TD 3 :
Fonction témoin et palindromes

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

Dans cette partie, on étudie un algorithme de recherche des positions d'un motif P dans un texte T 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 un motif de longueur m. Une fonction $tem : \{1, \ldots, m-1\} \mapsto \{0, \ldots, m-1\}$ est dite fonction *témoin* si elle vérifie la spécification suivante :

- $tem(i) \neq 0$ ssi il existe (au moins) un $1 \leq k \leq m - i$ tel que $P[i + k] \neq P[k]$
- si $tem(i) \neq 0$ alors $P[i + tem(i)] \neq P[tem(i)]$

Généralement il y a plusieurs fonctions témoins pour un motif (pour tout i, $tem(i)$ est égal à l'un des k vérifiant la première propriété).

Question 1 : Proposer un algorithme en $O(m)$ de calcul d'une fonction témoin. *Indication :* Utilisez la fonction *Pref* du cours.

On se donne maintenant un texte T de longueur n. Deux positions du texte $1 \leq i < j \leq n-m+1$ sont dites *incohérentes* si $j - i < m$ et $tem(j - i) \neq 0$.

Question 2 : Soient i et j deux positions incohérentes. Montrer que, **quel que soit** T, on a soit $T[i, i + m - 1] \neq P$, soit $T[j, j + m - 1] \neq P$ (les deux inégalités peuvent être vérifiées).

Question 3 : Démontrer que si $i < j$ sont deux positions cohérentes et $j < k$ sont deux positions cohérentes, alors i et k sont deux positions cohérentes.

On se propose de calculer un ensemble de positions de T, toutes cohérentes entre elles, qui soit un sur-ensemble des positions d'où démarre le motif. À cette fin, on gère une pile initialement vide et on parcourt les positions de 1 à $n - m + 1$. 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 au sommet $i < j$ sont incohérentes, si $T[j - 1 + tem(j - i)] \neq P[tem(j - i)]$, alors on supprime j de la pile, sinon on supprime i.

L'ensemble recherché *Possible* est l'ensemble des positions dans la pile à l'issue de l'algorithme.

Question 4 : Montrer que cet algorithme vérifie sa spécification et qu'il opère en $O(n)$.

À partir de *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 $k \leq i$ ($j = 0$ s'il n'existe pas une telle position).

— Si $j = 0$ ou $i - j \geq m$ ou $T[i] \neq P[i - j + 1]$ alors $T'[i] = 0$.

— Sinon $T'[i] = 1$.

Question 5 : Montrer que cet algorithme opère en $O(n)$ et que pour tout i de *Possible* $T'[i, i + m - 1] = 1^m$ si et seulement si $T[i, i + m - 1] = P$.

Question 6 : Proposer un algorithme qui opère en temps $O(m)$ en maintenant un unique compteur pour trouver les positions i de *Possible* telles que $T'[i, i + m - 1] = 1^m$.

2 Le plus long palindrome d'une chaîne

Si σ est un mot, $\widetilde{\sigma}$ désigne le *miroir* de σ, défini inductivement par $\widetilde{\varepsilon} = \varepsilon$ et $\widetilde{a\sigma} = \widetilde{\sigma}a$ avec a une lettre. Un *palindrome* est un mot de la forme $\sigma\widetilde{\sigma}$ ou $\sigma a\widetilde{\sigma}$ avec σ un mot et a une lettre.

Le problème du plus long palindrome consiste à déterminer le plus long facteur d'un texte T qui est un palindrome. La solution naïve est en temps $O(|T|^2)$ mais une solution en temps $O(|T|)$ existe, proposée par Manacher.

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. Le nouveau texte est noté $\widehat{T}$. Par exemple, `abc` se transforme en `^#a#b#c#$`. Cette transformation permet de traiter de manière uniforme les palindromes de longueur paire et impaire.

Question 7 : Montrez 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 en T une solution du problème pour $\widehat{T}$ donnée sous la forme position du centre du palindrôme et son rayon ?

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

Question 8 : Ecrivez l'algorithme naïf qui, pour chaque position i, augmente $p[i]$ jusqu'à trouver le plus grand palindrome centré en i. Prouvez la complexité annoncée.

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 (dont la fin est la plus à droite). Si $c + p[c]$ est inférieur à la position courante i, $p[i]$ est initialisé à 0 et on applique l'algorithme naïf.

Question 9 : Prouvez que si $c+p[c] > i$, alors $p[i]$ peut être initialisé à $\min(p[2*c-i], p(c)+c-i)$.

Question 10 : (DM) Utilisez la propriété ci-dessus pour écrire l'algorithme de calcul de p et prouvez sa complexité linéaire en temps.