Factorisation of Lyndon words

Theorem (Lyndon)

A word $w \in \mathcal{A}^+$ is a Lyndon word if and only if $w \in \mathcal{A}$ or there exists two Lyndon words u and v such that $w = uv$ and $u \prec v$.

In general, this factorisation is not unique since, for example, $w = aabac$ has two such factorisations:

$$w = (a)(abac) \quad \text{and} \quad w = (aab)(ac).$$

However, there is a unique factorisation of a given Lyndon word as a product uv of two Lyndon words u, v with $u \prec v$, called the standard factorisation.

Theorem (Chen-Fox-Lyndon 1958)

If $w = uv$ is a Lyndon word with v its lexicographically smallest proper suffix, then u and v are also Lyndon words and $u \prec v$.

So the standard factorisation of a Lyndon word $w = uv$ is obtained by choosing v to be the lexicographically least proper suffix of w, which also happens to be the longest proper suffix of w that is Lyndon.

Example: $w = aabac$ has standard factorisation $w = (a)(abac)$.

An Application in Algebra

- In algebraic settings, Lyndon words give rise to commutators using standard factorisation iteratively.
- For example, the Lyndon word $aababb$ with standard factorisation $(a)(ababb)$ gives rise to the commutator $[a,[[a,b],[[a,b],b]]]$.
- These commutators can be viewed either as elements of the free group with $[x,y] = xyx^{-1}y^{-1}$, or as elements of the free Lie algebra with $[x,y] = xy - yx$.

  In either case, Lyndon words give rise to a basis of some algebra.
- For specific details, see [C. Reutenauer, *Free Lie Algebras*, 1993] and [M. Lothaire, *Combinatorics on words*, 1983].

Generation of Lyndon Words

Duval's Algorithm: Generates the Lyndon words over $\mathcal{A}$ of length at most n ($n \geq 2$).

If w is one of the words in the list of Lyndon words up to length n (not equal to $\max(\mathcal{A})$), then the next Lyndon word after w can be found by the following steps:

1. Repeat the letters from w to form a new word x of length exactly n, where the i-th letter of x is the same as the letter at position $i \pmod{|w|}$ in w.
2. If the last letter of x is $\max(\mathcal{A})$ for the given order on $\mathcal{A}$, remove it, producing a shorter word, and take this to be the new x.
3. If the last letter of x is $\max(\mathcal{A})$, then repeat Step 2. Otherwise, replace the last letter of x by its successor in the sorted ordering of the alphabet $\mathcal{A}$.

Note: Since, in general, the factorisation of a Lyndon word as a product uv of two Lyndon words u, v with $u \prec v$ is not unique, Duval's algorithm may produce the same Lyndon word more than once.

Duval's Algorithm in Action

Let's use the algorithm to generate all the Lyndon words up to length 4 over the alphabet $\{a, b\}$ with $a \prec b$.

We begin with the Lyndon words of length 1: a and b. List: $\mathcal{L} = \{a, b, \ldots\}$

Lyndon words of length at most 2:

$$\boxed{a} \longrightarrow aa \longrightarrow \boxed{ab}$$
$$\boxed{b} \longrightarrow bb \longrightarrow b \longrightarrow \not\varepsilon$$

Updated List: $\mathcal{L} = \{a, b, ab, \ldots\}$

Lyndon words of length at most 3:

$$\boxed{a} \longrightarrow aaa \longrightarrow \boxed{aab}$$
$$\boxed{ab} \longrightarrow aba \longrightarrow \boxed{abb}$$

Updated List: $\mathcal{L} = \{a, b, ab, aab, abb, \ldots\}$

Duval's Algorithm in Action ...

Lyndon words of length at most 4:

$$\boxed{a} \longrightarrow aaaa \longrightarrow \boxed{aaab}$$

$$\boxed{ab} \longrightarrow abab \longrightarrow \cancel{abb} \text{ (repeat)}$$

$$\boxed{aab} \longrightarrow aaba \longrightarrow \boxed{aabb}$$

$$\boxed{abb} \longrightarrow abba \longrightarrow \boxed{abbb}$$

Updated List: $\mathcal{L} = \{a, b, ab, aab, abb, aaab, aabb, abbb, \ldots\}$

And on it goes ...