Lyndon tree construction

Maxime

September 22, 2014

Abstract
Use of lcp’s to improve the right-to-left Lyndon tree construction.

1 Lyndon tree computation

LYNDON_TREE(y Lyndon word of length n)
1  v ← y[n − 1]
2  T(v) ← (y[n − 1])
3  for i ← n − 2 downto 0 do
4    u ← y[i]
5    T(u) ← (y[i])
6      while u < v do
7        u ← uv
8        T(u) ← (new node, T(u), T(v))
9        v ← next phrase, ϵ if none
10  return T(y)

If the comparison u < v is done by mere letter comparisons, the algorithm runs in quadratic time on \( y = a^k b a^k c \) (each factor \( a^i b \) is compared with the prefix \( a^{i+1} \) of \( a^k c \) or with \( a^k c \) itself).

It runs in linear time if the test \( u < v \) is done in constant time because each execution of instructions 7-9 decreases the number of Lyndon phrases, which goes from \( n \) to 1.

It is realised with the suffix array of \( y \), RMQ, etc.

In the comparison model, it is also realised if we know lcp(\( u, v \)) because the comparison resumes to a letter comparison.