Algorithm 1: The Algorithm Based on Corollary 1.

Function CheckByFP(x, sa, lcp, n)

$ST_1 := [(sa[i], i, null) | i \in [0, n)]$

$ST_2 := [(sa[i] + lcp[i+1], i, null, null) | i \in [0, n-1)]$

$ST_3 := [(sa[i] + lcp[i], i, null, null) | i \in [1, n)]$

sort the tuples in ST_1, ST_2 and ST_3 by the 1st components, respectively;

$fp := 0$

for $i \in [0, n]$ **do**

  if ST_1.notEmpty() and ST_1.top().$1st = i$ **then**

    $e := ST_1$.top(), ST_1.pop(), $e.3rd := fp$, ST'_1.push(e)

  end

  else

    return false // condition 1 is violated

  end

  while ST_2.notEmpty() and ST_2.top().$1st = i$ **do**

    $e := ST_2$.top(), ST_2.pop(), $e.3rd := fp$, $e.4th := x[i]$, ST'_2.push(e)

  end

  while ST_3.notEmpty() and ST_3.top().$1st = i$ **do**

    $e := ST_3$.top(), ST_3.pop(), $e.3rd := fp$, $e.4th := x[i]$, ST'_3.push(e)

  end

  $fp := fp \cdot \delta + x[i] \mod P$ // $x[n]$ is the virtual character

end

sort the tuples in ST'_1, ST'_2 and ST'_3 by the 2nd component, respectively;

for $i \in [1, n)$ **do**

  $fp_1 := ST'_1$.top().$3rd$, ST'_1.pop(), $fp_2 := ST'_2$.top().$3rd$, $ch_1 := ST'_2$.top().$4th$, ST'_2.pop()

  $\hat{fp_1} = fp_2 - fp_1 \cdot \delta^{lcp[i]} \mod P$

  $fp_1 := ST'_1$.top().$3rd$, $fp_3 := ST'_3$.top().$3rd$, $ch_2 := ST'_3$.top().$4th$, ST'_3.pop()

  $\hat{fp_2} = fp_3 - fp_1 \cdot \delta^{lcp[i]} \mod P$

  if $\hat{fp_1} \neq \hat{fp_2}$ or $ch_1 \geq ch_2$ **then**

    return false // condition 2 or 3 is violated

  end

end

return true