Recherche de motif dans un texte

Manon Ruffini

Soit Σ un alphabet. On considère le problème suivant :

Définition 1

> ***Entrée :*** *Un texte $t = t_1 \dots t_n$; un motif $x = x_1 \dots x_m$*
> ***Sortie :*** *Les positions auxquelles le motif x apparaît dans le texte t (rien s'il n'apparaît pas)*

Dans l'algorithme naïf, on teste toutes les positions, ce qui est couteux. Donc, on voudrait optimiser les décalages du motif. (on veut se servir des lettres qu'on a lues quand on cherchait si le motif correspondait)

1 Algorithme de Simon

On va construire l'automate du motif : lorsqu'on se trouve dans le q-ème état de l'automate, c'est qu'on vient de lire les q premières lettres du motif. Pour cela, on définit :

Définition 2 (*Fonction suffixe*)

> *C'est la fonction qui à un mot u associe le plus long suffixe de u qui est aussi préfixe de x :*
>
> $$\sigma : \left| \begin{array}{ccc} \Sigma^* & \longrightarrow & [|0, m|] \\ u & \longmapsto & \sigma(u) = \max\{k, x_1 \dots x_k \sqsupset u\} \end{array} \right.$$

Comme ε est suffixe de tout mot, la fonction σ est bien définie.

On définit l'automate du motif $\mathcal{A} = (Q, \Sigma, I, \delta, F)$ comme suit [1] [2] :
— $Q = [|0, m|]$ ensemble d'états
— $I = 0$ état initial
— $F = m$ état final
— Fonction transition : $\forall q \in Q, \forall a \in \Sigma, \delta(q, a) = \sigma(x_1 \dots x_q a)$
On construit l'automate en pré-traitement. On obtient l'algorithme suivant :

Algorithme 1 : Algorithme de Simon

Entrées : Un mot $t = t_1 \dots t_n$; un motif $x = x_1 \dots x_m$
Résultat : Les positions auxquelles apparaissent le motif x dans t
$A \leftarrow$ automate du motif x;
$q \leftarrow 0$;
pour $1 \leqslant i \leqslant n$ **faire**
  $q \leftarrow \delta(q, t_i)$;
  si $q = m$ **alors**
    Imprimer "position $i - m$"
fin

L'algorithme termine (facile) et est correct (long)

1. Parce qu'on veux pister le plus long préfixe de x ayant jusqu'ici concordé avec une portion de la chaîne t
2. On pourrait montrer que cet automate est déterministe, complet, et minimal.

L'algorithme a une complexité globale en : $O(n + m^3|\Sigma|)$. [3]

2 Algorithme de Knuth-Morris-Pratt

On veut améliorer l'algorithme précédent en faisant un pré-traitement moins coûteux. Pour ça, on va calculer une fonction préfixe π, stockée dans un tableau, qui exprime les correspondances entre le motif et ses décalages. En fait, pour chaque $q \in [|1, m|]$, on cherche le plus grand k tel que $x_1 \dots x_k$ soit suffixe de $x_1 \cdots x_q$. [4]

Algorithme 2 : Calcul de la fonction préfixe

Entrées : Un motif $x = x_1 \dots x_m$
Résultat : Un tableau $\pi[1, m]$ correspondant à la fonction préfixe de x
$\pi(1) \leftarrow 0$;
$k \leftarrow 0$;
pour $2 \leqslant q \leqslant m$ **faire**
  tant que $k > 0$ *et* $x_{k+1} \neq x_q$ **faire**
    $k \leftarrow \pi(k)$
  fin
  si $x_{k+1} = x_q$ **alors**
    $k \leftarrow k + 1$
  fin
  $\pi(q) \leftarrow k$
fin
retourner π

Exemple 1

On considère le motif $x = ababaca$ sur l'alphabet $\Sigma = \{a, b, c\}$. On obtient :

x	a	b	a	b	a	c	a
π	0	0	1	2	3	0	1

Puis on utilise l'algorithme de Knuth Morris et Pratt : Algorithme 3

3. Voir Algo 4
On pourrait même avoir : $O(m|\Sigma|)$
4. Sachant ce qu'on a lu du motif, on détermine son plus long préfixe propre qui est suffixe de la portion lue

Algorithme 3 : Knuth-Morris-Pratt

Entrées : Un texte $t = t_1 \dots t_n$; Un motif $x = x_1 \dots x_m$
Résultat : Les positions auxquelles le motif x apparaît dans le texte t
$\pi \leftarrow$ Préfixe(x);
$q \leftarrow 0$;
`/* q = nombre de caractères qui concordent */`
pour $1 \leqslant i \leqslant n$ **faire**
  tant que $q > 0$ *et* $x_{q+1} \neq t_i$ **faire**
    $q \leftarrow \pi(q)$
  fin
  si $x_{q+1} = t_i$ **alors**
    $q \leftarrow q + 1$
  fin
  si $q = m$ **alors**
    Imprimer "position $i - m$";
    $q \leftarrow \pi(q)$
  fin
fin
retourner π

On peut étudier la complexité (nombre de comparaisons de lettres) des algorithmes.

KMP On procède en comptant le nombre de tests positifs et le nombres de tests négatifs.

- Si le test est positif, on incrémente i. Or $1 \leqslant i \leqslant n$, donc il y a au plus n tests positifs.
- Si le test est négatif, i est inchangé mais q diminue strictement. Ainsi, $i - q$ augmente strictement. Or, cette quantité est inchangée dans le cas d'un test positif. Comme $i - q = 0$ au début, on obtient que le nombre de tests négatifs est inférieure à la valeur de $i - q$ à la fin de l'exécution. Or $i - q \leqslant i \leqslant n$, donc il y a au plus n tests négatifs.

Préfixe On raisonne de la même manière :

- Il y a au plus n tests positifs
- La quantité des tests négatifs est majoré par la valeur de $q - k$ à la fin de l'exécution, qui est elle-même majorée par m

La complexité de KMP est $O(m + n)$.

Complément : fonction transition (automate du motif)

Algorithme 4 : Fonction transition

pour $1 \leqslant q \leqslant m$ **faire**
  pour chaque *caractère* $a \in \Sigma$ **faire**
    $k \leftarrow \min(m + 1, q + 2)$;
    répéter
      $k \leftarrow k - 1$
    jusqu'à $x_1 \dots x_k \sqsupset x_1 \dots x_q a$;
    $delta(q, a) \leftarrow k$
  fin
fin

Références

[1] Thomas H. Cormen, *Algorithmique*. Dunod, 3è édition, 2010.