TD 2 – Recombinaisons et recherche en espace constant

Margot Catinaud margot.catinaud@lmf.cnrs.fr
Valentin Dardilhac valentin.dardilhac87@gmail.com

Dans ce TD, on considère Σ un alphabet fini.

I Recombinaison de mots

Soient $u, v \in \Sigma^*$ deux mots. On note $u \circlearrowleft v$ lorsque l'on peut passer de u à v par une rotation des lettres. Plus précisément, si u se décompose en xy, avec $x, y \in \Sigma^*$ deux mots, et que $u \circlearrowleft v$, alors v se décompose en yx. Par exemple, on a $abcad \circlearrowleft cadab$ où $a, b, c, d \in \Sigma$.

Question 1 *Donner un algorithme qui teste si $u \circlearrowleft v$ en temps $\Theta(|u|)$.*

On note $\widetilde{u}$ le mot obtenu en retournant u. Par exemple, $\widetilde{abcc} = ccba$.

Définition 1 : Recombinaison

On dit qu'*un mot u se recombine en v*, noté $u \sim v$, lorsque u peut se décomposer en $u = u_1 u_2 u_3$ de sorte que $v = u_1 \widetilde{u_2} u_3$, *i.e.* lorsque v est obtenu en retournant un facteur quelque part dans u.

Remarque : Ce genre de transformation apparaît en biologie, lorsque les gènes sont copiés.

Question 2 *Soient $a \in \Sigma$ une lettre et $u, v \in \Sigma^*$ deux mots. Montrer l'équivalence suivante :*

$$au \sim av \iff u \sim v.$$

Qu'en déduire de la relation entre $ua \sim va$ et $u \sim v$?

Question 3 *En déduire un algorithme efficace qui teste, étant donnés deux mots u et v, si $u \sim v$. On précisera la complexité de l'algorithme en fonction de la taille des mots u et v.*

Soient $\mathcal{R}$ et $\mathcal{R}'$ deux relations sur les mots de Σ^*. On définit la relation $\mathcal{R} \circ \mathcal{R}'$ comme suit :

$$\forall\, u, v \in \Sigma^*,\ u \mathcal{R} \circ \mathcal{R}' v \overset{\text{def}}{\iff} \Big[\exists\, w \in \Sigma^*,\ \big(u \mathcal{R} w\big) \wedge \big(w \mathcal{R}' v\big)\Big].$$

Question 4 *Montrer que, pour deux mots $u, v \in \Sigma^*$ tels que $u \sim \widetilde{v}$, on a $u\big(\circlearrowleft \circ \sim \circ \circlearrowleft\big)v$.*

Question 5 *Montrer que, pour deux mots $u, v \in \Sigma^*$ tels que $u\big(\sim \circ \circlearrowleft\big)v$, alors*

$$\Big(u\big(\circlearrowleft \circ \sim\big)v\Big) \vee \Big(u\big(\circlearrowleft \circ \sim\big)\widetilde{v}\Big).$$

On étend désormais la notion de recombinaison par la définition suivante :

$$\forall\, u, v \in \Sigma^*,\ u \approx v \overset{\text{def}}{\iff} \Big[\exists\, u', v' \in \Sigma^*,\ u \circlearrowleft u' \sim v' \circlearrowleft v\Big].$$

Autrement dit, $u \approx v$ si et seulement si $u\big(\circlearrowleft \circ \sim \circ \circlearrowleft\big)v$.

Question 6 *Proposer un algorithme efficace qui teste, étant donnés deux mots $u, v \in \Sigma^*$, si $u \approx v$. On établira la complexité de l'algorithme en fonction de la taille des mots.*
Indication : *Commencer par montrer la propriété suivante :*

$$u \approx v \iff \Big(u\big(\circlearrowleft \circ \sim\big)v\Big) \vee \Big(u\big(\circlearrowleft \circ \sim\big)\widetilde{v}\Big).$$

II Recherche en espace constant

II.1 Recherche d'un motif auto-maximal

On fixe un ordre total $\preccurlyeq_\Sigma$ sur l'alphabet Σ, et on note $\mathsf{MaxSuf}(w)$ le suffixe maximal d'un mot $w \in \Sigma^*$ au sens de l'ordre lexicographique induit par $\preccurlyeq_\Sigma$. Le mot w est dit *auto-maximal* lorsque $\mathsf{MaxSuf}(w) = w$.

Définition 2 : Période

On dit que $p \in [\![1; |u|]\!]$ est une *période* d'un mot $u \in \Sigma^*$ lorsque, pour tout $i \in [\![1; |u|-p]\!]$, $u[i] = u[i+p]$. On note $\mathsf{Period}(u)$ la plus petite période de u.

Lors du TD précédent, on a vu que, pour tout mot $w \in \Sigma^*$, $\mathsf{Period}(w) = |w| - \pi_w(|w|)$.

Question 7 *Montrer qu'il existe des mots qui ne sont pas auto-maximaux, quel que soit l'ordre $\preccurlyeq_\Sigma$ choisit sur l'alphabet Σ.*

Question 8 *Montrer que tout les préfixes d'un mot auto-maximal sont également auto-maximaux.*

On considère l'algorithme suivant :

Algorithme 1 : Calcul naïf de la plus petite période d'un mot.

```
fun periode_naive w j =
    p ← 1;
    pour i = 2 à j faire
        si w[i] ≠ w[i − p] alors p ← i;
    retourner p;
```

Question 9 *Soit $w \in \Sigma^*$ un mot auto-maximal et $j \in [\![1; |w|]\!]$. Montrer que* **periode_naive** w j *calcule bien l'entier* $\mathsf{Period}(w[1:j])$.

Indication : *Proposer l'invariant de boucle adéquat et le montrer en utilisant le lemme suivant :*

Lemme 1 :

Pour $p = \mathsf{Period}(w[1:i-1])$, si $w[i] \neq w[i-p]$, alors $w[i] \prec_\Sigma w[i-p]$ et $\mathsf{Period}(w[1:i]) \geqslant i$.

Question 10 *On suppose que le motif w à rechercher est auto-maximal. Adapter l'algorithme de* Morris-Pratt *pour le faire fonctionner en espace mémoire* ***constant****. Plus exactement, le nombre de variables codant des entiers doit rester constant (ce qui correspond en réalité à un espace logarithmique). Montrer alors que la complexité en temps de cette adaptation reste linéaire.*

II.2 Recherche de texte en espace constant

On suppose connue la décomposition $w = u \cdot v$ avec $v = \mathsf{MaxSuf}(w)$. On recherche alors les occurrences de w à l'intérieur d'un texte quelconque $T \in \Sigma^*$.

Question 11 *On suppose dans cette question que v apparaît avec un décalage de i dans T. Montrer que u ne peut apparaître avec un décalage de j dans T, pour tout $j \in [\![1 - |u| + 1; i]\!]$.*

Devoir maison 2 *En déduire un algorithme de recherche de w dans T en temps linéaire et espace constant.*