Recherche de motifs

Algorithmes avancés - ITI orientation logiciel et systèmes complexes

Michel Lazeyras, Paul Albuquerque

Table des matières

1 Introduction

Ce document traite de la recherche de motifs dans un texte, un motif étant une suite de lettres (un mot par exemple) de l'alphabet sur lequel le texte est construit. Ce problème se retrouve dans de nombreux contextes : les éditeurs de texte (rechercher et remplacer), les moteurs de recherche, les outils de décodage du génome humain par exemple.

Le texte et le motif sont représentés par des chaînes de caractères de longueurs respectives t et m, avec $m \leq t$. Les éléments du texte, et donc du motif, appartiennent à un alphabet fini, noté Σ. Le but des algorithmes présentés ici est de trouver toutes les occurences du motif à l'intérieur du texte, chacune d'entre-elles étant repérée par la première lettre du motif dans le texte (décalage).

Pour plus de renseignements sur la recherche de motifs, citons deux ouvrages :

- T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. *Introduction à l'algorithmique*, 2ème édition, Dunod, chap. 32, pp. 875-900.
- M. Crochemore, C. Hancart, T. Lecroq. *Algorithmique du texte*, Vuibert.

Dans tout ce qui suit :

- T représentera le texte exploré, $|T|$ sa longueur ($|T|$ est aussi noté t.
- M représentera le motif de longueur $|M|$ ($|M|$ est aussi noté m).
- $T[i]$ représentera la i-ème lettre de T, idem pour M.
- $T[1..5]$ représentera les 5 premières lettres de T, idem pour M.
- d représentera un décalage, c'est-à-dire la position précédant la première lettre du motif dans le texte, plus formellement : $M[1..m] = T[d + 1..d + m]$.

2 Algorithme naïf

L'algorithme naïf trouve tous les décalages en déplaçant le motif de gauche à droite sur le texte. A chaque étape, tous les caractères du motif sont comparés avec les m caractères du texte correspondant.

Exemple Rechercher le motif 'ECI' dans le texte 'CECITE DE CECILE'

```
CECITE DE CECILE     CECITE DE CECILE    CECITE DE CECILE
ECI                   ECI                  ECI
```

Et ainsi de suite jusqu'à la fin du texte. Il y aura deux réponses pour le décalage : 1 et 11.

Algorithme

```
Naïf(T,M)
pour d de 0 à t-m faire
   si M[1..m] = T[d+1..d+m] alors
      imprimer d
   fin si
fin faire
```

Derrière le test se cache une boucle de complexité $\Theta(m)$, la boucle globale étant quant à elle de complexité $\Theta(t-m+1)$, ce qui fait que l'algorithme est de complexité $\Theta((t-m+1)m)$. En abrégeant la boucle correspondant au test dès la première occurence fausse, la complexité est améliorée.

Exercices

Ex. 1 *Ecrire une procédure ou une fonction acceptant un texte et un motif en paramètre et imprimant tous les décalages.*

Ex. 2 *Si tous les caractères du motif sont différents, trouver le moyen d'accélérer l'algorithme naïf afin qu'il prenne un temps $\Theta(t)$ pour trouver tous les décalages.*
Ecrire un organigramme de cet algorithme.

3 Algorithme de Rabin-Karp

L'algorithme de Rabin-Karp utilise une notion qui fait penser au hachage que l'on rencontre dans les fichiers à accès direct. Chacune des lettres de l'alphabet est remplacée par un chiffre défini dans une base notée B assez grande (au moins $|\Sigma|$). L'algorithme utilise la théorie des nombres, en particulier l'équivalence de deux nombres modulo un troisième.

Afin de faciliter la compréhension de l'algorithme, l'alphabet utilisé ici sera : $\Sigma = 0, 1, 2, ..., 9$, ce qui fait que $B = 10$. Les chaînes de caractères correspondront donc à des nombres : $T =$ "2450" correspondra au nombre 2450. On notera p la valeur numérique de M et t_s celle de $T[s+1..s+m]$. Le calcul de p se fera comme suit :

$$p = M[m] + 10(M[m-1] + 10(M[m-2] + ... + 10M[1])...)$$

Le calcul de t_s s'effectuera de la même manière pour $s = 0$, pour les valeurs suivantes de s le calcul se fera comme suit :

$$t_{s+1} = 10(t_s - 10^{m-1}T[s+1]) + T[s+m+1]$$

Exemple $m = 3$ et $T = 47368$

$$t_0 = 473$$
$$t_1 = 10(473 - 100 * 4) + 6 = 736$$
$$t_2 = 10(736 - 100 * 7) + 8 = 368$$

L'algorithme de Rabin-Karp s'exécute comme l'algorithme naïf, en comparant p avec $t_s, 0 < s < t-m$. Les différents nombres pouvant être très grands, cette comparaison se fera modulo un 3ème nombre q qui sera premier avec t et m. Chaque fois que $p = t_s \mod q$, il faudra vérifier que les chaînes sont bien identiques. Un pré-traitement permettra de calculer p.

Algorithme

```
Rabin-Karp(T,M,B,q)
calculer p mod q
calculer t_0 mod q
pour s de 0 à t-m faire
   si p = t_s mod q alors
      si M[1..m] = T[s+1..s+m] alors
         imprimer s
      fin si
   fin si
   calculer t_{s+1} si s+1 < t-m
fin faire
```

Le pré-traitement correspondant à la 1ère boucle est de complexité $\Theta(m)$. Comme pour l'algorithme naïf, le test est une boucle de complexité $\Theta(m)$, la boucle globale étant quant à elle de complexité $\Theta(t - m + 1)$, ce qui fait que l'algorithme est de complexité $\Theta((t - m + 1)m)$ dans le pire des cas.

Exercices

Ex. 3 *Ecrire complètement le calcul de t_{s+1} de l'algorithme de Rabin-Karp, en utilisant $t_s \mod q$.*

Ex. 4 *Ecrire l'organigramme de l'algorithme de Rabin-Karp.*

Ex. 5 *Appliquer l'algorithme de Rabin-Karp pour :*
- *le texte $T = 314159265358$*
- *le motif $M = 26$*
- *la base $B = 10$*
- *le modulo $q = 11$.*

4 Automates finis

Les automates finis permettent d'examiner chaque caractère du texte une seule fois. En revanche ils nécessitent, comme dans le cas de l'algorithme de Rabin-Karp, un pré-traitement. Avant d'aller plus loin, voyons ce qu'est un automate fini.

Définition

Un automate fini est défini par 5 éléments :

- Q un ensemble fini d'états de l'automate
- q_0 : l'état initial, $q_0 \in Q$
- $A \subseteq Q$: un ensemble d'états terminaux
- Σ : un alphabet fini
- δ : une fonction de $Q \times \Sigma$ vers Q appelée fonction de transition de M.

Dans cet algorithme, l'automate fini permettra de savoir ce qu'il y a à faire chaque fois que l'on a comparé une lettre du texte avec une lettre du motif afin de ne pas perdre de décalage en route. L'automate sera construit sur M uniquement et indiquera où l'on en est dans M après chaque comparaison entre M et T. Le nombre d'états vaudra $m+1$, soit un état par position dans le motif et un état initial qui correspond à l'attente de la première lettre du motif. L'état terminal correspondra à la dernière lettre du motif. La fonction de transition indiquera à quel état passer lorsque l'on a une nouvelle entrée, c'est-à-dire une nouvelle lettre du texte.

La figure 1 donne un exemple d'automate fini à deux états (1 et 2) avec l'alphabet $\Sigma = a, b$. L'état initial est 1, l'état final est 2. La fonction de transition est interprétée comme suit : entrée dans l'état 1 ; s'il arrive un b, passage à l'état 2 ; puis s'il arrive un a, retour à l'état 1 ; dans les autres cas, il n'y a pas de changement d'état.

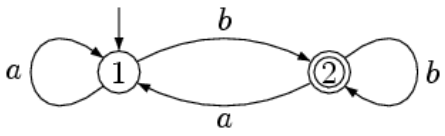

FIGURE 1 – Exemple d'automate fini déterministe

L'automate fini déterministe de la figure 1 peut être représenté par un tableau comme suit :

	a	b
état 1	1	2
état 2	1	2

Le pré-traitement consiste à définir δ. Il s'agit de remplir un tableau à deux dimensions comme ci-dessus, donnant pour chaque couple (état, entrée) un nouvel état. Une entrée est un caractère de l'alphabet.

Pour le motif $M = ababaca$, l'automate sera constitué de 8 états. L'état 0 correspond à l'entrée dans l'automate ; 7 est l'état final : le motif est trouvé. La fonction de transition sera donnée par le tableau suivant :

	a	b	c
état 0	1	0	0
état 1	1	2	0
état 2	3	0	0
état 3	1	4	0
état 4	5	0	0
état 5	1	4	6
état 6	7	0	0
état 7	1	2	0

Algorithme

```
Automate(T,Tab)
q := 0 (q est un état)
pour i variant de 1 à t faire
   q := Tab(q,T(i))
   si q = m alors
      s := i-m
      imprimer le décalage (s)
   fin si
fin pour
```

Avant l'exécution de l'algorithme, il faut créer l'automate, qui correspond à la fonction de transition. Cette fonction est ensuite insérée dans un tableau (`Tab`) à deux entrées, la première : l'état, la seconde : la lettre.

Exercices

Ex. 6 *Soit l'alphabet $\Sigma = a, b$, construire l'automate de recherche fini du motif $M = aabab$ et appliquer cet automate au texte $T = aaaababaabaababaab$.*

Ex. 7 *Soit l'alphabet $\Sigma = a, b$, dessiner le diagramme de transitions d'état pour un automate de recherche du motif $M = ababbabbababbababbabb$.*

5 Algorithme de Knut-Morris-Pratt

L'algorithme de Knut-Morris-Prat, souvent appelé algorithme KMP, est aussi efficace que celui des automates finis, mais il permet d'éviter le calcul de la fonction de transition δ. Comme pour les deux algorithmes vus précédemment, un pré-traitement doit être effectué sur le motif. Sans compter ce pré-traitement, l'efficacité de l'algorithme est de l'ordre de $\Theta(t)$. Il n'y a jamais de retour en arrière lors du parcours du texte.

La particularité de cet algorithme réside dans le pré-traitement qui permet de sauter un certain nombre de lettres du texte chaque fois qu'il n'y a plus correspondance entre la lettre courante du motif et la lettre courante du texte. En effet, lors de la vérification du texte, un pointeur parcourra le texte et un autre pointeur parcourra le motif tant qu'il y a correspondance. Il s'agit donc de construire un tableau qui indiquera pour chaque lettre du motif où replacer le pointeur lorsque la lettre suivante du texte n'est pas la bonne. Un exemple va permettre de clarifier le propos.

Exemple Soit le motif : $M = ababaca$ à trouver dans le texte :
$T = bacbababaabcbab$.
Le pré-traitement consiste à calculer pour chacune des lettres de M son préfixe π. Pour une position i, on aura donc deux données : $M(i)$ et $\pi(i)$. Seuls les $\pi(i)$ seront utilisés plus tard. Le préfixe $\pi(i)$ est le plus long préfixe de M qui soit suffixe de $M[1..i]$; $\pi(i)$ est défini par :

$$\pi(i) = \max\{k \text{ tel que } k < i \text{ et } M[1..k] \text{ est un suffixe de } M[1..i]\}$$

Le tableau est donc le suivant :

i	1	2	3	4	5	6	7
$M(i)$	a	b	a	b	a	c	a
$\pi(i)$	0	0	1	2	3	0	1

Le déroulement de la vérification marche de la façon suivante : au départ, les positions de T et M sont les suivantes ($i = 0$) :

```
T : b a c b a b a b a a b c b a b
M : a b a b a c a
```

Tant que $T(i) \neq$ `a`, décaler le motif ($i = i + 1$)

```
T : b a c b a b a b a a b c b a b
M :   a b a b a c a
```

$T(3)$ qui vaut `c` est différent de $M(2)$ qui vaut `b` ; le décalage est de 1.
Il faut continuer tant que $T(i) \neq$ `a`

```
T : b a c b a b a b a a b c b a b
M :         a b a b a c a
```

Cette fois, il y a correspondance jusqu'à la lettre `c` du motif ; la dernière bonne lettre est `a` de l'indice 5 du motif. Le tableau nous indique $\pi(5) = 3$

```
T : b a c b a b a b a a b c b a b
M :             a b a b a c a
```

Après un nouveau décalage, le motif dépasse du texte ; l'algorithme s'arrête, le motif n'a pas été trouvé.

```
T : b a c b a b a b a a b c b a b
M :                   a b a b a c a
```

Algorithmes

```
KMP(T,M,Tab_prefixes)
calcul de la table des préfixes: pi
q := 0
pour i variant de 1 à t faire
   tant que q > 0 et M(q+1) /= T(i) faire
      q := pi(q)
   fin faire
   si M(q+1) = T(i) alors
      q := q+1
   fin si
   si q = m alors
      imprimer le décalage c.-à-d. i-m
      q := pi(q)
   fin si
fin faire
```

Avant l'exécution de cet algorithme, il faut calculer la fonction préfixe. Ceci est donné en exercice.

Exercice

Ex. 8 *Concevoir, puis décrire l'algorithme permettant de calculer la fonction préfixe (π). Faire un organigramme de cette fonction.*

6 Algorithme de Boyer-Moore

L'algorithme de Boyer-Moore, comme l'algorithme KMP effectue un prétraitement du motif avant d'effectuer la recherche dans le texte lui-même. Au démarrage, deux tableaux δ_1 et δ_2, lesquels permettront d'effectuer des sauts dans le parcours du texte lui-même, doivent être remplis. Ainsi, la phase de recherche du motif dans le texte prendra un temps inférieur à $\Theta(T)$. En général, plus le motif est long, plus la recherche dans le texte est courte.

La caractéristique majeure de l'algorithme Boyer-Moore est le fait qu'il compare les lettres du motif avec les lettres du texte de droite à gauche (à l'envers). Pour chercher 'Lazeyras' dans un texte, il faut commencer par regarder s'il y a un 's' en 8ème position, si c'est le cas, il faut vérifier s'il y a un 'a' en 7ème position sinon il faut sauter à la 16ème position. Ce saut accélère beaucoup l'exécution de l'algorithme.

Algorithmes

Avant de décrire l'algorithme, il faut présenter les deux tableaux construits à partir du motif. Leurs descriptions se fera à l'aide d'un exemple, le motif : 'ANPANMAN'.

Le premier tableau est celui du calcul de δ_1. Il donne pour chaque lettre le saut qui peut être effectué à droite s'il n'y a pas correspondance entre le motif et le texte.

Sur cet exemple, le tableau (`Tab1` ci-dessous) est donc le suivant :

lettre	A	N	P	M	autre
décalage	1	0	5	2	8

Le second tableau est celui du calcul de δ_2. Il donne pour chaque partie du motif déjà parcouru, le saut qui peut être effectué à droite s'il n'y a pas correspondance entre le motif et le texte.

Sur cet exemple, le tableau (`Tab2` ci-dessous) est donc le suivant :

Partie de motif	décalage
~~A~~N	8
~~M~~AN	3
~~N~~MAN	6
~~A~~NMAN	6
~~P~~ANMAN	6
~~N~~PANMAN	6
~~A~~NPANMAN	6
ANPANMAN	6

Algorithme de Boyer-Moore de recherche dans le texte (troisième partie) :

```
Boyer-Moore(T,M,Tab1,Tab2)
s := m
tant que s <= t faire
   j := m
   tant que j > 0 et T(s-m+j) = M(j) faire
      j := j-1
   fin faire
   si j = 0 alors
      imprimer le décalage, c.-à-d. la valeur de s-m
   fin si
   si j = m alors
      s := s + Tab1(T(s))
   sinon
      s := s + Tab2(m-j)
   fin si
fin faire
```

Exercice

Ex. 9 *Appliquer l'algorithme de Boyer-Moore afin de rechercher le motif : 'CONSCIENCE' dans le texte :*

ET SCIENCE SANS CONSCIENCE N'EST QUE RUINE DE L'AME

Construire les deux tableaux δ_1 et δ_2.

7 Exercice final

Ex. 10 *Soient les données suivantes :*

- *l'alphabet* $\Sigma = a, b$
- *le motif* $M = aabaab$
- *le texte* $T = aabaabaabaabaab$

Appliquer les 5 algorithmes aux données ci-dessus.
Parmi tous les algorithmes décrits ci-dessus, lesquels ne trouvent pas le bon nombre de décalages.