String Processing Algorithms 2015 - Week 3 Exercises

Rodion Efremov

November 8, 2015

Exercise 1

Describe how to modify the LSD radix sort algorithm to handle strings of varying length. The time complexity should be the one given in Theorem 1.27.

Solution

The time complexity mentioned in the Theorem 1.27 is $\mathcal{O}(||\mathcal{R}|| + m\sigma)$. All we need to do is to modify the COUNTING-SORT procedure:

Now, the desired LSD radix sort for variable-length strings is

Exercise 2

Use the lcp comparison technique to modify the standard insertion sort algorithm so that it sorts strings in $\mathcal{O}(\Sigma LCP(\mathcal{R}) + n^2)$ time.

Solution

Why do we set LCP to zero?

Consider the following:

$aaaba$	0
$abaaa$	1
aaa	1
$baaa$	0
$aabba$	0
$abaaa$	1

Algorithm 1: COUNTING-SORT($\mathcal{R} = \{S_1, S_2, \ldots, S_n\}, \ell$)

```
for i = 0 to σ − 1 do
    C[i] = 0
s = 0
for i = 1 to n do
    if |S_i| < ℓ then
        s = s + 1
    else
        C[S_i[ℓ]] = C[S_i[ℓ]] + 1
sum = s
for i = 0 to σ − 1 do
    tmp = C[i]
    C[i] = sum
    sum = sum + tmp
p = 0
for i = 1 to n do
    if |S_i| < ℓ then
        J[p] = S_i
        p = p + 1
    else
        J[C[S_i[ℓ]]] = S_i
        C[S_i[ℓ]] = C[S_i[ℓ]] + 1
R = J
```

Algorithm 2: LSDRadixSort($\mathcal{R} = \{S_1, S_2, \ldots, S_n\}$)

```
m = max_{i=1,2,...,n}{|S_i|}
for ℓ = m to 1 do
    COUNTING-SORT(R, ℓ)
```

Algorithm 3: INSERTIONSORT($\mathcal{R}, \mathrm{LCP}_{\mathcal{R}}$)

```
for i = 2 to n do
    s = S_i
    j = i − 1
    if LCP_R[i − 1] = 0 then
        LCP_R[i] = 0
    while j > 0 and LCPCOMPARE(S_j, s, LCP_R[i]) > 0 do
        S_{j+1} = S_j
        j = j − 1
    S_{j+1} = s
```