Algorithm 1 Extraction of the relevant substrings

Input: suffix array SA, lcp-array LCP, T, C'', minf, maxf
Output: List r containing n tuples (p, a, b, lcp)
S is a stack containing values l
a, b are arrays of length n
push(0) on S
Let $LCP[n+1] = 0$
for i=2 to n+1 **do**
  push(i-1) on S
  $h \leftarrow n - SA[i-1] + 1$
  l $\leftarrow$ i-1
  a[i-1] $\leftarrow \infty$, b[i-1] $\leftarrow 0$
  while h $>$ LCP[i] **do**
    l $\leftarrow$ pop(S)
    if minf $\leq i - l - (C''[i-1] - C''[l]) \leq$ maxf **then**
      if a[l] $= \infty$ **then**
        b[l] $\leftarrow$ h
      end if
      a[l] $\leftarrow$ max{LCP[l], LCP[i]}+1
    end if
    $h \leftarrow$ LCP[l]
  end while
  if h $<$ LCP[i] **then**
    push(l) on S
  end if
end for
{now C'' is not needed anymore and can be reused for the next values}
$C''[n] \leftarrow 1$
for $i = n-1$ to 1 **do**
  if $T_i = \#$ **then**
    $C''[i] \leftarrow 1$
  else
    $C''[i] \leftarrow 1 + C''[i+1]$
  end if
end for
for i = 1 to n **do**
  $r[i] \leftarrow (SA[i], a[i], \min\{b[i], C''[SA[i]] - 1\}, LCP[i])$
end for

Algorithm 2 Intersect two lists of solution tuples – slink_merge

Input: Lists L_1 and L_2 of tuples (p, a, b, lcp), ms, c
Output: List L_{out} with the tuples representing the intersection of L_1 and L_2
$p_2 \leftarrow n_2$, $a_{cur} \leftarrow \infty$, $a_{prev} \leftarrow \infty$
$b_{cur} \leftarrow 0$, $b_{prev}, \leftarrow 0$, $lcp_{cur} \leftarrow 0$, $lcp_{prev} \leftarrow 0$
for $p_1 = n_1 + 1$ to 1 **do**
  $a_{cur} \leftarrow a_{prev}$, $b_{cur} \leftarrow b_{prev}$, $lcp_{cur} \leftarrow lcp_{prev}$
  if $p_1 \leq n_1$ **then**
    $lcp_{prev} \leftarrow \min\{lcp_{prev}, L_1[p_1].lcp\}$
    if $L_1[p_1].lcp < a_{cur}$ **then**
      $a_{prev} \leftarrow \infty$, $b_{prev} \leftarrow 0$
    else
      $a_{prev} \leftarrow a_{cur}$, $b_{prev} \leftarrow \min\{b_{cur}, L_1[p_1].lcp\}$
    end if
  end if
  for $i = 1$ to $c[p_1]$ **do**
    if $ms[L_2[p_2].p] < 0$ **then**
      $lcp_1 \leftarrow -ms[L_2[p_2].p]$, $lcp_2 \leftarrow L_1[p_1].lcp$
    else
      $lcp_1 \leftarrow L_1[p_1].lcp$, $lcp_2 \leftarrow ms[L_2[p_2].p]$
    end if
    $lcp_{prev} \leftarrow \max\{lcp_{prev}, lcp_1\}$
    $lcp_{cur} \leftarrow \max\{lcp_{cur}, lcp_2\}$
    if $L_2[p_2].a \leq L_2[p_2].b$ **then**
      if $lcp_1 \geq L_2[p_2].a$ **then**
        $a_{prev} \leftarrow \min\{a_{prev}, L_2[p_2].a\}$
        $b_{prev} \leftarrow \max\{b_{prev}, \min\{lcp_1, L_2[p_2].b\}\}$
      end if
      if $lcp_2 \geq L_2[p_2].a$ **then**
        $a_{cur} \leftarrow \min\{a_{cur}, L_2[p_2].a\}$
        $b_{cur} \leftarrow \max\{b_{cur}, \min\{lcp_2, L_2[p_2].b\}\}$
      end if
    end if
    $p_2 \leftarrow p_2 - 1$
  end for
  if $p_1 \leq n_1$ **then**
    $L_{out}[p_1].a \leftarrow \max\{a_{cur}, L_1[p_1].a\}$
    $L_{out}[p_1].b \leftarrow \min\{b_{cur}, L_1[p_1].b\}$
    if $minf^{\mathcal{D}_2} = 0$ **then**
      $L_{out}[p_1].b \leftarrow L_1[p_1].b$
      if $a_{cur} > b_{cur}$ **then**
        $L_{out}[p_1].a \leftarrow \max\{lcp_{cur} + 1, L_1[p_1].a\}$
      end if
    end if
  end if
end for