Daniel Ocone

CHAPTER 7

7.1 INTRODUCTION

This chapter discusses the problem of aligning biological sequences to probe for homologous or functional similarities. Given two sequences, for instance AGHGHVWR and AHGIIRN, an alignment is simply a superposition

$$\begin{matrix} A & G & H & G & H & V & W & R & - \\ A & - & H & G & - & I & I & R & N \end{matrix}$$

in which each letter of each sequence is aligned with a letter or gap in the other sequence. When biological sequences are homologous, there will be a natural alignment between them that expresses their biological similarity. Recall from Section 6.7 that two protein sequences are homologous if they have descended from a common, ancestral sequence. We gave a made-up example in Section 6.7.1 of a genealogical tree showing the evolution of the ancestral sequence ARNGVW into the homologous family {GRQGVW, GRGVW, ARWNGVW, ARNGVW}. To capture the homological relationship between, say, the second and third sequences of this family, we wrote the alignment

$$\begin{matrix} G & R & - & - & G & V & W \\ A & R & W & N & G & V & W \end{matrix} \qquad (7.1)$$

The aligned pairs of letters in (7.1) represent amino acids descending from amino acids in the ancestral sequence. Thus, in the first pair $\begin{matrix}G\\A\end{matrix}$, G came from a substitution mutation replacing the A in ARNGVW by G, while the A in $\begin{matrix}G\\A\end{matrix}$ is just the original A of ARNGVW. The pairs $\begin{smallmatrix}R\\R\end{smallmatrix}$, $\begin{smallmatrix}G\\G\end{smallmatrix}$, $\begin{smallmatrix}V\\V\end{smallmatrix}$, $\begin{smallmatrix}W\\W\end{smallmatrix}$ in the alignment all represent descent without modification of the original R, G, V, and W in the ancestral sequence. On the other hand, at some point in the evolution of ARWNGVW, a W was inserted between R and N of the ancestral ARNGVW. There is no amino acid in the sequence GRGVW related to the inserted W and so $\begin{smallmatrix}-\\W\end{smallmatrix}$ is used in the alignment of (7.1). Likewise, the pair $\begin{smallmatrix}-\\N\end{smallmatrix}$ represents the deletion of N in the evolution of GRGVW. In general, in a correct alignment of two homologous sequences, an aligned pair $\begin{smallmatrix}x\\y\end{smallmatrix}$ represents amino acids in each sequence that have evolved from a common amino acid in an ancestral protein sequence; an amino acid x aligned with a gap, as in $\begin{smallmatrix}x\\-\end{smallmatrix}$ or $\begin{smallmatrix}-\\x\end{smallmatrix}$, represents either an insertion or deletion event.

The alignment in (7.1) was derived from full knowledge of the evolutionary history of its two sequences. But of course, biologists are not privy to such full information. Rather, the problem they face is, given several protein sequences, to decide whether the sequences, or even just parts of the sequences, are homologous, and, if so, to infer how they might be aligned. In chapter 6 we discussed how to score <u>aligned</u> sequences to test for homology. But before applying a test it is necessary to align the sequences in the first place. Two sequences may be aligned in a myriad of ways when gaps are allowed; even more possibilities are generated if the aim is not to align the entire sequences but to look for parts of each that may be similar. Even when it is known that two sequences are homologous it is not clear how they should be aligned. For example, the alignment in (7.1) of GRGVW and ARWNGVW looks good by inspection because it aligns many matching pairs, but without knowledge of the genealogy, other alignments are certainly within the realm of possibility.

The strategy for choosing an alignment from among many possibilities for two given strings of letters is first, to choose a scoring method for scoring alignments, and then to find that alignment maximizing the score. Given the scoring method, the challenge is to find a computationally efficient method for finding maximally scoring alignments. That is the question to be addressed in this chapter. We will use a technique called <u>dynamic programming</u> which has an importance in applied mathematics going far beyond the alignment problem.

We shall treat two scoring methods both based on using a substitution matrix $\{S(a,b)\}_{a,b\in\mathcal{A}}$ where $\mathcal{A}$ is the alphabet. In the first method we fix a penalty $-d$ for each gap -- this is equivalent to adding elements $S(a,-)=S(-,a)=-d$ for every $a\in\mathcal{A}$ to the substitution matrix (gaps are never aligned with gaps). This is called the linear gap penalty method because a gap $\begin{array}{cccccc} x_1 & - & - & \cdots & - & x_2 \\ y_1 & y_2 & y_3 & \cdots & y_g & y_{g+1} \end{array}$ of length g adds a penalty $-dg$ to the score (the penalty is linear in g. For example the score of

$$\begin{array}{cccc} A & GG & K & H \\ A & -- & K & P \end{array}$$

would be $S(A,A)-2d+S(K,K)+S(H,P)$.

The second scoring method is called the affine gap penalty method. In this method there is a penalty $-d$ for opening a gap and an additional penalty $-e$ for each new aligned pair $\frac{-}{y}$ or $\frac{x}{-}$ in the same gap. To explain, suppose we see

$$\begin{matrix} - \; - \; \cdots & x_1 & - & - & \cdots & - & x_2 & \cdots \\ & y_1 & y_2 & y_3 & \cdots & y_g & y_{g+1} & \end{matrix}$$

in an alignment. Here we see a gap of length g. The first alignment $\frac{-}{y_2}$ costs $-d$; each following alignment $\frac{-}{y_3}$, $\frac{-}{y_4}$, etc. costs $-e$. Thus this gap contributes

$$-d - (g-1)e \qquad (*)$$

to the total score. Since the expression $(*)$ is an affine function of g, the gap penalty is called affine. Always, d is chosen to be larger than e.

7.2 LINEAR GAP PENALTY ALGORITHMS

7.2.1 The edit graph and representation of alignments.

Let $x_1 \cdots x_n$ and $y_1 \cdots y_m$ be two sequences. We shall represent gapped alignments of these sequences using paths in an <u>edit graph</u>. The edit graph for the two sequences is the directed graph

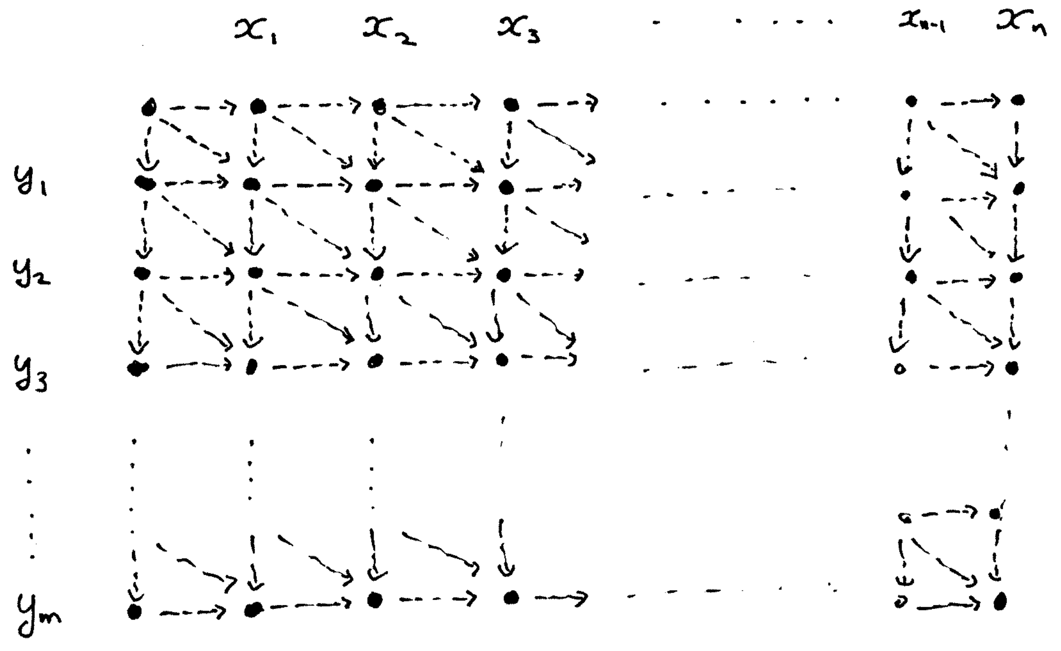


Figure 7.1

This is a graph with vertices in a rectangular array with a column for each x_i and an unlabelled column at the left hand side; similarly, the first row is not labelled, but the remaining are labelled by the letters $y_1, \ldots, y_m$. The only directed edges allowed are of the type shown, either pointing right horizontally one column, down one row, or diagonally down and to the right one column and one row.

Usually we shall suppress writing in all the directed edges -- they are there implicitly.

We think of the first row as row 0 and of the first column as column 0. We label the vertex in column i and row j by (i, j)

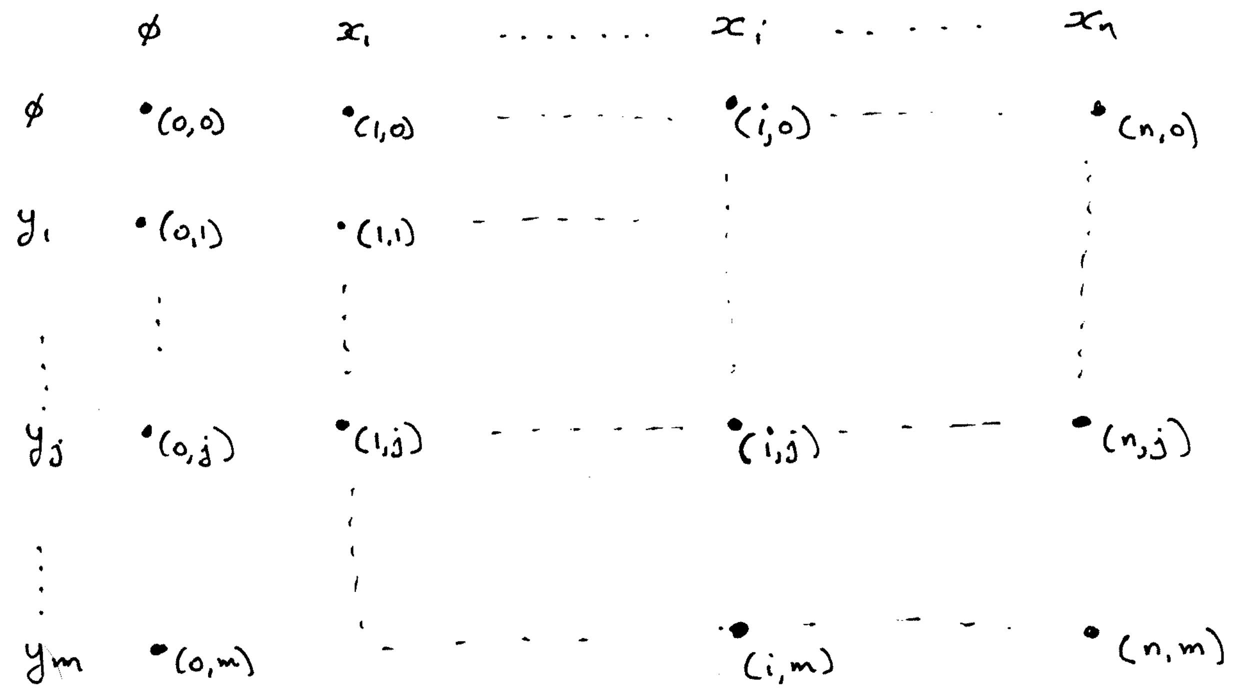


Figure 7.2

The viewpoint from vertex (i,j) with $i \geq 1$, $j \geq 1$ look like (with directed edges into (i,j) shown only):

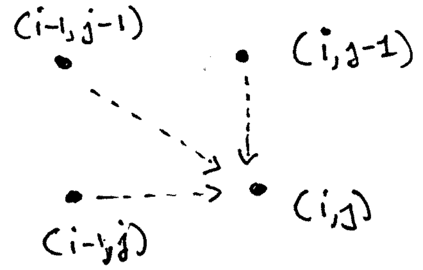


Figure 7.3

A path in the edit graph is a continuous sequence of directed edges, that is each successive edge departs from the vertex attained by the previous edge.

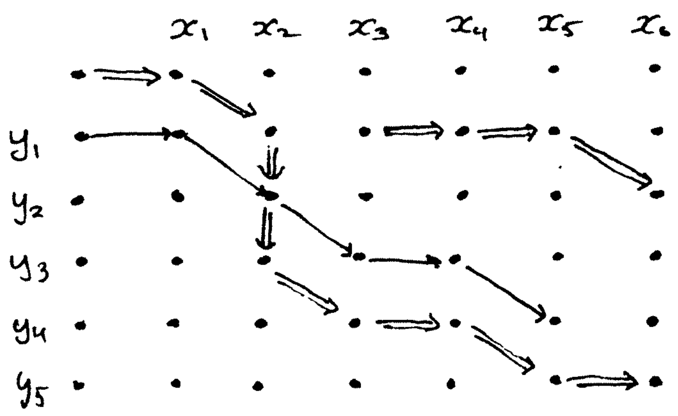

x_1 x_2 x_3 x_4 x_5 x_6
y_1 y_2 y_3 y_4 y_5

Figure 7.4: Some paths.

We shall call a stretch $x_i x_{i+1} \cdots x_j$ of consecutive letters in $x_1 \cdots x_n$ a subsegment of $x_1 \cdots x_n$.

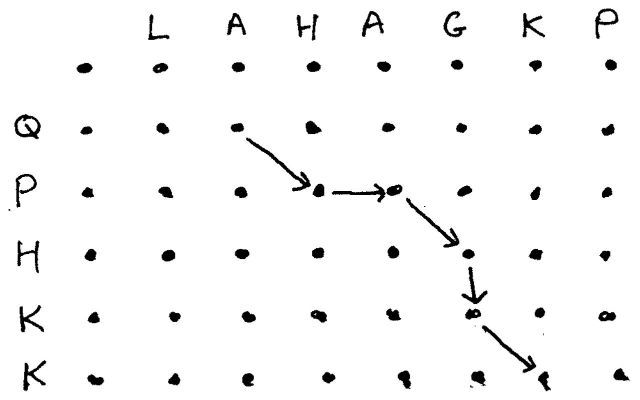
We will now specify a recipe by which any gapped alignment of a subsegment of $x_1 \cdots x_n$ with a subsegment of $y_1 \cdots y_m$ can be represented by a path in the edit graph, and, conversely, any path represents an alignment of subsegments. The recipe is

$$\begin{tikzcd} (i-1, j-1) \arrow[dr] & \\ & (i,j) \end{tikzcd}$$ means "add $\begin{matrix} x_i \\ y_j \end{matrix}$ to the alignment"

$$\begin{tikzcd} (i-1,j) \arrow[r] & (i,j) \end{tikzcd}$$ means "add $\begin{matrix} x_i \\ - \end{matrix}$ to the alignment"

$$\begin{tikzcd} (i,j-1) \arrow[d] \\ (i,j) \end{tikzcd}$$ means "add $\begin{matrix} - \\ y_j \end{matrix}$ to the alignment"

EXAMPLE 7.1



This path represents the alignment $\begin{array}{ccccc} H & A & G & - & K \\ P & - & H & K & K \end{array}$

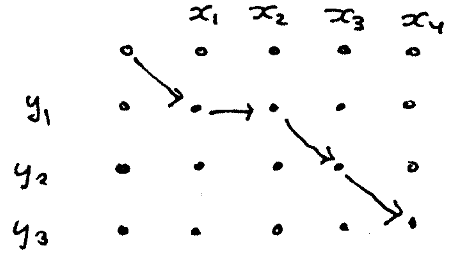
of the subsegment HAGK in LAHAGKP with the subsegment PHKK in QPHKK.

7.2.2 Types of alignments

Let $x_1 \cdots x_n$, $y_1 \cdots y_m$ be two sequences.

A global alignment of these sequences is an alignment of the entire sequence $x_1 \cdots x_n$ with the entire sequence $y_1 \cdots y_m$.

A global alignment corresponds to a path in the edit graph from vertex $(0,0)$ (upper left) to vertex (n,m) (lower right)



global alignment:

$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ y_1 & - & y_2 & y_3 \end{array}$

A <u>local alignment</u> is an alignment of a subsegment of $x_1 \cdots x_n$ with a subsegment of $y_1 \cdots y_m$, where at least one subsegment is proper, that is, is not the entire segment. The alignment of Exercise 7.1 is local. We look for local alignments when we have long strings and want to determine if they contain homologous domains. Any path in the edit graph, except those giving global alignments, represents a local alignment

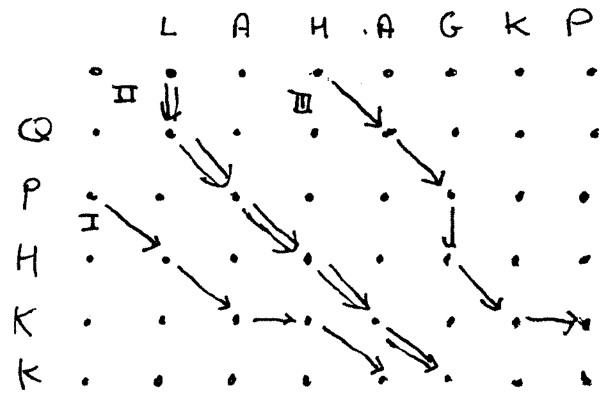
An <u>overlap alignment</u> is a special type of local alignment in which either

a) an initial segment $x_1 \cdots x_i$ of $x_1 \cdots x_n$ is aligned with a terminal segment $y_j \cdots y_m$ of $y_1 \cdots y_m$; or

b) a terminal segment $x_k \cdots x_n$ is aligned with an initial segment $y_1 \cdots y_\ell$; or

c) the entire segment $x_1 \cdots x_n$ aligns with a subsegment of $y_1 \cdots y_m$ or vice-versa.

Example 7.2



Each path in this graph represents an overlap alignment. For example, path I, with the unaligned parts appended in parentheses, is

```
     L A H A (G K P)
(Q P) H K - K
```

Similarly, path II is

```
(L) _ A H A G (K P)
      Q P H K K
```

and path III is

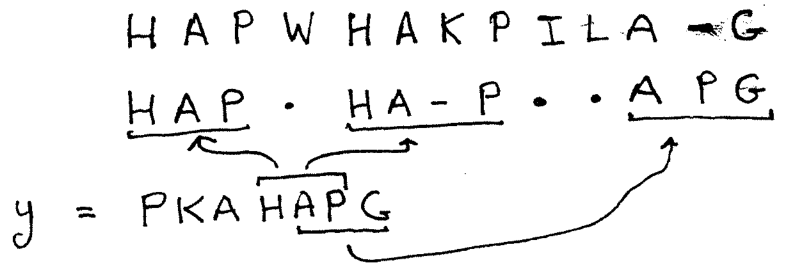
```
(LA) A G - K P
     Q P H K - (K) .
```

$\boxed{}$

Notice that the path of an overlap alignment must start at a vertex in the topmost row or leftmost column and must end in the bottom row or rightmost column.

Suppose we are interested in isolating nonoverlapping subsegments of $x_1 \cdots x_n$ that align to subsegments of $y_1 \cdots y_n$. Such an alignmnent is called a <u>repeat</u> <u>match</u> <u>alignment</u>.

<u>Example 7.3</u> Let $x =$ HAPWHAKPILAG... and $y =$ PKAHAPG Here is the representation of a repeat match alignment:



In the top row the sequence $x = x_1 \cdots x_n$ appears in its entirety in proper order, with possible gaps. In the bottow row are the various subsegments of y that are being aligned with the nonoverlapping subsegments of x. We have indicated where the subsegments of y come from by drawing y below with arrows and bars to indicate what is aligning with what. Dots (not gaps) are used to separate the sub alignments. We represent the alignment again, this time with brackets to enclosed the non-overlapping, aligned subsegments of the x sequence

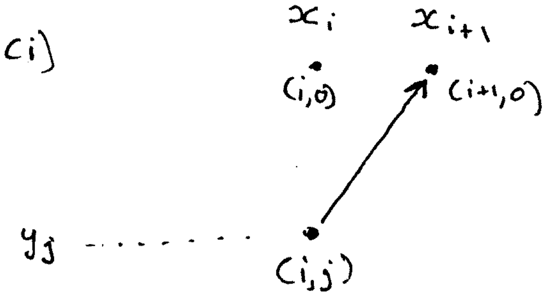
$$\begin{array}{ccccccccccccccc} [H & A & P] & W & [H & A & K & P] & I & L & [A & - & G] \\ H & A & P & \cdot & H & A & - & P & \cdot & \cdot & A & P & G \end{array}$$

▩

In this example the non-overlapping segments are strictly separated (we do not see [HAP][HAKP] for example). Ideally one should allow adjacent ~~sub~~ aligned subsegments without separation, but for technical simplicity we shall impose the condition that the aligned segments in the x sequence be strictly separated.

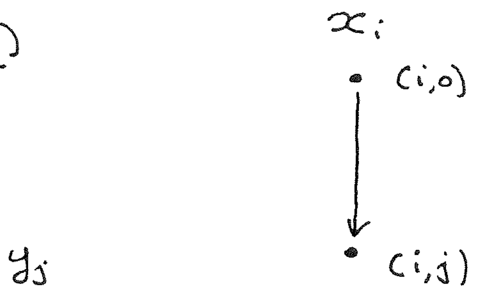
To graph repeat match alignments we add two new types of directed edges to the edit graph:

Type (i)



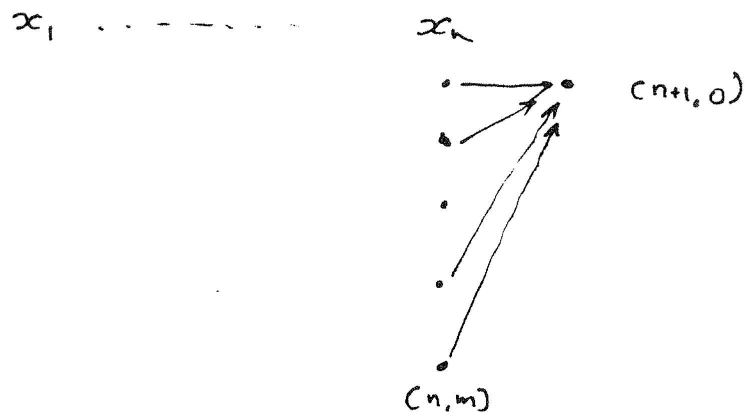
In a path this edge ends a subsegment alignment (a match) by adding $\left[\begin{smallmatrix} x_{i+1} \\ \cdot \end{smallmatrix}\right]$ to the alignment

Type (ii)

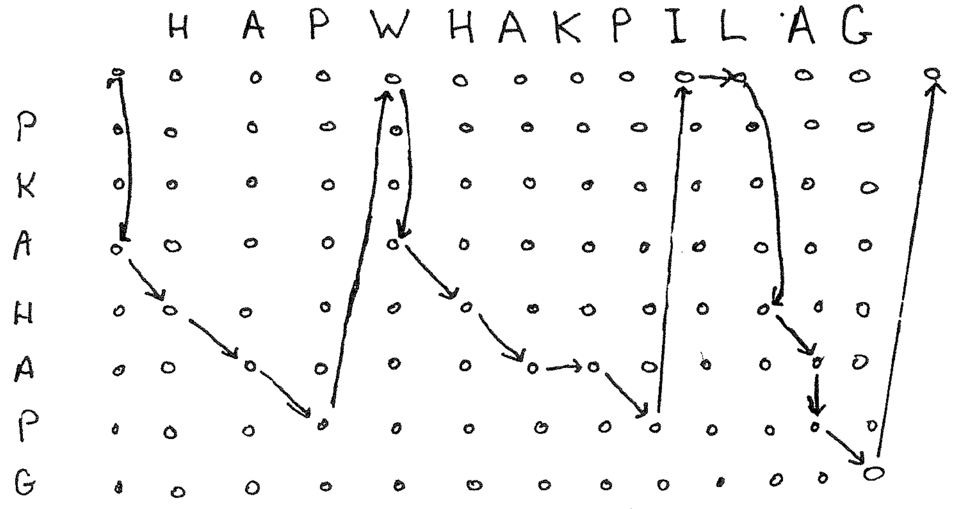


In a path, this edge marks the start of a new subsegment alignment starting at (i,j).

Also, we add vertex to the edit graph in position $(n+1,0)$ and end every path representing a repeat match alignment with an edge to $(n+1,0)$



Here is the path representing the repeat match alignment of Example 7.3.



7.2.3 Scoring paths

We will show how to score paths so that the score of a path is the score of the alignment it represents.

Assume we are given a substitution matrix S and a gap penalty $-d$. The idea is simple: each directed edge is assigned a weight which equals the contribution to the alignment score of the aligned pair or gap represented by the directed edge.

(i) The edge
$$\begin{tikzcd} (i-1,j-1) \arrow[dr] & \\ & (i,j) \end{tikzcd}$$
adds $\begin{matrix} x_i \\ y_j \end{matrix}$ to the alignment, so the weight of this edge is $s(x_i, y_j)$

(ii) The edge
$$\begin{tikzcd} (i,j-1) \arrow[d] \\ (i,j) \end{tikzcd}$$
adds $\begin{matrix} - \\ y_j \end{matrix}$ to the alignment so it has weight $-d$.

(iii) The edge
$$\begin{tikzcd} (i-1,j) \arrow[r] & (i,j) \end{tikzcd}$$
adds $\begin{matrix} x_i \\ - \end{matrix}$ to the alignment so it has weight $-d$.

(So far we have not dealt with the extra, repeat match alignment edges. We do this later.)

To summarize:

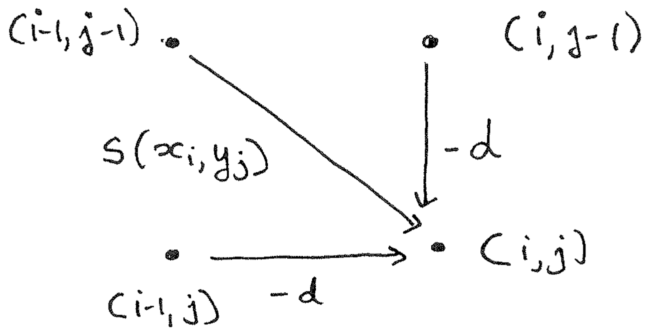
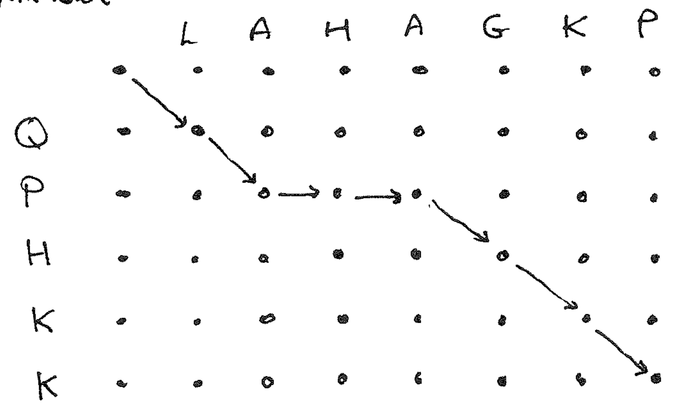


Figure 7.5 Edge weights for scoring.

The score of a path is then defined to be the sum of the weights of its edges.

The path score will then give the score of the alignment it represents using linear gap penalties. This is easy to see by example

EXAMPLE 7.4 Here we show a path representing a global alignment



The weight of the 1st edge is $s(Q,L)$, of the 2nd $s(P,A)$, of the third $-d$, etc. The path score is thus

$$s(Q,L) + s(P,A) - d - d + s(H,G) + s(K,K) + s(K,P)$$

and this is precisely the score of the alignment

```
L A H A G K P
Q P - - H K K
```

7.2.4 Finding optimal (highest scoring) global alignments

One way ~~to find~~ the highest scoring global alignment is simply to list each global alignment, score it, and go down the list to find the highest scoring ones. This is not practical the number of paths in the edit graph representing alignments grows exponentially with n and m. For even moderately sized graphs there are just too many possibilities.

Rather, we take an approach called dynamic programming, working in the edit graph. We define the following function, called the value function, on the points (i,j) $0 \le i \le n$, $0 \le j \le m$, of the edit graph

$$B(0,0) \triangleq 0$$

$$B(i,j) \triangleq \text{maximum score of all paths from } (0,0) \text{ to } (i,j)$$

$$= \text{maximum score of all alignments of } x_1 x_2 \cdots x_i \text{ with } y_1 \cdots y_j$$

We propose to find $\{B(i,j) : 0 \le i \le n, 0 \le j \le m\}$.

At first this might seem crazy. Haven't we just said there are too many paths to count to do this practically? True, but it turns out that B satisfies a simple, recursive equation.

Recall that $B(i,j)$ is the maximal score of all paths from $(0,0)$ to (i,j). Now all paths from $(0,0)$ to (i,j) end with one of the edges in the diagram

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

What is the maximum score from among all paths from $(0,0)$ to (i,j) the end with the diagonal edge from $(i-1,j-1)$ to (i,j)? Clearly it is

$$B(i-1,j-1) + s(x_i,y_j)$$

because $B(i-1,j-1)$ is the maximum score of all paths from $(0,0)$ to $(i-1,j-1)$ and $s(x_i,y_j)$ is the weight added to this score by the final alignment of $\begin{smallmatrix} x_i \\ y_j \end{smallmatrix}$.

Similarly

$B(i,j-1) - d$ is the maximum score of all paths from $(0,0)$ to (i,j) that end in the edge $\begin{tikzcd} (i,j-1) \arrow[d] \\ (i,j) \end{tikzcd}$

$B(i-1,j) - d$ is the maximum score of all paths from $(0,0)$ to (i,j) that end in the edge $(i-1,j) \longrightarrow (i,j)$.

[redacted]

These three cases exhaust all the possibilities from paths from (0,0) to (i,j), so we can draw two conclusions:

(i) $$B(i,j) = \max\{B(i-1,j-1) + s(x_i, y_j),\ B(i,j-1) - d,\ B(i-1,j) - d\}$$ (DPE)

(ii) The last edge on a maximally scoring path from (0,0) to (i,j) points from the vertex that gives the maximum in (i). (For example if $B(i-1,j-1) + s(x_i, y_j)$ is greater than the other two terms, the edge

$$\begin{tikzcd} (i-1,j-1) \arrow[dr] & \\ & (i,j) \end{tikzcd}$$

will be the last edge of a maximally scoring path from (0,0) to (i,j))

(DPE)

Equation (i) is called *the dynamic programming equation* for B. The beauty of it is that it can be solved simply and recursively starting from vertex (0,0) and moving down and rightward in the edit graph. We get a simple algorithm for finding maximal score global alignments:

A) Solve for B at all vertices in the edit graph by using the dynamic programming equation.

B) At each step draw a 'traceback' arrow from (i,j) to the vertex (or vertices) giving the maximum in the dynamic programming equation

c) When A) and B) are done, follow the traceback arrows from the final vertex (n,m) to (0,0). When read in a forward direction this path will have maximal score

Before we give an example we have to add one more part of the dynamic programming equation, because we derived this only for (i,j) with $i \geq 1$, $j \geq 0$.

What if $j = 0$? Then there is only one edge into $(i,0)$, namely

$$\begin{tikzcd} (i-1,0) \arrow[r] & (i,0) \end{tikzcd}$$

Thus

$$B(i,0) = B(i-1,0) - d, \; i \geq 1, \qquad (DPE_2)$$

Similarly

$$B(j,0) = B(j-1,0) - d, \; j \geq 1. \qquad (DPE_3)$$

In both cases the traceback is the reverse of the single edge into $(i,0)$ or $(0,j)$

We work a full example below.

EXAMPLE 7.5. Find the maximal score alignment of $x =$ AAQCCDN and $y =$ ACC using $d=6$ and the substitution matrix

$$S = \begin{array}{c|cccccc} & A & R & N & D & C & Q \\ \hline A & 5 & -2 & -1 & -2 & -1 & -1 \\ R & & 7 & -1 & -2 & -4 & 1 \\ N & & & 7 & 2 & -2 & 0 \\ D & & & & 8 & -4 & 0 \\ C & & & & & 13 & -3 \\ Q & & & & & & 7 \end{array}$$

We shall do this problem by entering the values of $B(i,j)$ directly in the edit graph.

Let's start by finding $B(i,0)$ $1 \le i \le 7$ and $B(0,j)$, $1 \le j \le 5$. This is easy using (DPE_2) and (DPE_3).

First, using (DPE_2)

$B(0,0) = 0$ $B(1,0) = B(0,0) - d = -6$, $B(2,0) = B(1,0) - 6 = -12$, and continuing the obvious trend $B(3,0) = -3(6) = -18$, $B(4,0) = -4(6) = -24$ $B(5,0) = -5(6) = -30$ $B(6,0) = -6(6) = -36$ $B(7,0) = -7(6) = -42$.

Next, using (DPE_3) and similarly reasoning $B(j,0) = -j(6)$ for each j.

We fill those values in the edit graph, adding in the tracebacks in the next figure

$$\begin{tikzcd}
 & & A & A & Q & C & C & D & N \\
 & 0 & -6 \arrow[l, dashed] & -12 \arrow[l, dashed] & -18 \arrow[l, dashed] & -24 \arrow[l, dashed] & -30 \arrow[l, dashed] & -36 \arrow[l] & -42 \arrow[l] \\
A & -6 \arrow[u, dashed] \\
C & -12 \arrow[u, dashed] \\
C & -18 \arrow[u, dashed] \\
Q & -24 \arrow[u, dashed]
\end{tikzcd}$$

<u>First step:</u> Filling in the border values.

Now we are set up to find $B(1,1)$, $B(2,1)$, etc. recursively

$$B(1,1) = \max\{B(0,0) + s(A,A),\ B(1,0)-6,\ B(0,1)-6\}$$
$$= \max\{0+5,\ -6-6,\ -6-6\} = 5$$

The trace back is from $(1,1)$ to $(0,0)$. Thus we can add

$$\begin{tikzcd}
 & & A & A & \\
 & 0 & -6 \arrow[l, dashed] & -12 \arrow[l] & \cdots \\
A & -6 \arrow[u, dashed] & 5 \arrow[ul, dashed] \\
 & \vdots
\end{tikzcd}$$

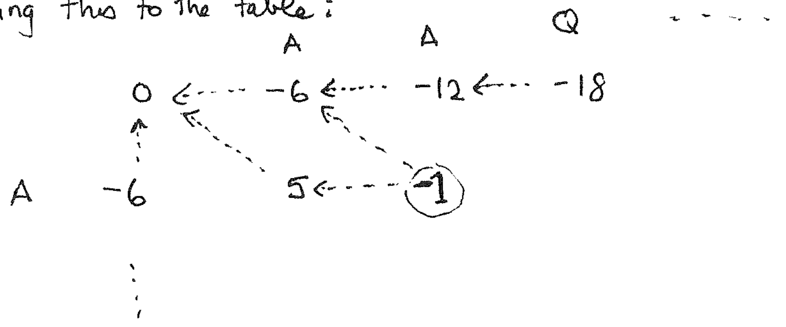
to the table started above.

Next

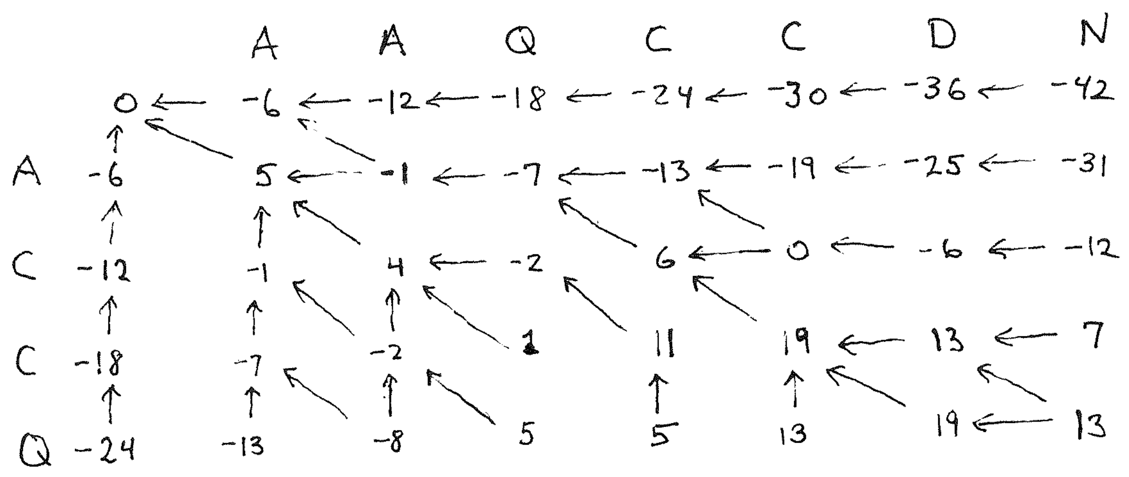
$$B(2,1) = \max\{B(1,0) + s(A,A),\ B(2,0)-6,\ B(1,1)-6\}$$
$$= \max\{-6+5,\ -12-6,\ 5-6\} = -1$$

and the trace back is from $(2,1)$ to $(1,0)$, or from $(2,1)$ to $(1,1)$ (both lead to the max of -1)

Adding this to the table:

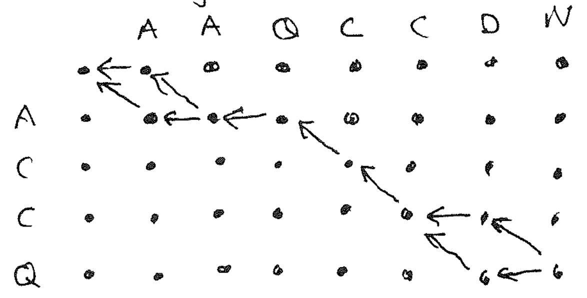


We continue in this manner, moving from row to row; we could also go column by column. The algorithm could be easily implemented on a spread sheet. We give here the final result. The reader should check some of the new entries using the (DPE) (we hope to have made no mistakes!)



		A	A	Q	C	C	D	N
	0	-6	-12	-18	-24	-30	-36	-42
A	-6	5	-1	-7	-13	-19	-25	-31
C	-12	-1	4	-2	6	0	-6	-12
C	-18	-7	-2	1	11	19	13	7
Q	-24	-13	-8	5	5	13	19	13

This is cluttered so we draw the graph only with those tracebacks along paths connecting (0,0) to (7,4)



By taking different branches of the paths going forward we can read off 4 alignments each achieving the maximum score $B(7,4)=13$. These are

```
A A Q C C D N
A - - C C Q -

A A Q C C D N
A - - C C - Q

A A Q C C D N
- A - C C Q -

A A Q C C D N
- A - C C - Q
```