

Practical Bioinformatics

Mark Voorhies

5/2/2011

Exercise: Scoring an ungapped alignment

$$s = \left\{ \begin{array}{l} \text{"A"} : \{ \text{"A"} : 1.0, \text{"T"} : -1.0, \text{"G"} : -1.0, \text{"C"} : -1.0 \}, \\ \text{"T"} : \{ \text{"A"} : -1.0, \text{"T"} : 1.0, \text{"G"} : -1.0, \text{"C"} : -1.0 \}, \\ \text{"G"} : \{ \text{"A"} : -1.0, \text{"T"} : -1.0, \text{"G"} : 1.0, \text{"C"} : -1.0 \}, \\ \text{"C"} : \{ \text{"A"} : -1.0, \text{"T"} : -1.0, \text{"G"} : -1.0, \text{"C"} : 1.0 \} \end{array} \right\}$$

$$S(x; y) = \sum_i^N s(x_i; y_i)$$

- 1 Given two equal length sequences and a scoring matrix, return the alignment score for a full length, ungapped alignment.

Exercise: Scoring an ungapped alignment

$s = \{$
 " A " : { " A " : 1.0 , " T " : -1.0 , " G " : -1.0 , " C " : -1.0 } ,
 " T " : { " A " : -1.0 , " T " : 1.0 , " G " : -1.0 , " C " : -1.0 } ,
 " G " : { " A " : -1.0 , " T " : -1.0 , " G " : 1.0 , " C " : -1.0 } ,
 " C " : { " A " : -1.0 , " T " : -1.0 , " G " : -1.0 , " C " : 1.0 } }
 $\}$

$$S(x; y) = \sum_i^N s(x_i; y_i)$$

- 1 Given two equal length sequences and a scoring matrix, return the alignment score for a full length, ungapped alignment.
- 2 Given two sequences and a scoring matrix, find the offset that yields the best scoring ungapped alignment.

Exercise: Scoring a gapped alignment

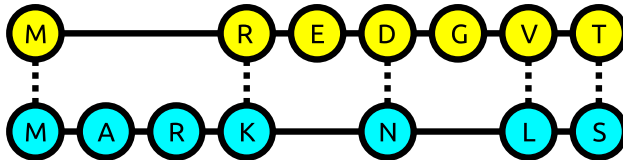
- 1 Given two equal length gapped sequences (where “-” represents a gap) and a scoring matrix, calculate an alignment score with a -1 penalty for each base aligned to a gap.

Exercise: Scoring a gapped alignment

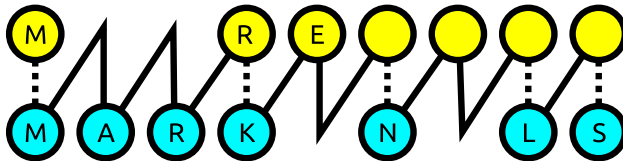
- 1 Given two equal length gapped sequences (where “-” represents a gap) and a scoring matrix, calculate an alignment score with a -1 penalty for each base aligned to a gap.
- 2 Write a new scoring function with separate penalties for opening a zero length gap (e.g., $G = -11$) and extending an open gap by one base (e.g., $E = -1$).

$$S_{gapped}(x; y) = S(x; y) + \sum_i^{gaps} (G + E * len(i))$$

How many ways can we align two sequences?



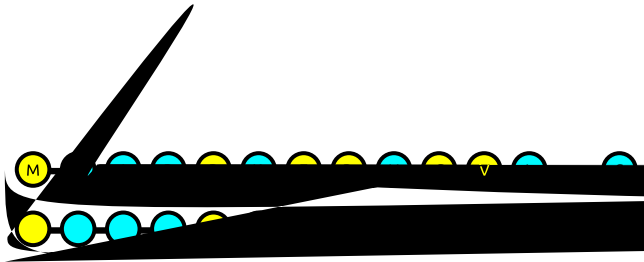
How many ways can we align two sequences?



How many ways can we align two sequences?



How many ways can we align two sequences?



How many ways can we align two sequences?



How many ways can we align two sequences?



Binomial formula:

$$\binom{k}{r} = \frac{k!}{(k-r)!r!} \quad (1)$$

How many ways can we align two sequences?



Binomial formula:

$$\binom{k}{r} = \frac{k!}{(k-r)!r!} \quad (1)$$

$$\binom{2n}{n} = \frac{(2n)!}{n!n!} \quad (2)$$

How many ways can we align two sequences?



Binomial formula:

$$\binom{k}{r} = \frac{k!}{(k-r)!r!} \quad (1)$$

$$\binom{2n}{n} = \frac{(2n)!}{n!n!} \quad (2)$$

Stirling's approximation:

$$x! \approx \sqrt{2\pi} \left(x^{x+\frac{1}{2}}\right) e^{-x} \quad (3)$$

How many ways can we align two sequences?



Binomial formula:

$$\binom{k}{r} = \frac{k!}{(k-r)!r!} \quad (1)$$

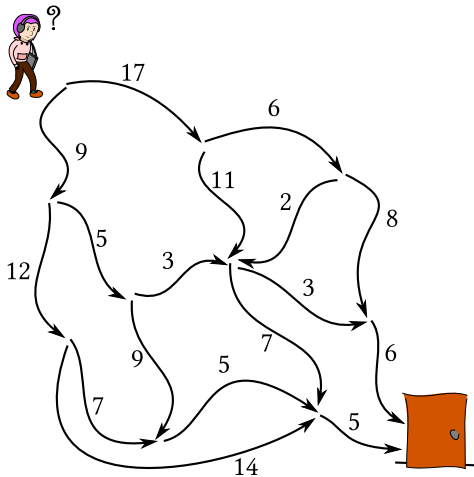
$$\binom{2n}{n} = \frac{(2n)!}{n!n!} \quad (2)$$

Stirling's approximation:

$$x! \approx \sqrt{2\pi} \left(x^{x+\frac{1}{2}}\right) e^{-x} \quad (3)$$

$$\binom{2n}{n} \approx \frac{2^{2n}}{\sqrt{\pi n}} \quad (4)$$

Dynamic Programming



$\min((1, 2, 3, 4, 5)) = 1$

$\max((1, 2, 3, 4, 5)) = 5$

Types of alignments

Global Alignment Each letter of each sequence is aligned to a letter or a gap (e.g., Needleman-Wunsch)

Local Alignment An optimal pair of subsequences is taken from the two sequences and globally aligned (e.g., Smith-Waterman)

Needleman-Wunsch

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

Needleman-Wunsch

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

Needleman-Wunsch

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

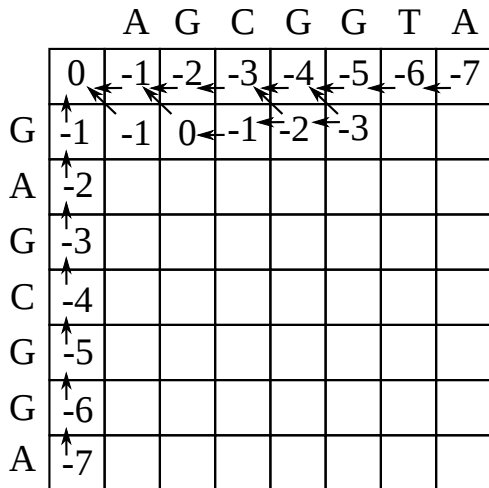
Needleman-Wunsch

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

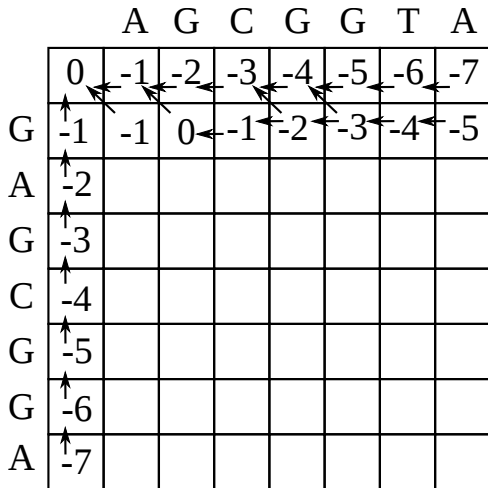
Needleman-Wunsch

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

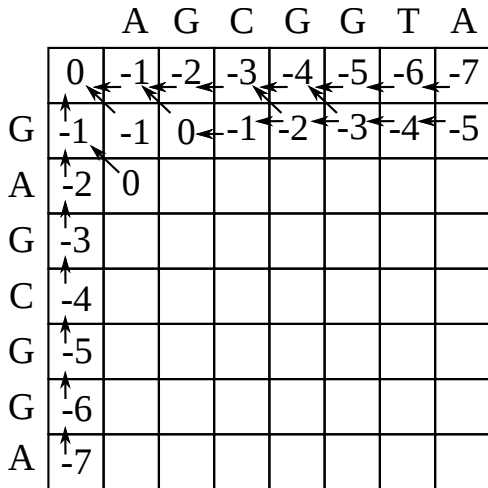
Needleman-Wunsch



Needleman-Wunsch



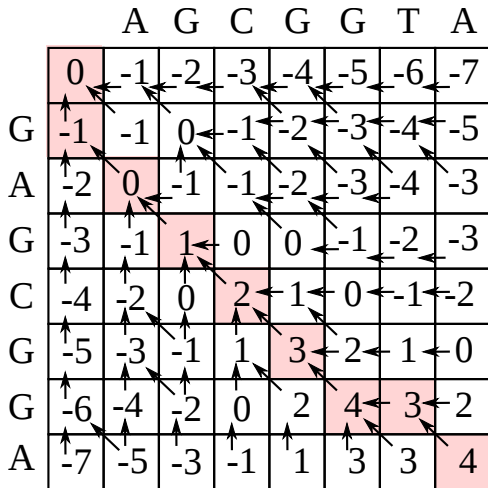
Needleman-Wunsch



Needleman-Wunsch

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

Needleman-Wunsch



dp.nw_fill: Needleman-Wunsch with $g=0$


```
def nw_fill(seq1, seq2, s, e):  
    # m[i][j] = best score for subalignment  
    #   of seq1[:i+1], seq2[:j+1]
```

m = `[[0]]`

p[*i*][*j*] = *list of best paths from*
m[*i*][*j*], or *None* if the alignment
stops here.

p = `[[None]]`

dp.nw_fill: Needleman-Wunsch with $g=0$

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

```
# Fill first row as leading gaps
for j in range(len(seq2)):
    m[-1].append(m[0][j]+e)
    p[-1].append([(0, j)])
```

dp.nw_fill: Needleman-Wunsch with g=0

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

```
for i in range(len(seq1)):
    # First column is leading gaps
    m.append([m[i][0]+e])
    p.append([(i,0)])
    for j in range(len(seq2)):
        # Score for aligning seq1[i] with seq2[j]
        match = m[i][j]+s[seq1[i]][seq2[j]]
        # Score for aligning seq1[i] with a gap
        hgap = m[i+1][j]+e
        # Score for aligning seq2[i] with a gap
        vgap = m[i][j+1]+e

        best = max(match, vgap, hgap)
        m[-1].append(best)
        p[-1].append([])

        if (match == best):
            p[-1][-1].append((i,j))
        if (hgap == best):
            p[-1][-1].append((i+1,j))
        if (vgap == best):
            p[-1][-1].append((i,j+1))
```


Paraphrasing from the BLAST book

- Initialize edges to 0 (*no penalty for starting in the middle of a sequence*)
- The maximum score is never less than 0, and no pointer is recorded unless the score is greater than 0
- The trace-back starts from the highest score in the matrix and ends at a score of 0 (*local, rather than global, alignment*)

Real world dynamic programming implementations

Program	files	algorithms
clustalw	pairalign.c	Myers and Miller pairwise alignment
NCBI blast	blast_gapalign.c	Gapped extension
hmmer2	core_algorithms.c, fast_algorithms.c	Forward, Backward, Viterbi

- Read chapter 3 of the BLAST book (Sequence Alignment).
- Try initializing and filling in a dynamic programming matrix by hand (e.g, try reproducing one of the examples from the BLAST book on paper).
- Try extending the Needleman-Wunsch implementation to handle non-zero gap opening penalties.
- Try implementing Smith-Waterman local alignment