Université de Nice-Sophia Antipolis — Licence 3 Informatique
UE Automates & Langages — 2011–2012

TP n°3

Reconnaissance de motifs (suite)

On se propose de poursuivre l'implantation des algorithmes de recherche de motif les plus répandus. Avant de commencer, veillez à terminer tous les exercices du TP précédent.

L'algorithme de Knuth-Morris-Pratt

L'algorithme de Knuth-Morris-Pratt (1977) compare un texte et un motif de la gauche vers la droite. Dès qu'ils ne coïncident plus, l'algorithme recherche dans la partie du motif qui a déjà été lue le plus long suffixe qui est aussi préfixe du motif. Il en déduit de combien le motif peut être décalé à droite, sans qu'aucune occurence du motif ne soit ratée. Ainsi, le motif fait l'objet d'un prétraitement où ses propres redondances sont détectées.

Exercice 1 : prétraitement du motif

1. Reprenez le squelette fourni au TP précédent.
2. On veut remplir une table qui ne dépend que du motif. A la fin, `table[i]` doit contenir l'indice à partir duquel on peut repartir dans le motif si la coïncidence avec le texte a échoué à l'indice `i`. Comprenez le rôle de cette table en vous aidant de l'exemple suivant.
   Exemple : Le prétraitement du motif `murmure` donne la table suivante de même longueur que le motif :

0	1	2	3	4	5	6
-1	0	0	0	1	2	3

   En effet, si à la place du deuxième `r` du motif (position `i=5`) on lit autre chose dans le texte, on passe l'indice du motif à `table[5]=2`, ce qui revient à faire glisser le motif à droite. Autrement dit, on a lu `murmu` dans le texte puis cela a échoué. De la partie lue avant l'échec, on conserve le plus grand suffixe qui est aussi préfixe du motif, c'est `mu`.
3. Ecrivez une méthode dont l'entête est le suivant :
   ```
   static int [] construireTable (String motif)
   ```
   Elle contient principalement l'extrait suivant :

```
int i = -1;
table[0] = -1;
for (int j=1; j< lgMotif; j++)
     while (i != -1 && motif.charAt(i) != motif.charAt(j-1))
          i = table[i];
     i = i+1;
     table[j]=i;
```

Exercice 2 : implantation de l'algorithme

1. Ecrivez la fonction de recherche à l'entête suivant :
   ```
   static int recherche (String motif, String texte)
   ```
   Elle est similaire à celle de l'algorithme naïf excepté que quand la lecture du motif échoue après en avoir lu un préfixe, on consulte la table pour savoir où se repositionner dans le motif.

2. *Facultatif* Si vous avez bien compris, vous pouvez encore améliorer un peu cet algorithme et vous obtiendrez alors le véritable algorithme KMP, celui-ci n'étant que l'algorithme MP.

L'algorithme de Aho-Corasik

L'algorithme de Aho-Corasick date de 1975 et est implanté dans `grep` d'UNIX. Pour rechercher un motif dans un texte, il utilise tout simplement un automate fini. Vous savez déjà construire un automate minimal pour reconnaître un motif donné. Ce que vous ne saviez pas, c'est qu'un tel automate peut être construit inductivement sur la longueur du motif. Ainsi, cet algorithme ne nécessite pas de prétraitement du motif et il est également adapté à la recherche incrémentale.

L'algorithme est en deux temps. Tout d'abord, il faut construire l'automate pour un `motif` donné, ensuite, il faut le faire fonctionner sur un `texte` donné. Nous n'aurons peut-être pas le temps de l'implanter en totalité ... mais il serait déjà bien de savoir construire l'automate.

Exercice 3 : construction de l'automate

1. Voici sous forme algorithmique la méthode pour construire l'automate minimal reconnaissant un `motif` sur un alphabet A donné.

```
fonction construireAutomate (motif) : automate
     créer un état init
     terminal := init
     pour tout b dans A faire
          delta  (init, b) :=  init
     pour a de la première à la dernière lettre du motif faire
          temp := delta (terminal,  a)
          delta (terminal,  a) := nouvel état x
          pour tout b de A faire
               delta (x,  b) := delta (temp, b)
          terminal := x
     end
     return automate
```

   Considérons l'alphabet $A = \{0, 1, 2\}$. Fabriquez à la main et en suivant cette méthode l'automate reconnaissant la présence du motif `1012` dans un texte.

2. Programmez cette fonction en ayant soin de choisir une bonne représentation pour l'automate.
3. *Facultatif* Pour terminer, complétez votre programme pour qu'il mime le fonctionnement d'un automate sur un texte passé en argument.