TD 3 – Fonction témoin et palindromes

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

Dans ce TD, on note Σ un alphabet fini.

I Calcul d'une fonction témoin – variations sur Boyer-Moore

Dans cette partie, on étudie un algorithme de recherche des positions d'un motif $P \in \Sigma^*$ dans un texte $T \in \Sigma^*$ en utilisant autrement la fonction Pref de l'algorithme Boyer-Moore. L'idée est de calculer en temps linéaire en $|T|$ un ensemble de positions possibles de début du motif, puis de tester en temps linéaire (en $|P|$) chaque position possible.

Soit $P \in \Sigma^*$ un motif de longueur $m \in \mathbb{N}$. Une fonction $\mathsf{tem} : [\![1; m-1]\!] \longrightarrow [\![0; m-1]\!]$ est appelée *fonction témoin* lorsqu'elle vérifie la spécification suivante :

- $\mathsf{tem}(i) \neq 0 \iff \exists\, k \in [\![1; m-i]\!],\ P[i+k] \neq P[k]$;
- $\mathsf{tem}(i) \neq 0 \implies P[i + \mathsf{tem}(i)] \neq P[\mathsf{tem}(i)]$.

En général, il existe plusieurs fonctions témoins pour un motif : pour tout $i \in [\![1; m-1]\!]$, $\mathsf{tem}(i)$ est égal à l'un des entiers k vérifiant la première propriété.

Question 1 *Proposer un algorithme en $\Theta(m)$ de calcul d'une fonction témoin. Pour cela, on pourra utiliser la fonction* Pref *vu en cours.*

On se donne maintenant un texte $T \in \Sigma^*$ de longueur $n \in \mathbb{N}$. Deux positions $i, j \in [\![1; n-m+1]\!]$ telles que $i \neq j$ du texte T sont dites *incohérentes* lorsque $j - i < m$ et $\mathsf{tem}(j-i) \neq 0$.

Question 2 *Soient $i, j \in [\![1; n-m+1]\!]$ deux positions incohérentes. Montrer que,* ***quel que soit le texte*** *T, on est dans au moins l'un des deux cas suivants :*

(i) Soit $T[i : i+m-1] \neq P$;

(ii) Soit $T[j : j+m-1] \neq P$.

Question 3 *Soient $i, j, k \in [\![1; m]\!]$ trois positions. Montrer que si $i < j$ sont deux positions cohérentes et $j < k$ sont deux positions cohérentes, alors les positions i et k sont cohérentes.*

On se propose dans la suite de calculer un ensemble de positions de T, toutes cohérentes entre elles et qui soit un sur-ensemble des positions d'où démarre le motif P recherché. À cette fin, on gère une pile initialement vide et on parcourt les positions de 1 à $n - m + 1$. De plus, à chaque itération, on effectue les opérations suivantes :

- On empile la position courante ;
- Tant que la pile a au moins deux éléments et que les deux positions $i < j$ au sommet de la pile sont incohérentes, alors, dans le cas où $T[j - 1 + \mathsf{tem}(j-i)] \neq P[\mathsf{tem}(j-i)]$, on supprime j de la pile. Dans le cas contraire, on supprime i de la pile.

L'ensemble recherché Possible est l'ensemble des positions présentes dans la pile à l'issue de l'algorithme sus-définit.

Question 4 *Montrer que cet algorithme vérifie la spécification espérée et qu'il opère en temps $\Theta(n)$.*

À partir de l'ensemble Possible, on construit un texte T' de longueur n sur l'alphabet $\{0, 1\}$ en parcourant le texte T à l'aide d'un indice i variant de n à 1. On maintient simultanément un entier j égal à la plus grande position k de Possible telle que $k \leqslant i$. Si une telle position n'existe pas, on convient que $j = 0$.

- Si on est dans l'un des cas suivants (i) $j = 0$, (ii) $i - j \geqslant m$, ou (iii) $T[i] \neq P[i-j+1]$, alors $T'[i] \stackrel{\text{def}}{=} 0$.
- Sinon, $T'[i] \stackrel{\text{def}}{=} 1$.

Question 5 *Montrer que cet algorithme ainsi définit opère en $\Theta(n)$ étapes et que, pour tout $i \in \mathsf{Possible}$, on a :*

$$T'[i : i+m-1] = 1^m \iff T[i : i+m-1] = P.$$

Question 6 *En déduire un algorithme opérant en temps $\Theta(n)$ en maintenant un unique compteur pour trouver toutes les positions i de l'ensemble* $\mathsf{Possible}$ *telles que $T'[i : i+m-1] = 1^m$.*

II Plus long palindrome d'une chaîne

Pour un mot $\sigma \in \Sigma^*$, on désigne par $\widetilde{\sigma}$ le *mot miroir de σ*, définit inductivement par

$$\widetilde{\varepsilon} \stackrel{\text{def}}{=} \varepsilon \qquad \text{et} \qquad \forall\, a \in \Sigma,\ \widetilde{a\sigma} \stackrel{\text{def}}{=} \widetilde{\sigma}a.$$

On appelle *mot palindrome* tout mot de la forme $\sigma\widetilde{\sigma}$ ou $\sigma a\widetilde{\sigma}$ avec $\sigma \in \Sigma^*$ un mot et $a \in \Sigma$ une lettre.

Le problème du plus long palindrome consiste à déterminer le plus long facteur d'un texte $T \in \Sigma^*$ qui soit un palindrome. La solution naïve est en temps quadratique $\Theta(|T|^2)$. Une autre solution, proposée par Manacher, résoud le même problème seulement en temps linéaire $\Theta(|T|)$.

Tout d'abord, on transforme l'entrée T en insérant un séparateur # autour de chaque caractère et en ajoutant des sentinelles ! et \$ autour du texte T. Le nouveau texte ainsi obtenu est noté $\widehat{T}$. Par exemple, le mot $u = \mathsf{abc}$ se transforme en $\widehat{u} \stackrel{\text{def}}{=}$!#a#b#c#\$. Cette transformation permet alors de traiter de manière uniforme les palindromes de longueur paire et impaire.

Question 7 *Montrer que tout palindrome maximal facteur du texte T se retrouve dans le texte transformé $\widehat{T}$ entouré de caractères # et que les limites des palindromes dans $\widehat{T}$ se trouvent à des positions ayant la même parité. Comment se traduit alors dans le texte T une solution du problème pour $\widehat{T}$ donnée sous la forme (i, r) où i est la position du centre du palindrome et r son rayon (*i.e. *le palindrome solution de $\widehat{T}$ est le mot $\widehat{T}[i-r : i+r]$) ?*

La sortie de l'algorithme doit être un tableau p de longueur égale à $|\widehat{T}|$ qui indique, pour chaque position $i \in [\![1; |\widehat{T}|]\!]$, le plus grand rayon $r \geqslant 0$ tel que le mot $\widehat{T}[i-r : i+r]$ est un palindrome.

Question 8 *Ecrire l'algorithme naïf qui, pour chaque position $i \in [\![1; |\widehat{T}|]\!]$, augmente la valeur $p[i]$ jusqu'à trouver le plus grand palindrome centré en i. Prouver la complexité annoncée pour l'algorithme naïf.*

L'algorithme linéaire est obtenu en exploitant les propriétés des facteurs d'un palindrome pour un calcul plus rapide de p. Ainsi, lors du parcours de $\widehat{T}$ de gauche à droite, on maintient la position du centre c du palindrome de rayon non nul le plus à droite (*i.e.* la position de la fin du palindrome est la plus à droite). Lorsque $c + p[c]$ est inférieur à la position courante i, $p[i]$ est initialisé à 0 et on applique l'algorithme naïf.

Question 9 *Dans le cas où $c + p[c] > i$, prouver que $p[i]$ peut être initialisé à $p[2 * c - i]$.*

Devoir maison 3 *Utiliser la propriété démontrée dans la question précédente pour écrire l'algorithme de calcul de p puis prouver que sa complexité est linéaire en temps.*