Algorithm 1 Backtrack a single alignment in a recursive way

```
$S_1$: Array($m$), $S_2$: Array($n$), $M$: Array($m+1$, $n+1$),
function BacktrackRecurse($i$, $j$)
    if $i > 0$ and $j > 0$ then
        $substitute = M[i-1][j-1]$
        $delete = M[i-1][j]$
        $insert = M[i][j-1]$
        $min = \min\{substitute, delete, insert\}$
        if $substitute = min$ then
            $z = $ BacktrackRecurse($S_1$, $S_2$, $M$, $i-1$, $j-1$)
            $z = \begin{pmatrix} S_1[i-1] \\ S_2[j-1] \end{pmatrix} \circ z$
        else if $delete = min$ then
            $z = $ BacktrackRecurse($S_1$, $S_2$, $M$, $i-1$, $j$)
            $z = \begin{pmatrix} S_1[i-1] \\ \varepsilon \end{pmatrix} \circ z$
        else
            $z = $ BacktrackRecurse($S_1$, $S_2$, $M$, $i$, $j-1$)
            $z = \begin{pmatrix} \varepsilon \\ S_2[j-1] \end{pmatrix} \circ z$
        end if
    else if $i > 0$ then
        $z = $ BacktrackRecurse($S_1$, $S_2$, $M$, $i-1$, $j$)
        $z = \begin{pmatrix} S_1[i-1] \\ \varepsilon \end{pmatrix} \circ z$
    else if $j > 0$ then
        $z = $ BacktrackRecurse($S_1$, $S_2$, $M$, $i$, $j-1$)
        $z = \begin{pmatrix} S_1[i-1] \\ \varepsilon \end{pmatrix} \circ z$
    else
        return []
    end if
    return $z$
end function
function Backtrack($S_1$: Array($m$), $S_2$: Array($n$), $M$: Array($m+1$, $n+1$))
    return BacktrackRecurse($S_1$, $S_2$, $M$, $m$, $n$)
end function
```

Algorithm 2 Backtrack all the optimum alignments in a recursive way

```
procedure Recurse($S_1$: Array($m$), $S_2$: Array($n$), $M$: Array($m+1$, $n+1$), $i$, $j$)
    if $i > 0$ and $j > 0$ then
        $substitute = M[i-1][j-1]$
        $delete = M[i-1][j]$
        $insert = M[i][j-1]$
        $min = \min\{substitute, delete, insert\}$
        if $substitute = min$ then
            $value = \begin{pmatrix} S_1[i-1] \\ S_2[j-1] \end{pmatrix}$
            $z' = value \circ z$
            BacktrackRecurse($S_1$, $S_2$, $M$, $i-1$, $j-1$, $z'$)
        end if
        if $delete = min$ then
            $value = \begin{pmatrix} S_1[i-1] \\ \varepsilon \end{pmatrix}$
            $z' = value \circ z$
            BacktrackRecurse($S_1$, $S_2$, $M$, $i-1$, $j$, $z'$)
        end if
        if $insert = min$ then
            $value = \begin{pmatrix} \varepsilon \\ S_2[j-1] \end{pmatrix}$
            $z' = value \circ z$
            BacktrackRecurse($S_1$, $S_2$, $M$, $i$, $j-1$, $z'$)
        end if
    else if $i > 0$ then
        $value = \begin{pmatrix} S_1[i-1] \\ \varepsilon \end{pmatrix}$
        $z' = value \circ z$
        BacktrackRecurse($S_1$, $S_2$, $M$, $i-1$, $j$, $z'$)
    else if $j > 0$ then
        $value = \begin{pmatrix} \varepsilon \\ S_2[j-1] \end{pmatrix}$
        $z' = value \circ z$
        BacktrackRecurse($S_1$, $S_2$, $M$, $i$, $j-1$, $z'$)
    end if
    print($z$)
end procedure
procedure Backtrack($S_1$: Array($m$), $S_2$: Array($n$), $M$: Array($m+1$, $n+1$))
    return BacktrackRecurse($S_1$, $S_2$, $M$, $m$, $n$, [])
end procedure
```