"If you align expectations with reality, you will never be disappointed."
-- Terrell Owens

Dynamic Programming

- Breaks the problem into overlapping subproblems
- Uses memorization to keep solutions to subproblems we've already seen.
- works for either DNA or protein sequences, although the substitution matrices used differ
- finds an optimal alignment

Global alignment algorithm: *Needleman-Wunsch.*

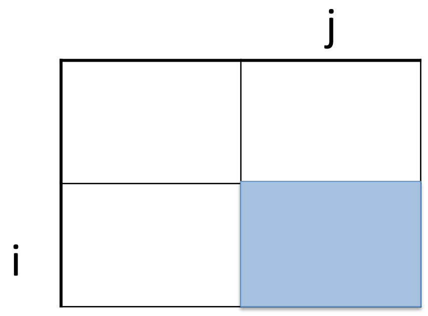
- Align sequence x and y.
- F is the DP matrix; s is the substitution matrix; d is the linear gap penalty.

$$F(0,0) = 0$$

$$F(i,j) = \max\begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

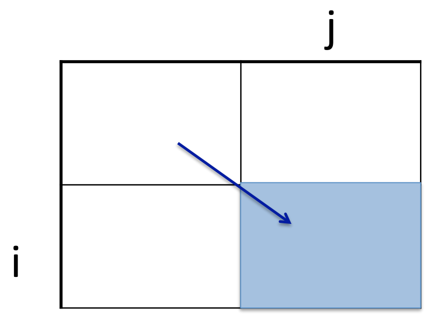
Dynamic Programming Idea

- three possible options; in each we'll choose a different pairing for end of alignment, and add this to the best alignment of previous characters, consider position i in seq 1 and position j in seq2:



Dynamic Programming Idea

- three possible options; in each we'll choose a different pairing for end of alignment, and add this to the best alignment of previous characters, consider position i in seq 1 and position j in seq2:

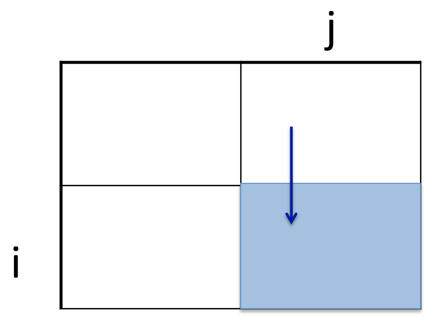


If i aligns to j, then the previous position in each sequence must be i-1 and j-1.

My score must change by s(i,j).

Dynamic Programming Idea

- three possible options; in each we'll choose a different pairing for end of alignment, and add this to the best alignment of previous characters, consider position i in seq 1 and position j in seq2:

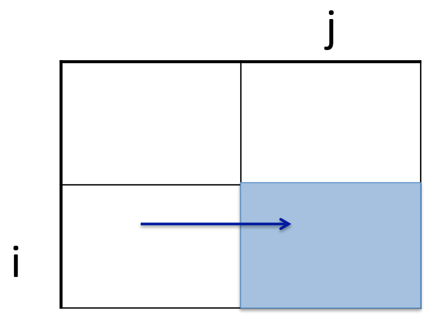


If i aligns to a gap, then the previous position in each sequence must be i-1 and j.

My score must change by the gap penalty.

Dynamic Programming Idea

- three possible options; in each we'll choose a different pairing for end of alignment, and add this to the best alignment of previous characters, consider position i in seq 1 and position j in seq2:

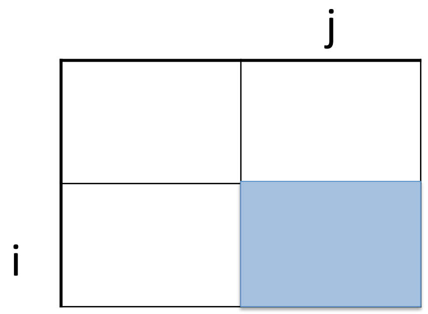


If j aligns to a gap, then the previous position in each sequence must be i and j-1.

My score must change by the gap penalty.

Dynamic Programming Idea

- three possible options; in each we'll choose a different pairing for end of alignment, and add this to the best alignment of previous characters, consider position i in seq 1 and position j in seq2:

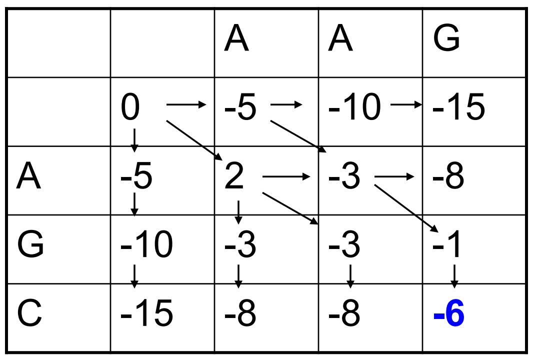


Regardless which choice is best, the value in F(i,j) now contains the best global alignment score of seq1(1…i) to seq2(1…j).

Needleman-Wunsch

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.



		A	A	G
	0	-5	-10	-15
A	-5	2	-3	-8
G	-10	-3	-3	-1
C	-15	-8	-8	**-6**

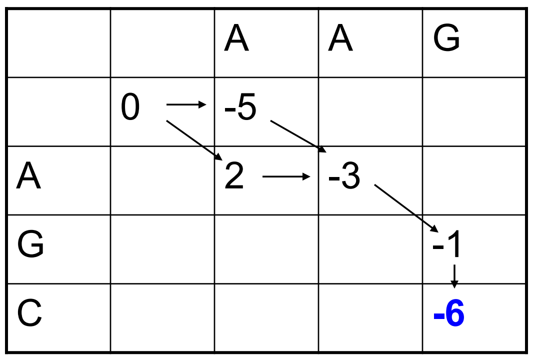
$$\begin{tikzcd}
F(i-1,j-1) \arrow[dr] & & F(i,j-1) \arrow[d] \\
& s(x_i,y_j) \arrow[dr] & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[r] & F(i,j)
\end{tikzcd}$$

Keeping track of which choice is made (by the arrows) enables rapid traceback.

Needleman-Wunsch

Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

- The best score for the entire global alignment is ALWAYS in the lower right corner.
- Following the arrows backwards and recording each choice can recover the optimal alignment.



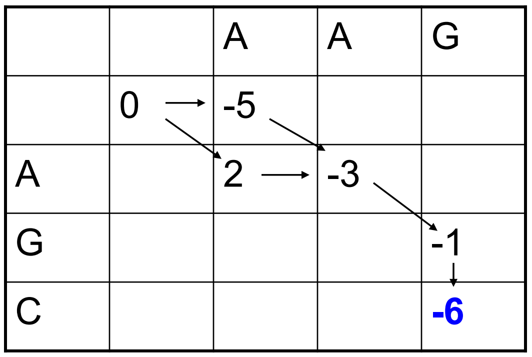
		A	A	G
	0	-5		
A		2	-3	
G				-1
C				**-6**

```
AAG-    AAG-
-AGC    A-GC
```

Needleman-Wunsch

Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

- Each arrow introduces one character at the end of each aligned sequence.
- A horizontal move puts a gap in the left sequence.
- A vertical move puts a gap in the top sequence.
- A diagonal move uses one character from each sequence.



		A	A	G
	0 →	-5		
A		2 →	-3	
G				-1 ↓
C				**-6**

```
AAG-     AAG-
-AGC     A-GC
```

Computational Complexity

- initialization: $O(m)$, $O(n)$ where sequence lengths are m, n
- filling in rest of matrix: $O(mn)$
- traceback: $O(m + n)$
- hence, if sequences have nearly same length, the computational complexity is

$$O(n^2)$$

Notice!

$$O(n^2) \quad << \quad \binom{2n}{n}$$

For example: n = 11
Needleman-Wunch = 121
Total Alignments = 705,432

So Needleman-Wunch is at least 5830 fold faster than iterating through all possible alignments!!

Needleman-Wunch Summary

- Requires:
  - A scoring scheme for matches, mismatches and gaps
  - A recurrence relationship (the iteration steps):

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \end{cases}$$

- There are 2 parts to the algorithm: fill (find best score in lower right) and traceback (recover alignment)

Smith-Waterman: Local alignment

- To be local, should be able to “start over” if the score gets too bad. How?
- Two differences with respect to global alignment:
  - No score is negative.
  - Traceback begins at the highest score in the matrix and continues until you reach 0.

Local Alignment DP Algorithm

- original formulation: Smith & Waterman, *Journal of Molecular Biology*, 1981
- interpretation of array values is somewhat different
  - $F(i, j)$ = score of the best alignment of a prefix of $x[1\ldots i]$ and a prefix of $y[1\ldots j]$

Local alignment DP

- Align sequence x and y.
- F is the DP matrix; s is the substitution matrix; d is the linear gap penalty.

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \\ 0 \end{cases}$$

Local DP in equation form

$$\begin{tikzcd}
 & 0 \arrow[dddr] & \\
F(i-1,j-1) \arrow[dr] & & F(i,j-1) \arrow[d] \\
 & s(x_i, y_j) \arrow[dr] & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[r] & F(i,j)
\end{tikzcd}$$

Smith-Waterman

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	A	G
A				
G				
C				

$$\begin{tikzcd}
F(i-1,j-1) \arrow[dr] & & 0 \arrow[ddr] & F(i,j-1) \arrow[d] \\
& s(x_i,y_j) \arrow[drr] & & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Smith-Waterman

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and AGC.
Use a gap penalty of d=-5.

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

$$\begin{tikzcd}
F(i-1,j-1) \arrow[dr] & & 0 \arrow[ddr] & F(i,j-1) \arrow[d] \\
& s(x_i,y_j) \arrow[drr] & & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Smith-Waterman

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	A	G
	0	0	0	0
A	0	2 -5 -5 0		
G	0			
C	0			

$$\begin{tikzcd}
F(i-1,j-1) \arrow[dr] & & 0 \arrow[ddr] & F(i,j-1) \arrow[d] \\
& s(x_i,y_j) \arrow[drr] & & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Smith-Waterman

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	A	G
	0	0	0	0
A	0	2		
G	0	?		
C	0	?		

$$\begin{tikzcd}
F(i-1,j-1) \arrow[dr] & & 0 \arrow[ddr] & F(i,j-1) \arrow[d] \\
& s(x_i,y_j) \arrow[drr] & & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Smith-Waterman

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	A	G
	0	0	0	0
A	0	2	?	?
G	0	0	?	?
C	0	0	?	?

$$\begin{tikzcd}
F(i-1,j-1) \arrow[dr] & & 0 \arrow[ddr] & F(i,j-1) \arrow[d] \\
& s(x_i,y_j) \arrow[drr] & & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Smith-Waterman

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	A	G
	0	0	0	0
A	0	2	2	0
G	0	0	0	4
C	0	0	0	0

$$\begin{tikzcd}
F(i-1,j-1) \arrow[dr] & & 0 \arrow[ddr] & F(i,j-1) \arrow[d] \\
& s(x_i,y_j) \arrow[drr] & & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Smith-Waterman

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	A	G
	0	0	0	0
A	0	2	2	0
G	0	0	0	4
C	0	0	0	0

AG
AG

$$\begin{tikzcd}
F(i-1,j-1) \arrow[dr] & & 0 \arrow[ddrr] & & F(i,j-1) \arrow[d] \\
& s(x_i,y_j) \arrow[drrr] & & & d \arrow[d] \\
F(i-1,j) \arrow[rr] & & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Smith-Waterman

- So the best score is the MAX value of the table.
  - Not necessarily in lower right corner!
- And you traceback until you reach ZERO, i.e. until the score says “you started over here.”
  - Not necessarily in upper left corner!

Local alignment

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and GAAGGC.
Use a gap penalty of d=-5.

		A	A	G
	0	0	0	0
G	0	0	0	2
A	0	2	2	0
A	0	2	4	0
G	0	0	0	6
G	0	0	0	2
C	0	0	0	0

$$\begin{tikzcd}
 & & 0 \arrow[dddr] & F(i,j-1) \arrow[dd] \\
F(i-1,j-1) \arrow[dr] & & & \\
 & s(x_i,y_j) \arrow[drr] & & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Local alignment

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal local alignment of AAG and GAAGGC.
Use a gap penalty of d=-5.

AAG
AAG

		A	A	G
	0	0	0	0
G	0	0	0	2
	0	2	2	0
A	0	2	4	0
G	0	0	0	6
G	0	0	0	2
C	0	0	0	0

$$\begin{tikzcd}
 & & 0 \arrow[dddr] & F(i,j-1) \arrow[dd] \\
F(i-1,j-1) \arrow[dr] & & & \\
 & s(x_i,y_j) \arrow[drr] & & d \arrow[d] \\
F(i-1,j) \arrow[r] & d \arrow[rr] & & F(i,j)
\end{tikzcd}$$

Local Alignment

- Take note that for Smith-Waterman to work, at least one score in our scoring scheme MUST be positive!!

More On Gap Penalty Functions

- a gap of length *k* is more probable than *k* gaps of length 1
  - a gap may be due to a single mutational event that inserted/deleted a stretch of characters
  - separated gaps are probably due to distinct mutational events
- a linear gap penalty function treats these cases the same
- it is more common to use gap penalty functions involving two terms
  - a penalty *g* associated with <u>opening</u> a gap
  - a smaller penalty *s* for <u>extending</u> the gap

Gap Penalty Functions

- linear

$$w(k) = kd$$

- affine

$$w(k) = \begin{cases} g + sk, & k \geq 1 \\ 0, & k = 0 \end{cases}$$

- Linear gap penalty: every gap receives a score of d.

```
GAAT-C    d=-4
CA-TAC
```

-5 + 10 + -4 + 10 + -4 + 10 = 17

- Affine gap penalty: opening a gap receives a score of d; extending a gap receives a score of e.

```
G--AATC     g=-3
CATA--C     s=-1
```

-5 + -4 + -1 + 10 + -4 + -1 + 10 = 5

General gap penalties

```
AAAGAATTCA          AAAGAATTCA
A-A-A-T-CA    vs.   AAA----TCA
```

In a linear gap scoring model, these alignments have the same score. But the second is likely more biologically plausible.

Linear model: cost of a run of k gaps is k*d

Affine gap model: cost of a run of k gaps is g+k*s

General gap penalties

```
AAAGAATTCA          AAAGAATTCA
A-A-A-T-CA    vs.   AAA----TCA
```

Previous dynamic programming strategy no longer works because the score of the last character depends on the details of the previous alignment (e.g. we have "broken" the independence assumption!:

```
AAAGAA|C          AAAGAAT|C
AAA---|-          AAA----|-
```

Instead we need to "know" how long a final run of gaps is in order to give a score to the last subproblem.

Affine gap model

- Can also be solved by dynamic programming, but it is a more sophisticated model (inherently it's a Markov model!!).