INFORMATIQUE THÉORIQUE. — *Recherche linéaire d'un carré dans un mot.* Note (*) de **Max Crochemore,** présentée par Marcel Schützenberger.

On montre que la recherche d'un carré dans un mot peut être réalisée en temps proportionnel à la longueur du mot sur une machine à accès direct, pourvu que l'alphabet soit donné.

COMPUTER SCIENCE. — Linear Searching for a Square in a Word.

The search for a square in a word may be implemented in time proportional to the length of the word on a random access machine provided the alphabet is fixed.

Plusieurs méthodes ont été développées pour rechercher un carré dans un mot x([1], [2], [4]). Leur point commun est leur complexité : $O(|x|\log_2|x|)$. On peut montrer que si l'algorithme pour effectuer cette recherche doit fonctionner indépendamment de l'alphabet du mot x, ces méthodes sont optimales. On montre ici que la situation change si l'alphabet est connu à l'avance. La technique principale que nous utilisons est le fait que, sur un alphabet donné, les suffixes d'un mot peuvent être ordonnés suivant l'ordre lexicographique en temps linéaire ([5], [7]). Cette voie avait déjà été explorée dans [1].

1. Définitions. — Les mots que l'on considère sont éléments du monoïde libre A^* engendré par un ensemble fini A, l'alphabet. On note 1 le mot vide, $A^+=A^*-1$ et $|x|$ la longueur d'un mot x de A^*.

Un mot x *contient un carré* yy si $y\in A^+$ et x peut s'écrire $x_1 yyx_2$ avec $x_1, x_2\in A^*$. Dans le cas contraire x est dit *sans carré*. En particulier le mot vide est sans carré.

Quand $x=yz$, y est dit *préfixe* de x et z suffixe de x. A chaque décomposition yz d'un mot x on associe son *séparateur* noté $\mathrm{sep}(y, z)$. C'est le plus grand couple de mots (v, w) qui vérifie $|v|<|y|$ si $w\neq 1$, vw est préfixe de yz et yw est préfixe de yz; l'ordre des couples est défini par :

$$(v', w')\leqq(v, w)\leftrightarrow|w'|<|w| \quad \text{ou} \quad (|w'|=|w| \quad \text{et} \quad |v'|\geqq|v|).$$

Le mot w est donc le plus long préfixe de z qui apparaît au moins deux fois dans le mot yw. Le mot v repère la première occurrence de w dans yw.

Exemple 1. — Si a est une lettre qui n'apparaît pas dans y, pour un mot quelconque z, $\mathrm{sep}(y, az)=(1, 1)$. En particulier pour un mot x, $\mathrm{sep}(1, x)=\mathrm{sep}(x, 1)=(1, 1)$. ■

Exemple 2. — Sur l'alphabet a, b, c, d on considère le mot :

$$x_0=abcdbabdcadbcbdbabcabcb.$$

On a sep $(abcdbabdcadbcbdbabc, abcb)=(1, abc)$. ■

Nous définissons la s-factorisation d'un mot à partir des séparateurs qui lui sont associés. Formellement, la s-factorisation de x est une suite de mots non vides $(u_1, \ldots, u_k)$ telle que :

$$x=u_1 u_2 \ldots u_k,$$

et qui se construit de façon itérative; si $x=u_1 u_2\ldots u_{i-1} az$, avec $a\in A$ et $z\in A^*$, et si sep $(u_1\ldots u_{i-1}, az)=(v, w)$, on pose $u_i=w$ si $w\neq 1$ et $u_i=a$ sinon. Cette factorisation est unique et au mot vide correspond la suite vide. On peut noter que u_1 est réduit à la première lettre de x.

Exemple 2 (suite). — La s-factorisation de x_0 est :

$$(a, b, c, d, b, ab, d, c, a, db, c, bd, bab, ca, bcb). \quad ■$$

On introduit, pour finir, les propriétés *droite* et *gauche* sur les couples de mots. Un couple(u, u') satisfait la propriété droite, notée $d(u, u')$, si u et u' sont sans carré et :

$$\exists u_1, u_2, u_3, u_4 \in A^*, \quad u_2 u_3 \neq 1, \qquad u = u_1 u_2 \qquad \text{et} \qquad u' = u_3 u_2 u_3 u_4.$$

La propriété $g(u, u')$ est vraie si on a $d(\bar{u}', \bar{u})$ où $\bar{u}'$ et $\bar{u}$ sont les images miroirs de u' et u respectivement.

Dire que (u, u') satisfait la propriété droite signifie donc que u, u' sont sans carré et que uu' contient un carré centré à droite de u.

Exemple 3. — On a $d(bab, cabcb)$ avec $u_1 = b$, $u_2 = ab$, $u_3 = c$ et $u_4 = b$. Le mot *babcabcb* contient le carré *abcabc*. ■

2. Résultats. — Les séparateurs à eux seuls ne permettent pas de caractériser les mots contenant un carré, bien qu'ils fournissent une condition suffisante éconcée dans le lemme qui suit. La s-factorisation est utilisée pour combler cette lacune; les conditions données dans le théorème servent à guider la recherche d'un carré dans un mot et sont à la base de notre algorithme.

Lemme. — *Soit* $x = a_1 \ldots a_n$ *un mot de* A^* *où les* a_i *sont des lettres de* A. *Ce mot contient un carré si :*

$$(1) \qquad \begin{cases} \exists i \in (1, \ldots, n), \quad \exists v, w \in A^* \ (v, w) = \operatorname{sep}(a_1 \ldots a_{i-1}, a_i \ldots a_n), \\ \qquad w \neq 1 \qquad \textit{et} \qquad a_1 \ldots a_{i-1} \ \textit{préfixe de } vw. \end{cases}$$

Preuve. — x contient un carré car deux occurrences de w se chevauchent ou sont consécutives dans le mot yw. ■

Théorème 1. — *Soit* $x = a_1 \ldots a_n$ *un mot de* A^*, *et* $(u_1, \ldots, u_k)$ *sa s-factorisation. Le mot* x *contient un carré si et seulement s'il satisfait* (1) *ou :*

$$(2) \qquad \exists i \in (1, \ldots, k-1) \quad g(u_i, u_{i+1}) \text{ ou } d(u_i, u_{i+1}) \text{ ou } d(u_1 \ldots u_{i-1}, u_i u_{i+1}).$$

Exemple 2 (suite). — Le mot x_0 contient le carré *abcabc*. Il ne vérifie pas (1) mais (2) car on a (*voir* exemple 3) :

$$d(abcdbabdcadbcbdbab, cabcb).$$

Preuve du théorème. — On montre que si x contient un carré et ne satisfait pas (1), il vérifie (2).

Soit $x_1 yy$ le plus court préfixe de x qui contient un carré. Soit i le premier entier pour lequel $x_1 yy$ est préfixe de $u_1 \ldots u_{i+1}$ (on a $i \geqq 1$).

Le mot $u_1 \ldots u_i$ est donc sans carré. Par définition de u_{i+1} et puisque (1) n'est pas vérifiée, u_{i+1} est contenu dans $u_1 \ldots u_i$ et donc est lui aussi sans carré.

En utilisant la définition de u_i on déduit que $u_1 \ldots u_{i-1}$ (éventuellement vide) est préfixe de $x_1 y$. Dans le cas contraire u_i serait contenu dans la seconde occurrence de y sans en être suffixe, ce qui contredirait la maximalité de sa longueur.

La seconde occurrence de y est donc contenue dans $u_i u_{i+1}$. Si $g(u_i, u_{i+1})$ ni $d(u_i, u_{i+1})$ ne sont vérifiées on a $d(u_1 \ldots u_{i-1}, u_i u_{i+1})$. ■

3. ALGORITHME. — Du théorème précédent se déduit immédiatement un algorithme pour vérifier si un mot contient un carré. Nous écrivons cet algorithme sous la forme d'une fonction dont la valeur est « vrai » uniquement quand le mot d'entrée contient un carré :

fonction carré $(a_1 \ldots a_n)$;
$i \leftarrow 1$;
tant que $i \leqq n$ *faire*
  $(v, w) \leftarrow \text{sep}\,(a_1 \ldots a_{i-1}, a_i \ldots a_n)$;
  si $vw=1$ *alors* $u' \leftarrow a_i$; $j \leftarrow i$; $i \leftarrow i+1$
  sinon $u \leftarrow u'$; $u' \leftarrow a_i \ldots a_{i+|w|-1}$;
    si$(|vw| \geqq i-1$ ou $g(u, u')$ ou $d(u, u')$ ou $d(a_1 \ldots a_{j-1}, uu'))$
    alors retour ('vrai') *fin de si*;
    $j \leftarrow i$; $i \leftarrow i+|w|$
  fin de si
fin de tant que; retour ('faux')
fin de fonction.

L'algorithme tient compte de la remarque suivante :

Remarque. — Si $a_1 \ldots a_{i-1}$ est sans carré et si sep $(a_1 \ldots a_{i-1}, a_i \ldots a_n)=(1, 1)$, la lettre a_i n'apparaît pas dans $a_1 \ldots a_{i-1}$ et $a_1 \ldots a_i$ est sans carré.

4. TEMPS DE CALCUL. — Le nombre d'opérations nécessaire au calcul de la fonction carré repose sur celui des fonctions sep, d et g.

Pour un alphabet donné, il est possible d'effectuer le calcul des séparateurs de toutes les décompositions yz d'un mot x en temps linéaire [5]. Il reste à examiner le cas des fonctions g et d. Pour l'évaluation de celles-ci, on peut se servir d'un algorithme classique utilisé dans les problèmes de « string-matching ».

Pour un mot x non vide on note $f(\omega)$ le plus long mot distinct de x qui en est à la fois préfixe et suffixe. Un algorithme de [3] permet le calcul de $f(u)$ pour tous les préfixes non vides d'un mot x en temps linéaire.

PROPOSITION 1. — *Soient deux mots u et u' qui vérifient $d(u, u')$ et soit psp le plus court préfixe de u' tel que s soit suffixe de u. Alors $p=f(psp)$.*

PROPOSITION 2. — *Mêmes hypothèses. Si $p' vs' p'$ est préfixe de u' avec $|psp|<|p' vs' p'|$ et s' le plus long suffixe commun à u et $p' vs'$, on a $|ps|<\max(|p' v|, |p' vs'|/2)$.*

Pour évaluer $d(u, u')$ avec deux mots sans carré u et u', on évalue f en lisant u' de la gauche vers la droite, puis de la droite vers la gauche connaissant psp un préfixe de u' avec $p=f(psp)$ on examine si s est suffixe de u. Le nombre de comparaisons de lettres, représentatif du temps de calcul, lors de cet examen est majoré par $2|u'|$ comme conséquence de la proposition 2.

COROLLAIRE. — *Si u et u' sont sans carré, l'évaluation de $d(u, u')$ [resp. $g(u, u')$] prend un temps $O(|u'|)$ [resp. $O(|u|)$].*

THÉORÈME 2. — *Sur un alphabet donné, la recherche d'un carré dans un mot x prend un temps $O(|x|)$ sur une machine à accès direct.*

5. CONCLUSION. — Quand l'alphabet n'est pas connu, il est néanmoins possible d'utiliser l'algorithme du 3. Cependant la réalisation de la fonction sep sera différente et devra pour être performante utiliser les techniques d'arbres de recherche ou de « hash-table » pour travailler sur l'ensemble des lettres du mot. Le temps de calcul est modifié et devient $O(|x| \log_2 (\text{card}\, A))$ où A est l'ensemble des lettres du mot x [5].

Il semble que l'algorithme suggéré dans [6] permette de vérifier si un mot contient un carré en temps linéaire. L'absence de détail sur le fonctionnement de l'algorithme, et de preuve, empêche la vérification du résultat.

(*) Remise le 2 mars 1983.

[1] A. Apostolico et F. P. Préparata, *Theor. Compt. Sc.*, 22, 1983, p. 297-315.

[2] M. Crochemore, *Information Processing Letters*, 12, 1981, p. 244-250.

[3] D. E. Knuth, J. H. Morris et V. R. Pratt, *S.I.A.M. J. Comput.*, 6, 1977, p. 323-350.

[4] M. Main et R. Lorentz, *An O (n log n) Algorithm for Finding Repetition in a String CS*-79-056, Washington State University, Pullman, WA, 1979.

[5] E. M. McCreight, *J. of the A.C.M.*, 23, n° 2, 1976, p. 262-272.

[6] A. O. Slisenko, *Soviet. Math. Dokl.*, 21, n° 2, 1980, p. 392-395.

[7] P. Wiener, *Linear Pattern Matching Algorithms*, in *Proceedings of the* 14*th Annual Symposium on Switching and Automata Theory*, 1973, p. 1-11.

Laboratoire d'Informatique, Université de Haute-Normandie,
B. P. n° 67, 76130 *Mont-Saint-Aignan.*