

Combinatoire des mots

Thierry Lecroq

Université de Rouen
FRANCE

Plan du cours

- 1 Rappels et notations
- 2 Périodes et bords
- 3 Recherche exacte de mot

Plan

- 1 Rappels et notations
- 2 Périodes et bords
- 3 Recherche exacte de mot

Rappels et notations

A alphabet (ensemble fini de lettres, symboles)

A^* ensemble des mots finis sur A

$|w|$ longueur du mot $w \in A^*$

ε mot vide, $|\varepsilon| = 0$

A^+ ensemble des mots finis non vides sur A

$w = uvv$

u préfixe

x facteur

v suffixe

Rappels et notations

Position

Pour un mot x de longueur m , on appelle **position** sur $x \in A^*$ tout entier i tel que $0 \leq i \leq m - 1$.

$$x = x[0]x[1] \dots x[m - 1]$$

$$x[i..j] = \begin{cases} x[i]x[i + 1] \dots x[j] & \text{si } i \leq j \\ \varepsilon & \text{sinon } (i > j) \end{cases}$$

Puissance

$$x^k = \underbrace{xxx \dots x}_{k \text{ fois}}$$

Exemple

$at^4 = atttt$ et $(at)^4 = atatat$

Rappels et notations

Occurrence

Soient $x, y \in A^*$, il y a une **occurrence** de x dans y si x est un facteur de y : autrement dit il existe une position j ($0 \leq j \leq n - m$) telle que

$$x = y[j..j + |x| - 1]$$

j position gauche

$j + |x| - 1$ position droite

Rappels et notations

Lemme de Lévi (extrait)

Soient $u, v, u', v' \in A^*$, si $uv = u'v'$ alors il existe $w \in A^*$ tel que
soit $u = u'w$ et $v' = wv$
ou $u' = uw$ et $v = wv'$.

Plan

- 1 Rappels et notations
- 2 Périodes et bords
- 3 Recherche exacte de mot

Périodes et bords

Périodes

Un entier p tel que $0 < p \leq |x|$ est une **période** d'un mot $x \in A^*$ si

$$x[i] = x[i + p]$$

pour $0 \leq i \leq |x| - p - 1$.

La période

On appelle **la période** et on note $pér(x)$ la plus petite des périodes de x .

Exemple

$x_1 = \text{atatata}$, 2, 4, 6 et 7 sont des périodes de x_1 et $pér(x_1) = 2$

$x_2 = \text{ataatata}$, 5, 7 et 8 sont des périodes de x_2 et $pér(x_2) = 5$

Proposition 1

Soient x un mot non vide et p un entier tel que $0 < p \leq |x|$. Les 5 propriétés suivantes sont équivalentes.

- 1 L'entier p est une période de x .
- 2 Il existe deux mots uniques $u, v \in A^*$ et un entier k tels que $x = (uv)^k u$ et $|uv| = p$.
- 3 Il existe un mot t et un entier ℓ tels que x est un préfixe de t^ℓ et $|t| = p$.
- 4 Il existe trois mots $u', v', w \in A^*$ tels que $x = u'w = wv'$ et $|u'| = |v'| = p$.
- 5 Il existe un mot z tel que x est un préfixe de zx et $|z| = p$.

Périodes et bords

Bords

Un mot w est un **bord** d'un mot x s'il est à la fois préfixe et suffixe de x ($w \neq x$).

$Bord(x)$ = plus long bord de x .

Exemple

$x_1 = \text{atatata}$, ε , a , ata et atata sont des bords de x_1 et

$Bord(x_1) = \text{atata}$

$x_2 = \text{ataatata}$, ε , a et ata sont des périodes de x_2 et $Bord(x_2) = \text{ata}$

Plan

- 1 Rappels et notations
- 2 Périodes et bords
- 3 Recherche exacte de mot

Recherche exacte de mot

Le problème de la recherche exacte d'un mot x de longueur m dans un texte y de longueur n consiste à localiser une ou plus généralement toutes les occurrences de x dans y .

Ce problème admet deux variantes :

- 1 Le mot x est connu à l'avance et peut subir un prétraitement. En général les solutions à cette variante ont une phase de prétraitement en $O(m)$ et une phase de recherche en $O(n)$.
- 2 Le mot y est connu à l'avance et peut subir un prétraitement. En général les solutions à cette variante ont une phase de prétraitement en $O(n)$ et une phase de recherche en $O(m)$.

On s'intéresse ici à la variante 1.

Notion de fenêtre glissante

Situation initiale

Situation générale

Situation finale

Notion de fenêtre glissante

Un algorithme de recherche de mot est donc une succession de

- **tentatives** (traitement consistant à comparer le mot et le contenu de la fenêtre) ;
- **décalages** (de la fenêtre vers la droite).

Algorithme naïf

Il est caractérisé par des décalages de longueur exactement 1.

LOCALISER-NAIVEMENT(x, m, y, n)

- 1 **pour** $j \leftarrow 0$ **à** $n - m$ **faire**
- 2 **si** $x = y[j..j + m - 1]$ **alors**
- 3 signaler une occurrence de x

Complexités

temps $O(mn)$

espace $O(1)$ (en plus de x et y)

Algorithme de Morris et Pratt (1970)

Considérons une tentative à la position gauche j :

on a reconnu un préfixe u de x dans y et on a une inégalité entre une lettre $x[i] = a$ dans le mot et une lettre $y[i + j] = b$ dans le texte.

Un décalage valide consiste à décaler la fenêtre de la période de

$$u = x[0..i - 1]$$

et de reprendre les comparaisons entre la lettre de x qui suit $Bord(u)$ soit $x[Bord(u)]$ et la lettre $y[i + j] = b$ de y .

Algorithme de Morris et Pratt (1970)

Soit la table *bon-préf* à $m + 1$ éléments définie comme suit

pour $0 \leq i \leq m$

$$\text{bon-préf}[i] = \begin{cases} -1 & \text{si } i = 0 \\ |\text{Bord}(x[0..i-1])| & \text{sinon.} \end{cases}$$

Algorithme de Morris et Pratt (1970)

LOCALISER-SELON-PRÉFIXE1(x, m, y, n)

```
1  $i \leftarrow 0$ 
2 pour  $j \leftarrow 0$  à  $n - 1$  faire
3   tantque  $i \geq 0$  and  $x[i] \neq y[j]$  faire
4      $i \leftarrow \text{bon-préf}[i]$ 
5    $i \leftarrow i + 1$ 
6   si  $i = m$  alors
7     signaler une occurrence de  $x$ 
8    $i \leftarrow \text{bon-préf}[i]$ 
```

Algorithme de Morris et Pratt (1970)

Théorème

L'algorithme LOCALISER-SELON-PRÉFIXE1(x, m, y, n) effectue au plus $2n - 1$ comparaisons.

Preuve

Il suffit de considérer la quantité $2j - i$.

Cette quantité croît d'au moins une unité après chaque comparaison :

- i et j sont incrémentés de 1 après chaque comparaison positive ;
- j reste inchangé après une comparaison négative alors que i décroît d'au moins une unité.

Valeur initiale $2j - i = 2 \times 0 + 0 = 0$

Valeur finale maximale $2j - i = 2(n - 1) - 0 = 2n - 2$

Donc au plus $2n - 1$ comparaisons sont effectuées.

Algorithme de Knuth, Morris et Pratt (1977)

Dans la situation générale

Si $c = a$ alors le résultat de la comparaison entre $y[i + j]$ et $x[\text{bon-préf}[i]]$ est connu à l'avance : il est négatif et la comparaison peut être évitée.

Il faut comparer $y[i + j]$ avec la lettre qui suit $\text{Bord}^k(u)$ tel que $k = \min\{\ell \mid x[|\text{Bord}^\ell(u)|] \neq a\}$.

Pour cela on définit la table *meil-préf* à $m + 1$ éléments de la manière suivante :

$$\text{meil-préf}[i] = \begin{cases} -1 & \text{si } i = 0 \\ |\text{Bord}(x[0..i-1])| & \text{si } x[|\text{Bord}(x[0..i-1])|] \neq x[i] \\ \text{meil-préf}[|\text{Bord}(x[0..i-1])|] & \text{sinon.} \end{cases}$$

Algorithme de Knuth, Morris et Pratt (1977)

LOCALISER-SELON-PRÉFIXE2(x, m, y, n)

```
1  $i \leftarrow 0$ 
2 pour  $j \leftarrow 0$  à  $n - 1$  faire
3   tantque  $i \geq 0$  and  $x[i] \neq y[j]$  faire
4      $i \leftarrow \text{meil-préf}[i]$ 
5    $i \leftarrow i + 1$ 
6   si  $i = m$  alors
7     signaler une occurrence de  $x$ 
8      $i \leftarrow \text{meil-préf}[i]$ 
```

Algorithme de Knuth, Morris et Pratt (1977)

Théorème

L'algorithme LOCALISER-SELON-PRÉFIXE2(x, m, y, n) effectue au plus $2n - 1$ comparaisons.

Preuve

Idem preuve de LOCALISER-SELON-PRÉFIXE1.

Calculs des tables *bon-préf* et *meil-préf*

BON-PRÉFIXE(x, m)

```
1 bon-préf[0] ← -1
2  $i \leftarrow 0$ 
3 pour  $j \leftarrow 1$  à  $m - 1$  faire
4   bon-préf[ $j$ ] ←  $i$ 
5   tantque  $i \geq 0$  and  $x[i] \neq x[j]$  faire
6      $i \leftarrow \text{bon-préf}[i]$ 
7    $i \leftarrow i + 1$ 
8 bon-préf[ $m$ ] ←  $i$ 
9 Retourner bon-préf
```

Calculs des tables *bon-préf* et *meil-préf*

MEILLEUR-PRÉFIXE(x, m)

```
1  meil-préf[0] ← -1
2   $i \leftarrow 0$ 
3  pour  $j \leftarrow 1$  à  $m - 1$  faire
4    si  $x[i] = x[j]$  alors
5       $\text{meil-préf}[j] \leftarrow \text{meil-préf}[i]$ 
6    sinon  $\text{meil-préf}[j] \leftarrow i$ 
7    faire
8       $i \leftarrow \text{meil-préf}[i]$ 
9    tantque  $i \geq 0$  and  $x[i] \neq x[j]$ 
10    $i \leftarrow i + 1$ 
11   $\text{meil-préf}[m] \leftarrow i$ 
12  Retourner meil-préf
```

Calculs des tables *bon-préf* et *meil-préf*

Théorème

Les algorithmes $\text{BON-PRÉFIXE}(x, m)$ et $\text{MEILLEUR-PRÉFIXE}(x, m)$ effectuent au plus $2m - 3$ comparaisons.

Preuve

Idem preuves de $\text{LOCALISER-SELON-PRÉFIXE1}$ et $\text{LOCALISER-SELON-PRÉFIXE2}$.

Valeur initiale $2j - i = 2 \times 1 + 0 = 2$

Valeur finale maximale $2j - i = 2(m - 1) - 0 = 2m - 2$

Donc au plus $2m - 3$ comparaisons sont effectuées.