Linear computation of the Lyndon array and the suffix array

Richard Groult Nicolas Guiomard-Kagan Thierry Lecroq
Arnaud Lefebvre Pierre Morisse Élise Prieur-Gaston
Tatiana Rocher Mikal Salson

Abstract

1 Introduction

2 Background and basic string definitions

A *string* is a sequence of zero or more symbols from an alphabet Σ; the string with zero symbols is denoted by ε. The set of all strings over the alphabet Σ is denoted by Σ^*. We consider an alphabet of size s; for $1 \leq i \leq s$, $\sigma[i]$ denotes the i-th symbol of Σ. A string x of length n is represented by $x[1..n]$, where $x[i] \in \Sigma$ for $1 \leq i \leq n$. A string u is a *prefix* of x if $x = uw$ for $w \in \Sigma^*$. Similarly, u is a *suffix* of x if $x = wu$ for $w \in \Sigma^*$. A string u is a *border* of x if u is a prefix and a suffix of x and $u \neq x$.

3 Baier's algorithm

4 Our algorithm

```
PHASE1(lastgroup, P)
 1  iter ← 1
 2  Prev[0] ← −1
 3  for G = lastgroup; G ≠ ∅; G = G.next do
 4     rank ← G.rank
 5     for Elt = G.head; Elt ≠ ∅; Elt = Elt.next do
 6        i ← Elt.pos
 7        j ← i − 1
 8        while j ≠ −1 and P[j].group.rank ≤ rank do
 9           j ← Prev[j]
10        Prev[i] ← j
11        if j = −1 then
12           groupj ← P[j].group
13           length ← groupj.length + G.length
14           nextgroup ← groupj.next
15           if nextgroup.iter = iter and nextgroup.length = length then
16              MOVE(P[j], groupj, nextgroup)
17           else if groupj.size = 1 then
18              groupj.length ← length
19           else newgroup ← NEWGROUPBEFORE(nextgroup, iter, length)
20              MOVEDOWN(P[j], groupj, newgroup)
21     iter ← iter + 1
```

```
PHASE2(firstgroup, P, n)
 1  if P[n].group.size > 1 then
 2     newgroup ← NEWGROUPBEFORE(P[n].group)
 3     MOVEUP(P[n], P[n].group, newgroup)
 4     SA[newgroup.rank] ← n
 5  for G = firstgroup; G ≠ ∅; G = G.next do
 6     for Elt = G.head; Elt ≠ ∅; Elt = Elt.next do
 7        i ← Elt.pos
 8        if P[i].group.size = 1 then
 9           SA[P[i].group.rank] ← i
10        j ← i − 1
11        if P[j].group.size > 1 then
12           newgroup ← NEWGROUPBEFORE(P[j].group)
13           MOVEUP(P[j], P[j].group, newgroup)
14           SA[newgroup.rank] ← j
15  return SA
```

Example

```
0 1 2 3 4 5 6 7 8 9 10
m i s s i s s i p p i

iter = 0, rank = 0, size = 4, length = 1, list = (1,4,7,10), context = i
iter = 0, rank = 4, size = 1, length = 1, list = (0), context = m
iter = 0, rank = 5, size = 2, length = 1, list = (8,9), context = p
iter = 0, rank = 7, size = 4, length = 1, list = (2,3,5,6), context = s

Prev[2] = 1

iter = 0, rank = 0, size = 3, length = 1, list = (4,7,10), context = i
iter = 1, rank = 3, size = 1, length = 2, list = (1), context = is
iter = 0, rank = 4, size = 1, length = 1, list = (0), context = m
iter = 0, rank = 5, size = 2, length = 1, list = (8,9), context = p
iter = 0, rank = 7, size = 4, length = 1, list = (2,3,5,6), context = s

Prev[3] = Prev[2] = 1

iter = 0, rank = 0, size = 3, length = 1, list = (4,7,10), context = i
iter = 1, rank = 3, size = 1, length = 3, list = (1), context = iss
iter = 0, rank = 4, size = 1, length = 1, list = (0), context = m
iter = 0, rank = 5, size = 2, length = 1, list = (8,9), context = p
iter = 0, rank = 7, size = 4, length = 1, list = (2,3,5,6), context = s

Prev[5] = 4

iter = 0, rank = 0, size = 2, length = 1, list = (7,10), context = i
iter = 1, rank = 2, size = 1, length = 2, list = (4), context = is
iter = 1, rank = 3, size = 1, length = 3, list = (1), context = iss
iter = 0, rank = 4, size = 1, length = 1, list = (0), context = m
iter = 0, rank = 5, size = 2, length = 1, list = (8,9), context = p
iter = 0, rank = 7, size = 4, length = 1, list = (2,3,5,6), context = s

Prev[6] = Prev[5] = 4

iter = 0, rank = 0, size = 2, length = 1, list = (7,10), context = i
iter = 1, rank = 3, size = 2, length = 3, list = (1,4), context = iss
iter = 0, rank = 4, size = 1, length = 1, list = (0), context = m
iter = 0, rank = 5, size = 2, length = 1, list = (8,9), context = p
iter = 0, rank = 7, size = 4, length = 1, list = (2,3,5,6), context = s

Prev[8] = 7

iter = 0, rank = 0, size = 1, length = 1, list = (10), context = i
iter = 2, rank = 1, size = 1, length = 2, list = (7), context = ip
iter = 1, rank = 3, size = 2, length = 3, list = (1,4), context = iss
```

```
iter = 0, rank = 4, size = 1, length = 1, list = (0), context = m
iter = 0, rank = 5, size = 2, length = 1, list = (8,9), context = p
iter = 0, rank = 7, size = 4, length = 1, list = (2,3,5,6), context = s

Prev[9] = Prev[8] = 7

iter = 0, rank = 0, size = 1, length = 1, list = (10), context = i
iter = 2, rank = 1, size = 1, length = 3, list = (7), context = ipp
iter = 1, rank = 3, size = 2, length = 3, list = (1,4), context = iss
iter = 0, rank = 4, size = 1, length = 1, list = (0), context = m
iter = 0, rank = 5, size = 2, length = 1, list = (8,9), context = p
iter = 0, rank = 7, size = 4, length = 1, list = (2,3,5,6), context = s

Prev[0] = -1

Prev[1] = Prev[0] = -1

Prev[4] = Prev[3] = Prev[1] = -1

Prev[7] = Prev[6] = Prev[4] = -1

Prev[10] = Prev[9] = Prev[7] = -1
```

5 Correctness

6 Experiments

7 Conclusion