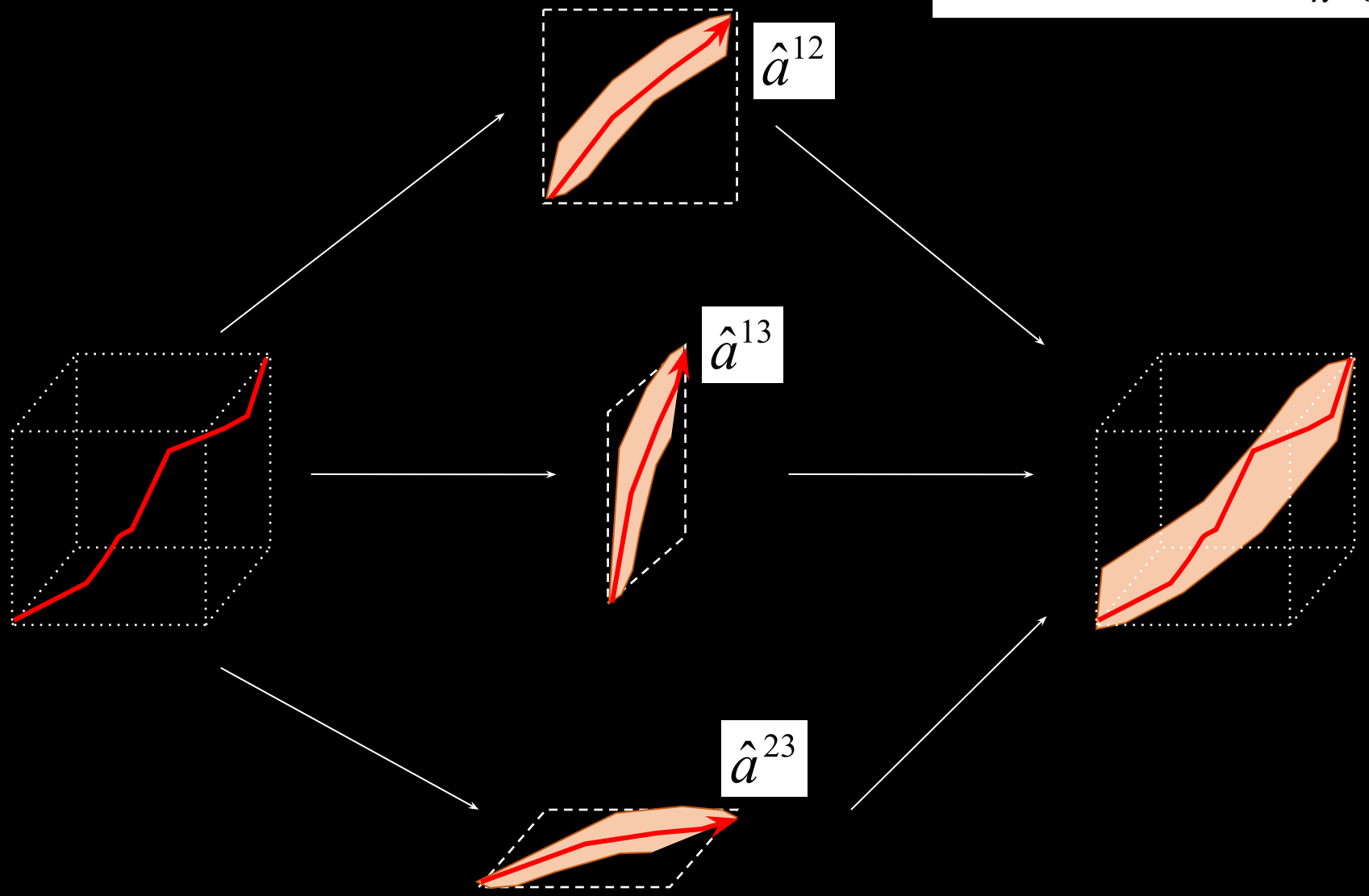
The Consequence of σ

- We can compute $S(\hat{a}^{k'l'})$ for each pair of sequences, and fill the DP grid
- Any cell of the DP grid that gives $S(a^{kl})$ less than σ can be discarded



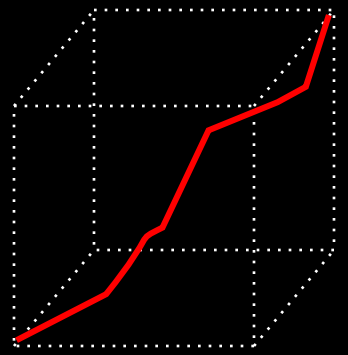
Unacceptable paths: score is too low!

Constrain each pairwise alignment to score no less than $\sigma + S(\hat{a}^{kl}) - \sum_{k'<l'} S(\hat{a}^{k'l'})$



The Last Step

- Multidimensional dynamic programming, restricted to "acceptable" band



- Still $O(n^k)$, but hopefully faster!

So we need all optimal pairwise alignments (again, way cheaper than naïve MSA)

We also need σ. How can we find it?

σ too large: we don't effectively constrain the search space!

σ too small: we may not find an optimal alignment!

Summary

- Dynamic programming allows the calculation of optimal pairwise alignments (for a given scoring scheme!)
- As soon as we go from 2 to >2 sequences, the exponential time complexity of the algorithm makes it impractical
- Need heuristics!