$$S(v) = \begin{cases} 0 & \text{if } v = (i,0) \text{ for } 0 \le i \le |x|, \\ 0 & \text{if } v = (0,j) \text{ for } 0 \le j \le |y|, \\ \max \left\{ \begin{array}{l} 0, \\ S\big((i-1,j-1)\big) + \mathit{score}(x[i], y[j]), \\ \max_{0 \le i' < i} \left\{ S\big((i-i',j)\big) - g(i-i') \right\}, \\ \max_{0 \le j' < j} \left\{ S\big((i,j-j')\big) - g(j-j') \right\} \end{array} \right\} & \text{if } v = (i,j) \text{ for } \left\{ \begin{array}{l} 1 \le i \le |x|, \\ 1 \le j \le |y| \end{array} \right\}, \\ \max\limits_{\substack{0 \le i \le |x| \\ 0 \le j \le |y|}} \left\{ S\big((i,j)\big) \right\} & \text{if } v = v_\bullet. \end{cases}$$

Table 5.4: Smith-Waterman algorithm for local alignment with general gap costs

the framework of general gap costs. We shall see, however, that a quadratic-time algorithm ($O(mn)$ time) exists; the idea is due to Gotoh (1982). We explain it for global alignment; the required modifications for the other alignment types are easy.

Recall that $S\big((i,j)\big)$ is the alignment score for the two prefixes $x[1 \dots i]$ and $y[1 \dots j]$. In general, such a prefix alignment can end with a match/mismatch, a deletion, or an insertion. In the indel case, either the gap is of length $\ell = 1$, in which case its cost is $g(1) = d$, or its length is $\ell > 1$, in which case its cost can recursively be computed as $g(\ell) = g(\ell - 1) + e$.

The main idea is to additionally keep track of (i.e., to tabulate) the *state* of the last alignment column. In order to put this idea into an algorithm, we define the following additional two matrices:

$$V\big((i,j)\big) := \max \left\{ \mathit{score}(A) \;\middle|\; \begin{array}{l} A \text{ is an alignment of the prefixes } x[1 \dots i] \text{ and } y[1 \dots j] \\ \text{that ends with a gap character in } y \end{array} \right\},$$

$$H\big((i,j)\big) := \max \left\{ \mathit{score}(A) \;\middle|\; \begin{array}{l} A \text{ is an alignment of the prefixes } x[1 \dots i] \text{ and } y[1 \dots j] \\ \text{that ends with a gap character in } x \end{array} \right\}.$$

Then

$$S\big((i,j)\big) = \max \left\{ S\big((i-1,j-1)\big) + \mathit{score}(x[i], y[j]),\, V\big((i,j)\big),\, H\big((i,j)\big) \right\},$$

which gives us a method to compute the alignment matrix S, given the matrices V and H. It remains to explain how V and H can be computed efficiently. Consider the case of $V\big((i,j)\big)$: A gap of length ℓ ending at position (i,j) is either a gap of length $\ell = 1$, in which case we can easily compute $V\big((i,j)\big)$ as $V\big((i,j)\big) = S\big((i-1,j)\big) - d$. Or, it is a gap of length $\ell > 1$, in which case it is an extension of the best scoring vertical gap ending at position $(i-1,j)$, $V\big((i,j)\big) = V\big((i-1,j)\big) - e$. Together, we see that for $1 \le i \le m$ and $0 \le j \le n$,

$$V\big((i,j)\big) = \max \left\{ S\big((i-1,j)\big) - d,\; V\big((i-1,j)\big) - e \right\}.$$

Similarly, for horizontal gaps we obtain for $0 \le i \le m$ and $1 \le j \le n$,

$$H\big((i,j)\big) = \max \left\{ S\big((i,j-1)\big) - d,\; H\big((i,j-1)\big) - e \right\}.$$

The border elements are initialized in such a way that they do not contribute to the maximum in the first row or column, for example:

$$V\big((0,j)\big) = H\big((i,0)\big) = -\infty.$$