TD 2: Recherche en espace constant et recombinaisons

1 Recherche de texte en espace constant

On rappelle que $Period(w[1,i]) = i - \pi(i)$. De plus, lorsque w est automaximal, on sait calculer $Period(w[1,i{+}1])$ à partir de $Period(w[1,i])$ en temps constant. Plus exactement, la suite $(Period(w[1,i]))_i$ est croissante et prend éventuellement une nouvelle valeur k exactement à l'indice k.

Exercice 1 On suppose que P est auto-maximal. Adapter l'algorithme de Morris-Pratt pour le faire fonctionner en espace constant, puis prouver que son temps d'exécution reste linéaire.

Dans la suite, on suppose avoir calculé la décomposition $P = uv$ où $v = MaxSuf(P)$.

Exercice 2 Concevoir un algorithme de recherche de motif pour P qui s'appuie sur l'algorithme MPAuto et qui fonctionne en espace mémoire constant.

Exercice 3 Optimiser l'algorithme précédent en prouvant la propriété suivante : si v apparaît avec un décalage de i dans T, alors uv ne peut apparaître avec un décalage de j dans T, pour tout $j \in \{i - |u|{+}1, i{-}|u|{+}2, \ldots, i\}$. Montrer que l'algorithme obtenu fait au plus $|T|$ comparaisons supplémentaires par rapport à MPAuto.

2 Recombinaisons de mots

Exercice 4 Donner un algorithme qui teste en temps $O(n)$ si deux mots de longueurs n sont des rotations circulaires l'un de l'autre. Par exemple, *abcad* et *cadab* sont des rotations circulaires l'un de l'autre.

On considère des mots $u, v, w, \ldots$ sur un alphabet fini $\Sigma = \{a, b, \ldots\}$ et on notera $\widetilde{u}$ le mot obtenu en retournant u. P.ex. $\widetilde{abcc} = ccba$.

On dit qu'un mot u se recombine en v, noté $u \sim v$, si u peut se décomposer en $u = u_1u_2u_3$ de sorte que $v = u_1\widetilde{u_2}u_3$, c.-à-d. que v est obtenu en retournant un facteur quelque part dans u. Ce genre de transformation apparaît en biologie, quand des gènes sont copiés.

Exercice 5 Montrez que $au \sim av$ ssi $u \sim v$.

Exercice 6 En déduire un algorithme efficace qui teste, étant donnés deux mots u et v, si $u \sim v$. Vous établirez sa complexité en fonction de la taille n des mots.

Soit $\mathcal{R}, \mathcal{R}'$ deux relations. La relation $\mathcal{R}\mathcal{R}'$ est définie par $u\mathcal{R}\mathcal{R}'v$ ssi $\exists w\ u\mathcal{R}w \wedge w\mathcal{R}'v$. On note $u \circlearrowleft v$ quand on peut passer de u à v par une rotation des lettres (i.e. u se décompose en xy et v en yx).

Exercice 7 Montrer que $u \sim \widetilde{v} \Rightarrow u \circlearrowleft\!\!\sim\!\!\circlearrowleft v$.

Exercice 8 Montrer que $u \sim\!\!\circlearrowleft v \Rightarrow u \circlearrowleft\!\!\sim v$ ou $u \circlearrowleft\!\!\sim \widetilde{v}$.

On étend la notion de recombinaison par la définition suivante : $u \approx v$ ssi il existe deux mots u' et v' tels que $u \circlearrowleft u' \sim v' \circlearrowleft v$ (autrement dit, $u \approx v$ ssi $u \circlearrowleft\!\!\sim\!\!\circlearrowleft v$).

Exercice 9 Proposez un algorithme efficace qui teste, étant donnés deux mots u et v, si $u \approx v$. Vous établirez sa complexité en fonction de la taille n des mots.