

Comment engendrer la liste des facteurs de longueur donnée d'un mot substitutif

<http://lma.homelinux.org/~twiki/bin/view/Substitutions/FactorsOfAGivenLength>

1 Cadre

On se donne un alphabet fini A et une substitution $\sigma : A \rightarrow A^*$. On suppose que σ est non effaçante, c'est-à-dire $\forall a \in A, \sigma(a) \neq \varepsilon$. On suppose qu'il existe une lettre a de A telle que $\sigma(a)$ est un mot commençant par a et de longueur au moins 2. Dans ces conditions, il existe un unique mot infini $x \in A^{\mathbb{N}}$ qui soit point fixe de σ et qui commence par a .

On cherche un algorithme qui engendre la liste des facteurs de longueur inférieure (ou égale) à n de x , dont l'ensemble est noté ici $L_{\leq n}(x)$.

2 Algorithme

Entrée : σ (substitution), a (lettre), n (entier).

Sortie : Liste des facteurs de longueur inférieure à n du point fixe infini de σ qui commence par a .

```
1  $L \leftarrow [a]$  (liste qui contient le mot 'a' comme seul élément, i.e.  $L[0] = a$ )
2  $i \leftarrow -1$  (pointeur)
3 tant que  $i$  est inférieure (ou égal) à la longueur de  $L$  faire
4    $i \leftarrow i + 1$ 
5   Ajouter à  $L$  les facteurs de longueur inférieure à  $n$  de  $\sigma(L[i])$  qui ne sont pas
   déjà des éléments de  $L$ 
6 fin
7 Retourner  $L$ 
```

3 Preuve

3.1 Terminaison

La liste L n'est modifiée qu'à la ligne 5, et de sorte que tous les mots de L sont distincts et de longueur inférieure à n . Le pointeur i augmente de 1 à chaque itération de la boucle

tant que et reste inférieur à la longueur maximale de L qui est inférieur à $\text{card}(A)^n$. Ainsi il y a au plus $\text{card}(A)^n$ itérations dans la boucle **tant que**.

On peut affiner cette borne grâce au théorème de Pansiot datant de 1984 qui affirme que la complexité d'un mot substitutif est dans l'une des classes $\Theta(1)$, $\Theta(n)$, $\Theta(n \log \log n)$, $\Theta(n \log n)$, $\Theta(n^2)$.

Ainsi, il y a $O(n^3)$ itérations dans la boucle **tant que**. Attention à ne pas faire dire n'importe quoi à ce résultat : la constante qui est dans le O dépend de la substitution σ , cette borne ne permet donc pas de déterminer la complexité de l'algorithme (sauf si on fixe σ et qu'on regarde la complexité en n uniquement).

Par ailleurs, si on voulait se lancer dans un calcul de complexité rigoureux, il faudrait expliciter le contenu de la ligne 5 (trop implicite pour être qualifiée d'algorithme) qui, quel que soit l'algorithme choisi, ne prendra probablement pas un temps uniformément borné.

3.2 Correction

On veut montrer qu'à la fin de l'exécution de l'algorithme, $L = L_{\leq n}(x)$ (en considérant L comme un ensemble).

L'inclusion \subseteq est évidente : à l'initialisation L ne contient que le mot 'a' qui est dans $L_{\leq n}(x)$ puisque x commence par a . La propriété est préservée à chaque itération de la boucle **tant que**, puisque si w est dans $L_{\leq n}(x)$, alors tout facteur de $\sigma(w)$ de longueur inférieure à n l'est aussi (l'ensemble des facteurs de x est stable par σ puisque $x = \sigma(x)$). Pour l'inclusion \supseteq , on veut montrer qu'on a oublié personne. On raisonne par l'absurde en supposant que $L_{\leq n}(x) \setminus L$ est non vide. Soit w le mot de $L_{\leq n}(x) \setminus L$ qui est de longueur minimale et qui apparaît le premier dans $x = \sigma(x) = x_0x_1x_2\dots$. Autrement dit, $w = x_i\dots x_{i+k-1}$ avec k et i les plus petits possibles (dans cet ordre) tels que w n'apparaît pas dans L . $x = \sigma(x)$ donc il existe k' et i' (choisis les plus petits possibles dans cet ordre) tels que w est facteur de $\sigma(x_{i'}\dots x_{i'+k'-1})$. La substitution est non effaçante donc $k' \leq k$. De plus, x commence par a et $\sigma(a)$ a une longueur supérieure à 2 donc $i' < i$. Ainsi, par minimalité de (k, i) , $x_{i'}\dots x_{i'+k'-1}$ est dans L . Lorsque la boucle **tant que** traitera $x_{i'}\dots x_{i'+k'-1}$, elle ajoutera w à L , car c'est un facteur de $\sigma(x_{i'}\dots x_{i'+k'-1})$ de longueur inférieure à n . Contradiction.

3.3 Accélération

Au niveau de la ligne 5, on peut simplement écrire $\sigma(L[i])$, énumérer ses facteurs de longueur inférieure à n et ne rajouter à L que ceux qui n'y sont pas.

On peut aussi éviter beaucoup de calcul inutile par exemple pour des i assez grands, $L[i] = L[j].L[k]$ avec $j, k < i$ de sorte que beaucoup de travail a déjà été fait, les seuls nouveaux facteurs qui peuvent apparaître dans $\sigma(L[i]) = \sigma(L[j]).\sigma(L[k])$ sont ceux à cheval entre $\sigma(L[j])$ et $\sigma(L[k])$.

Pour des i encore plus grands, $L[i] = u.v.w$ avec $u.v = L[j]$ et $v.w = L[k]$ et $j, k < i$. Dans ce cas, on ne doit traiter que les facteurs de $\sigma(L[i]) = \sigma(u).\sigma(v).\sigma(w)$ qui sont à cheval autour de $\sigma(v)$. Si $\sigma(v)$ a une longueur supérieure à n , il n'y a rien à faire.

Expliciter la ligne 5 en l'optimisant est un bon exercice.