Algorithm 1 Computation of the first and the next distance columns

function *firstEDtabcolumn*(E, u)
    $E(0) \leftarrow 0$
    for $i \leftarrow 1$ **upto** $|u|$ **do** $E(i) \leftarrow E(i-1) + \delta(u[i] \rightarrow \varepsilon)$
    end for
end function
function *nextEDtabcolumn*(E, E', u, b)
    $E(0) \leftarrow E'(0) + \delta(\varepsilon \rightarrow b)$
    for $i \leftarrow 1$ **upto** $|u|$ **do** $E(i) \leftarrow \min \left\{ \begin{array}{l} E(i-1) + \delta(u[i] \rightarrow \varepsilon) \\ E'(i) + \delta(\varepsilon \rightarrow b) \\ E'(i-1) + \delta(u[i] \rightarrow b) \end{array} \right\}$
    end for
end function
function *evaluateallEDtabcolumns*(E, u, v)
    allocate vector E' of length $|u| + 1$
    $\mathit{firstEDtabcolumn}(E, u)$
    for $j \leftarrow 1$ **upto** $|v|$ **do** $E' \leftarrow E$; $\mathit{nextEDtabcolumn}(E, E', u, v[j])$
    end for
end function

Algorithm 2 The recurrence for table R_{j_0}. The values depend on column E^{j-1}_{col} and E^{j}_{col}. Line 3 handles the case where the insertion edge is minimizing. Line 4 handles the case where the replacement edge is minimizing. Line 5 handles the case where the deletion edge is minimizing. If $j > j_0$, then at least one of the three cases applies, so the *else if* in line 5 could be replaced by *otherwise*. As usual, with a distance value $E^{j}_{\mathsf{col}}(i)$ we can store which of the incoming edges is minimizing, as to remove the dependency on other entries in E^{j-1}_{col} and E^{j}_{col}.

$$R_{j_0}(i,j) = \begin{cases} \mathit{undefined} & \text{if } 0 \le j \le j_0 - 1 \\ i & \text{else if } j = j_0 \\ R_{j_0}(i, j-1) & \text{else if } E^{j}_{\mathsf{col}}(i) = E^{j-1}_{\mathsf{col}}(i) + \delta(\varepsilon \to v[j]) \\ R_{j_0}(i-1, j-1) & \text{else if } E^{j}_{\mathsf{col}}(i) = E^{j-1}_{\mathsf{col}}(i-1) + \delta(u[i] \to v[j]) \\ R_{j_0}(i-1, j) & \text{else if } E^{j}_{\mathsf{col}}(i) = E^{j}_{\mathsf{col}}(i-1) + \delta(u[i] \to \varepsilon) \end{cases}$$

Algorithm 3 A recursive function to compute crosspoints.

Input: i, j, p, q with $1 \le i \le m$, $1 \le j \le n$, $0 \le p \le m - i + 1$, $0 \le q \le n - j + 1$, and uninitialized $(m+1)$-element vectors E and R to store distance and row index columns

Output: table C of crosspoints

1: **function** *evaluatecrosspoints*(C, i, j, p, q)
2: **if** $q \ge 2$ **then**
3: $j_0 \leftarrow \lfloor q/2 \rfloor$
4: $\mathit{evaluateallcolumns}(E, R, j_0, u[i \ldots i+p-1], v[j \ldots j+q-1])$
5: $i_0 \leftarrow R[p]$ ▷ last value of last R_{j_0}-column
6: $C(j - 1 + j_0) \leftarrow i - 1 + i_0$ ▷ i/j refer to strings ⇒ subtract 1
7: *evaluatecrosspoints*(C, i, j, i_0, j_0) ▷ eval. upper/left part
8: *evaluatecrosspoints*$(C, i + i_0, j + j_0, p - i_0, q - j_0)$ ▷ low./right
9: **end if**
10: **end function**
11: create table C of $n+1$ entries and set $C(0) \leftarrow 0$ and $C(n) \leftarrow m$
12: *evaluatecrosspoints*$(C, 1, 1, m, n)$