Informatique I

La notation tiendra compte de la rigueur des raisonnements et de la clarté des explications. Chaque question peut être traitée en admettant le résultat des questions précédentes.

Le problème s'intéresse à la recherche de motifs dans un texte et à quelques applications possibles.

Notations et définitions

- A est un alphabet supposé connu, contenant au moins deux lettres.
- A^* est l'ensemble des mots sur l'alphabet A. Dans la suite, t, u, v, w, x, y, z désignent des mots.
- $|u|$ désigne la longueur d'un mot u de A^*.
- On note ε le mot vide. On a donc $|\varepsilon| = 0$.
- $|X|$ est le cardinal d'un ensemble fini X.
- On dit que u est *facteur* de v si $v = u_1 u u_2$, avec $u_1, u_2 \in A^*$.
- Si $v = u_1 u u_2$ et $|u_1| = k - 1$, on dit que u a une *occurrence* dans v en *position* k.
- On dit que u est *préfixe* de v si $v = uu'$, avec $u' \in A^*$.
- On dit que u est *suffixe* de v si $v = u'u$, avec $u' \in A^*$.

La complexité en temps des algorithmes sera évaluée en nombre de comparaisons de lettres.

I Algorithmes de recherche

Dans toute cette partie, on s'intéresse au problème RECHERCHE-MOTIF suivant :

Donnée : – deux entiers m et n,
– un mot $x = x_1 \cdots x_m$ ($x_i \in A$) appelé le *motif*,
– un mot $t = t_1 \cdots t_n$ ($t_i \in A$) appelé le *texte*.

Question : Le mot x est-il facteur du mot t ? Si oui, quelle est la position de la première occurrence de x dans t ?

Un algorithme « naïf » consiste à tester l'égalité $x_1 \cdots x_m = t_k \cdots t_{k+m-1}$ pour k allant de 1 à $n - m + 1$. Chacun de ces tests se fait en comparant les mots lettre à lettre de gauche à droite. Durant le test $x_1 \cdots x_m = t_k \cdots t_{k+m-1}$, l'entier k est appelé la *position de comparaison* du motif.

Lorsqu'un test en position de comparaison k est négatif sur la $i^{\text{ème}}$ lettre de x ($x_i \neq t_{k+i-1}$), les algorithmes que l'on va considérer effectuent un *décalage* d du motif : la nouvelle position de comparaison est $k + d$. Dans l'algorithme naïf, le déplacement d vaut toujours 1, et le test $x_1 \cdots x_m = t_\ell \cdots t_{\ell+m-1}$ ($\ell = k + 1$ étant la nouvelle position de comparaison) reprend au début

des deux mots. La figure 1 montre comment l'algorithme décale le motif en cas d'échec d'un des tests $x_i = t_{k+i-1}$.

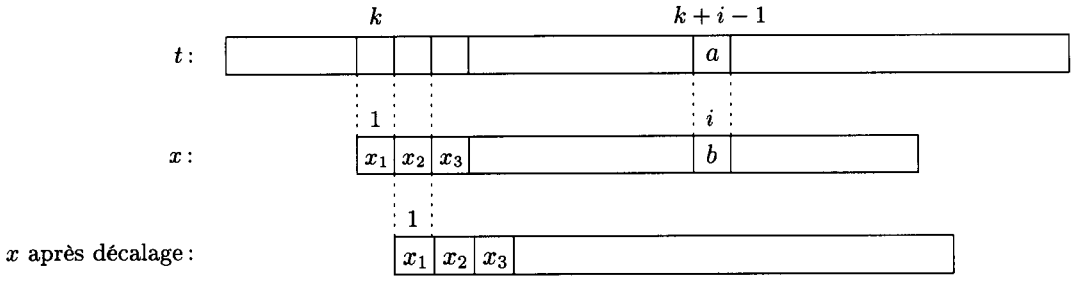


FIG. 1: *Décalage du motif en cas d'échec du test $x_i = t_{k+i-1}$ dans l'algorithme naïf*

La fonction de la figure 2 donne une version de cet algorithme. S'il n'y a pas d'occurrence de x dans t, la fonction calcule la valeur -1. Sinon, elle calcule la position de la première occurrence de x dans t. Dans ce programme, la première case d'un tableau est numérotée 1. Ainsi, on accède à la $i^{\text{ème}}$ lettre de x par `x[i]`. De plus, les tests sont évalués de façon paresseuse : par exemple, dans un test `c1 et c2`, l'expression `c2` n'est évaluée que si l'expression `c1` est vraie.

```
1    fonction recherche_naïve (x, t, m, n)
2        k := 1;
3        tantque k <= n-m+1 faire
4            i := 1;
5            tantque i <= m et x[i] = t[k+i-1] faire i := i+1 fintantque
6            si i = m+1 alors retourner k finsi
7            k := k+1;
8        fintantque
9        retourner -1;
```

FIG. 2: *Une fonction de recherche de motif dans un texte*

I.1 Calculer le nombre de comparaisons de cet algorithme dans le cas le pire en fonction de m et n. Donner un exemple de mots x et t pour lesquels le cas le pire est réalisé.

I.2 On rappelle qu'il est supposé que $|A| \geqslant 2$. On suppose que chaque lettre de l'alphabet a une probabilité $\frac{1}{|A|}$ d'apparaître à chaque position (dans le texte comme dans le motif). Calculer le nombre moyen de comparaisons de lettres faites lors de la comparaison de x à $t_k \cdots t_{k+m-1}$. Montrer que ce nombre est plus petit que 2. En déduire que l'algorithme effectue moins de $2n$ comparaisons en moyenne.

I.3 On cherche maintenant à améliorer les performances de l'algorithme dans le cas le pire. On fixe une position de comparaison k dans t, et on suppose que $x_1 \cdots x_m \neq t_k \cdots t_{k+m-1}$. L'*indice d'échec* $i_e(k)$ en position de comparaison k est le plus petit entier i tel que $x_i \neq t_{k+i-1}$. Un

décalage d est *inutile* en position de comparaison k lorsque $d < i_e(d) - 1$ et $x_1 \cdots x_{i_e(d)-d-1} \neq t_{k+d} \cdots t_{k+i_e(d)-2}$. Les autres décalages sont dits *utiles*.

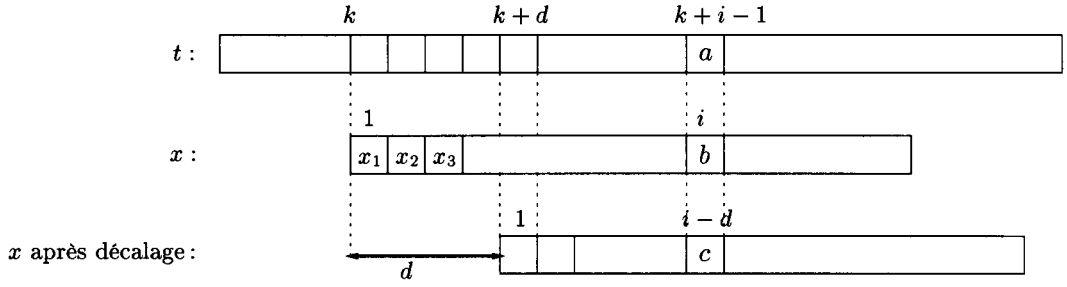


FIG. 3: *Décalage d du motif*

Montrer que si d est un décalage inutile en position de comparaison k, il n'y a pas d'occurrence de x en position $k + d$.

Notation et définition

- On dit que u est un *bord* de v si $u \neq v$ et si u est à la fois préfixe et suffixe de v.
- Pour $v \in A^*\backslash\{\varepsilon\}$, Bord($v$) est ***le** bord maximal* de v, c'est-à-dire le bord le plus long de v.

Ainsi par exemple, le mot *ababa* admet comme bords *aba*, *a* et ε. Le bord maximal est *aba*. On notera que dans cet exemple, le préfixe *aba* et le suffixe *aba* se chevauchent dans *ababa*. Il n'en est pas de même pour le bord maximal de *abcab*, qui est *ab*.

I.4 On note p_i le préfixe de longueur i du motif x, pour $i = 0, \ldots, |x|$. Donner une condition nécessaire portant uniquement sur les p_j et ne faisant pas intervenir t pour que le décalage d soit utile. Calculer le plus petit décalage utile $d_u(k)$ en fonction de x et $i_e(k)$.

I.5 Après un décalage utile d (cf. figure 3), à quel indice $j(d)$ dans le mot x peut-on reprendre la comparaison $x_1 \cdots x_m = t_{k+d} \cdots t_{k+d+m-1}$?

I.6 On définit une fonction f_x qui va de $\{0, \ldots, m\}$ dans $\{-1, \ldots, m-1\}$ de la façon suivante :

$$f_x(0) = -1$$
$$f_x(i) = |\text{Bord}(p_i)|$$

En supposant que l'on dispose de la fonction f_x, modifier la fonction donnée en figure 2 de façon à effectuer à chaque fois le plus petit décalage utile et à reprendre la comparaison $x_1 \cdots x_m = t_{k+d} \cdots t_{k+d+m-1}$ à l'indice $j(d_u(k))$.

I.7 On dit qu'une comparaison de lettres de la forme `x[i]=t[j]` effectuée par l'algorithme est un *test positif* si $x_i = t_j$, et un *test négatif* sinon. En considérant dans la fonction écrite en I.6 la variation des quantités $k+i$ et k après une comparaison de lettres, majorer le nombre de tests positifs et le nombre de tests négatifs. Montrer que l'algorithme fait toujours moins de $2n-m$ comparaisons de lettres dans le cas le pire si l'on dispose de la fonction f_x. Montrer par un exemple que cette borne est atteinte.

I.8 Calculer le nombre maximal de comparaisons qu'une lettre du texte peut subir avant que l'on passe à la lettre suivante.

I.9 On veut maintenant montrer que l'on peut calculer la fonction f_x en temps $O(m)$. Pour une fonction g, on note g^k la fonction $\underbrace{g \circ \cdots \circ g}_{k \text{ fois}}$ (quand elle existe).

a) Soit u un mot non vide. Comme la fonction Bord fait décroître la longueur, il existe un entier q tel que $\text{Bord}^q(u) = \varepsilon$. Soit p le plus petit entier tel que $\text{Bord}^p(u) = \varepsilon$. Montrer que l'ensemble des bords de u est $\mathscr{B}(u) = \{\text{Bord}(u), \text{Bord}^2(u), ..., \text{Bord}^p(u)\}$.

b) Soit a une lettre et $u \in A^*$ non vide. Montrer que $\text{Bord}(ua)$ est le plus long préfixe de u qui est dans l'ensemble $\mathscr{B}(u,a) = \{\text{Bord}(u)a, \text{Bord}^2(u)a, ..., \text{Bord}^p(u)a, \varepsilon\}$.

c) En déduire que

$$\text{Bord}(ua) = \begin{cases} \text{Bord}(u)a & \text{si Bord}(u)a \text{ est préfixe de } u \\ \text{Bord}(\text{Bord}(u)a) & \text{sinon} \end{cases}$$

d) Soit $j \in \{1, \ldots, m\}$, et $g = f_x$. Montrer que $g(j) = g^k(j-1)+1$, où $k \geqslant 1$ est le plus petit entier tel que l'on ait soit $g^k(j-1)+1 = 0$, soit $g^k(j-1)+1 \neq 0$ et $x_{g^k(j-1)+1} = x_j$.

e) En déduire un algorithme qui calcule tous les $f_x(j)$ pour $j = \{1, \ldots, m\}$ en temps $O(m)$.

I.10 En déduire un algorithme qui résout le problème RECHERCHE-MOTIF en temps $O(m+n)$.

II Applications

Dans cette partie, on pourra utiliser les résultats des questions I.9 et I.10.

II.1 Soit $x \in A^*$. Un facteur y de x est *répété* s'il a deux occurrences distinctes dans x. Ces occurrences peuvent se chevaucher (par exemple dans *abcabca*, le facteur *abca* est répété) ou non (par exemple dans ce même mot, *a* et *bca* sont répétés). Montrer que l'on peut déterminer le plus long préfixe répété d'un mot x en temps $O(|x|)$.

II.2 Un mot $x \neq \varepsilon$ est dit *primitif* s'il n'est pas de la forme y^n avec $n \geqslant 2$. Par exemple *abac* est primitif alors que $abab = (ab)^2$ n'est pas primitif. Montrer qu'un mot x est primitif si et seulement les seules occurrences de x dans xx sont celle commençant en position 1 et celle commençant en position $|x|+1$. Montrer que l'on peut déterminer en temps $O(|x|)$ si x est primitif.

II.3 Deux mots x et y sont *conjugués* s'il existe $u, v \in A^*$ tels que $x = uv, \quad y = vu$. Montrer que l'on peut déterminer en temps $O(n)$ si deux mots de longueur n sont conjugués.

II.4 L'*image miroir* d'un mot $u = u_1 u_2 \cdots u_p$ ($u_i \in A$) est le mot $\tilde{u} = u_p \cdots u_2 u_1$, c'est-à-dire le mot dont la suite des lettres est celle de u lue « à l'envers ». Un mot u est appelé un *palindrome* si $u = \tilde{u}$. Montrer qu'étant donné un mot x, on peut calculer le plus long préfixe de x qui est un palindrome en temps $O(|x|)$.

II.5 Un *carré* est un mot de la forme yy avec $y \neq \varepsilon$.

a) Montrer que x admet un préfixe qui est un carré si et seulement s'il existe $j \geqslant 2$ tel que $f_x(j) \geqslant j/2$.

b) Montrer que l'on peut déterminer si un mot x contient un carré comme facteur en temps $O(|x|^2)$.

III Une distance d'édition

On dit que l'on passe d'un mot y à un mot z par

- une insertion si $y = st$ et $z = sat$, avec $s, t \in A^*$, $a \in A$.
- une suppression si $y = sat$ et $z = st$, avec $s, t \in A^*$, $a \in A$.

On définit la fonction suivante d de $A^* \times A^*$ dans $\mathbf{N}$: $d(y, z)$ est le plus petit entier k tel qu'il existe une suite de $k+1$ mots $w_0, \ldots, w_k$ satisfaisant $y = w_0$, $z = w_k$, et telle que l'on passe de w_i à w_{i+1} par une insertion ou une suppression.

III.1 Montrer que d est une distance sur A^*, c'est-à-dire que d est positive et que l'on a

a) $d(y, z) = 0 \iff y = z$, b) $d(y, z) = d(z, y)$, c) $d(y, z) \leqslant d(y, t) + d(t, z)$

III.2 Une *correspondance* f entre deux mots $y = y_1 \cdots y_p$ ($y_i \in A$) et $z = z_1 \cdots z_q$ ($z_i \in A$) est une fonction partielle de l'ensemble $Y = \{1, \ldots, p\}$ dans l'ensemble $Z = \{1, \ldots, q\}$ telle que

- f strictement croissante: si $i_1 < i_2$ et si $f(i_1)$, $f(i_2)$ sont définis, alors $f(i_1) < f(i_2)$.
- si $f(i)$ est défini, alors $y_i = z_{f(i)}$.

On peut voir une correspondance graphiquement, comme sur la figure 4.



FIG. 4: *La correspondance* $1 \mapsto 1$, $2 \mapsto 4$, $3 \mapsto 7$, $8 \mapsto 8$ *entre* recherche *et* research

Soit f une correspondance et $G(f) = \{(i, f(i)) \mid i \in Y \text{ et } f(i) \text{ est défini}\}$ son graphe. On définit

$$I(f) = \{i \in Y \mid \forall j \in Z,\ (i, j) \notin G(f)\}$$
$$J(f) = \{j \in Z \mid \forall i \in Y,\ (i, j) \notin G(f)\}$$

Le *coût* de f est l'entier $c(f) = |I(f)| + |J(f)|$. Sur l'exemple de la figure 4, ce coût est $|\{4,5,6,7,9\}| + |\{2,3,5,6\}| = 9$.

a) Soit g (resp. h) est une correspondance entre u et v (resp. entre v et w). Montrer que la fonction $h \circ g$ de $\{1,\ldots,|u|\}$ dans $\{1,\ldots,|w|\}$ est une correspondance entre u et w telle que $c(h \circ g) \leqslant c(h) + c(g)$.

b) Montrer que $d(y,z)$ est égal au coût minimal d'une correspondance entre y et z.

III.3 On va maintenant établir un algorithme pour calculer d.

a) Soient y, z deux mots et a, b deux lettres. Montrer que

$$d(ya, zb) = \min\big[d(y,zb)+1,\ d(ya,z)+1,\ d(y,z)+2\delta(a \neq b)\big]$$

où $\delta(P)$ vaut 1 si la propriété P est vraie et 0 sinon.

b) Décrire un algorithme pour calculer $d(y,z)$ en temps $O(|y||z|)$. Montrer qu'on peut se contenter de $O(\min(|y|,|z|))$ emplacements en mémoire pour stocker des valeurs intermédiaires durant le calcul.

III.4 On rappelle que u est un *sous-mot* de v si l'on peut écrire $u = u_1 \cdots u_k$ et $v = u'_0 u_1 u'_1 \cdots u_k u'_k$, avec $u_i \in A$ et $u'_i \in A^*$, *i.e.*, si la suite des lettres de u est une suite extraite de la suite des lettres de v.

Soient y et z deux mots. Montrer que l'on peut calculer la longueur du plus long sous-mot commun à y et z en temps $O(|y||z|)$ et en utilisant $O(\min(|y|,|z|))$ emplacements en mémoire pour stocker des valeurs intermédiaires durant le calcul.

IV Plus long facteur commun, plus long sous-mot commun

La définition d'un sous-mot est donnée en III.4. On considère les problèmes PLFC (Plus Long Facteur Commun) et PLSMC (Plus Long Sous-Mot Commun) suivants :

PLFC **Donnée :** Un alphabet A fini, un ensemble E de mots de A^*, et un entier $k \geqslant 1$.
Question : Existe-t-il un facteur commun w à tous les mots de E de longueur $|w| \geqslant k$?

PLSMC **Donnée :** Un alphabet A fini, un ensemble E de mots de A^*, et un entier $k \geqslant 1$.
Question : Existe-t-il un sous-mot commun w à tous les mots de E de longueur $|w| \geqslant k$?

IV.1 Donner un algorithme polynomial pour résoudre PLFC.

IV.2 Montrer que PLSMC est dans la classe NP.

Étant donné un graphe non orienté $\mathscr{G} = (\mathscr{S}, \mathscr{A})$, l'ensemble $\mathscr{S} = \{s_1,\ldots,s_p\}$ désigne son ensenble de sommets, et l'ensemble $\mathscr{A}$ son ensemble d'arêtes. On notera $\{s_i, s_j\}$ une arête entre les deux sommets s_i et s_j, avec la convention $i < j$.

On considère le problème suivant, que l'on note CS (Couverture de Sommets). On sait que le problème CS est NP-complet :

CS **Donnée :** Un entier ℓ et un graphe non orienté $\mathscr{G} = (\mathscr{S}, \mathscr{A})$.

Question : Existe-t-il un sous-ensemble $\mathscr{S}' \subseteq \mathscr{S}$ de sommets tel que

- $|\mathscr{S}'| = \ell$, et
- pour chaque arête $\{u, v\}$ de $\mathscr{A}$ on a $u \in \mathscr{S}'$ ou $v \in \mathscr{S}'$ (ou les deux).

Soit $(\ell, \mathscr{G})$ une instance de CS. On note $\mathscr{S} = \{s_1, \ldots, s_p\}$ l'ensemble des sommets de $\mathscr{G}$, et $\mathscr{A} = \{a_1, \ldots, a_q\}$ l'ensemble de ses arêtes. On construit une instance (A, E, k) de PLSMC de la façon suivante.

- L'alphabet A est $\mathscr{S} = \{s_1, \ldots, s_p\}$.
- L'ensemble de mots E sur l'alphabet A contient $|\mathscr{A}| + 1$ mots : $E = \{u, u_1, \ldots, u_q\}$ où :

$$u = s_1 \cdots s_p,$$

$$u_r = \underbrace{s_1 \cdots s_{i-1} s_{i+1} \cdots s_p}_{\text{tous les sommets sauf } s_i} \cdot \underbrace{s_1 \cdots s_{j-1} s_{j+1} \cdots s_p}_{\text{tous les sommets sauf } s_j} \text{ pour chaque arête } a_r = \{s_i, s_j\} \ (i < j).$$

- $k = |\mathscr{S}| - \ell$.

IV.3 Montrer que le graphe $\mathscr{G}$ a une couverture de sommets de taille ℓ si et seulement si les mots de l'ensemble E ont un sous-mot commun de longueur k.

IV.4 Montrer que la fonction qui à l'instance $(\ell, \mathscr{G})$ de CS associe l'instance (A, E, k) de PLSMC définit une réduction polynomiale de CS à PLSMC.

IV.5 Que peut-on déduire des questions précédentes ?