Table des matières

Ce problème comporte des questions nécessitant un code Caml. Pour ces questions, les réponses ne feront pas appel aux fonctionnalités impératives de langage (en particulier pas de boucles, pas de références).

On considère ici un alphabet totalement ordonné de k symboles, noté Σ.

1 Mots de Lyndon

Le type Caml choisi pour représenter un mot est une chaîne de caractères (`string`). Les éléments de Σ sont représentés par le type `char`.

Dans toute la suite, les seules fonctions/méthodes Caml sur les chaînes de caractères qui peuvent être utilisées sont :

- `S.[i]` (valeur du i e caractère) ;
- l'opérateur de concaténation `^` ;
- la fonction `String.length : string -> int` ;
- la fonction `String.sub : string -> int -> int -> string`. `String.sub s start l` retourne une chaîne de longueur l contenant la sous-chaîne de s qui commence en `start`.

Définition 1.1. *(Préfixe, suffixe)*
Soit $(m,p) \in (\Sigma^*)^2$.

- $mp = m_0 ... m_{|m|-1} p_0 ... p_{|p|-1}$ *est le concaténé de* m *et* p ;
- m *est un préfixe de* p *si* $|m| < |p|$ *et* $m_i = p_i$ *pour* $0 \leq i \leq |m|-1$;
- m *est un suffixe de* p *si* $|m| < |p|$ *et* $m_i = p_{|p|-|m|+i}$ *pour* $0 \leq i \leq |m|-1$.

On définit alors la relation $\prec$ par :

$$m \prec p \Leftrightarrow (m \text{ est un préfixe de } p) \text{ ou } (\text{Il existe } k \in [\![0, |m|[\![\text{ tel que pour tout } i \in [\![0,k[\![,\ m_i = p_i \text{ et } m_k < p_k)$$

1. On donne le type

```
type comparaison = Inferieur | Egal | Superieur ;;
```

Écrire une fonction recursive `ordre : string -> string -> comparaison` telle que `ordre m p` est l'ordre relatif des mots m et p.

Définition 1.2. *(Ordre lexicographique).*
La relation définie par : $m \preceq p$ *si et seulement si* $m = p$ *ou* $m \prec p$ *est appellée « order lexicographique » sur* Σ^*.

Définition 1.3. *Soit* $m \in \Sigma^n$. *Un conjugué de* m *est un mot de la forme* $m_i \cdots m_{n-1} m_0 \cdots m_{i-1}$ *, pour* $i \in [\![0,n[\![$. *Par convention, le conjugué de* m *pour* $i = 0$ *est le mot* m *lui-même.*

La notion de conjugaison induit une relation C définie sur Σ^* par mCp si et seulement si p est un conjugué de m.

2. Écrire une fonction Caml `conjugue : string -> int -> string` telle que `conjugue m i` renvoie le conjugué de m débutant par le i-ème caractère de m, $0 \leq i \leq |m|-1$.
solution :

```
let conjugue m i =
  let l = String.length m in
  let sous_chaine1 = String.sub m 0 i
  and sous_chaine2 = String.sub m i (l-i) in
  sous_chaine2 ^ sous_chaine1
;;
```

3. Montrer que C est une relation d'équivalence.
   solution : Remarquons tout d'abord que deux mots conjugués ont toujours la même longueur.

   ⚠ Je vais réécrire différemment le fait d'être conjugué pour me simplifier les notations dans la suite.

   Soit $f : \begin{array}{ccc} \Sigma^* & \to & \Sigma^* \\ u & \mapsto & u_1...u_{|u|-1}u_0 \end{array}$ (la fonction effectuant une permutation circulaire d'une case vers la droite sur les lettres d'un mot). Ainsi, deux mots u et v sont conjugués si et seulement si il existe $i \in [\![0, n[\![$ tel que $v = f^i(u)$.

   - *Réflexivité :* Soit $m \in \Sigma^*$. En appliquant la définition 1.3 pour $i = 0$, on voit que m est conjugué à lui même, donc mCm.
   - *Symétrie :* Soit $(u, v) \in (\Sigma^*)^2$ tel que uCv. Notons $n = |u|$, on a aussi $n = |v|$. Par définition, il existe $i \in [\![0, n[\![$ tel que $v = u_i...u_{n-1}u_0...u_{i-1}$.
     Notons $j = n - i$. On a alors $u = v_j...v_{n-j}v_0...j - 1$. Donc vCu.
     Remarque : En utilisant la fonction f, il suffit de remarquer que $f^n = \mathrm{Id}$. Donc si $v = f^i(u)$, alors $u = f^{n-i}(v)$.
   - *Transitivité :* Soit $(u, v, w) \in (\Sigma^*)^3$ tel que $uCvCw$. Il existe $(i, j) \in [\![0, n[\![^2$ tel que $v = f^i(u)$ et $w = f^j(v)$. Alors $w = f^{i+j}(u)$. Donc u et w sont conjugués.

Définition 1.4. *(Collier).*
Un collier est le plus petit mot dans l'ordre lexicographique d'une classe de mots équivalents par la relation C.
Un collier d'ordre n est dit périodique s'il peut s'écrire m^l , où $m \in \Sigma^r$, $r \geq 2$ et $l > 1$. Il est dit apériodique sinon, autrement dit si deux conjugaisons non triviales des membres de sa classe d'équivalence ne sont jamais égales.

Définition 1.5. *(Mot de Lyndon).*
Un mot $m \in \Sigma^*$ est un mot de Lyndon si c'est un collier apériodique.

4. On suppose $0 \prec 1$. Pour les mots suivants, indiquer si ce sont ou non des mots de Lyndon. Dans le cas négatif, justifier votre réponse :
   (a) 0010011.
   (b) 010011.
   (c) 001001.

   solution :
   (a) Oui
   (b) Ce n'est pas un collier : 001101 est conjugué et plus petit.
   (c) Périodique.

5. Écrire une fonction Caml `Lyndon : string -> bool` telle que `Lyndon m` renvoie `true` si m est un mot de Lyndon. Cette fonction fera appel à une fonction récursive.

2 Génération de mots de Lyndon

Soit $m \in \Sigma^*$ un mot de Lyndon. Pour générer à partir de m un mot de Lyndon q de longueur au plus $n \geq |m|$ sur Σ, on utilise l'algorithme 1.

1. Donner l'indice dans m de la i-ème lettre de q en fonction de i et $|m|$.
   solution : Je suppose que par « i-ième lettre », l'énoncé entend « lettre d'indice i » sans quoi il faudrait rajouter -1 dans la formule ci-dessous.
   L'indice dans m de la lettre d'indice i de q est $\left\lfloor \frac{i}{|m|} \right\rfloor$.
2. Pour $\Sigma = \{0, 1\}$, on donne le mot de Lyndon $m = 00111$ et $n = 9$. Donner le mot de Lyndon généré par l'algorithme, en déroulant les différentes étapes produites par l'algorithme permettant d'aboutir au mot de Lyndon.
   solution : On trouve 0011101.

Données : $m \in \Sigma^*$ un mot de Lyndon, $n \geq |m|$.
Sortie : q un mot de Lyndon.
**** Étape 1 ****
Concaténer le mot m à lui même jusqu'à obtenir un mot q de longueur n. La dernière occurrence de m pourra être tronquée pour arriver à un mot de longueur exactement n.
**** Étape 2 ****
tant que le dernier symbole de q est le plus grand symbole de Σ faire :

> Ôter ce symbole de q.

**** Étape 3 ****
Remplacer le dernier symbole de q par le symbole qui suit dans Σ.
Renvoyer q.

FIGURE 1 – Algorithme de génération de mot de Lyndon

L'algorithme peut être utilisé pour générer tous les mots de Lyndon de longueur au plus n. Pour ce faire, on part du plus petit symbole de Σ et on itère les trois étapes (1-2-3) de l'algorithme jusqu'à arriver au mot vide.

3. Partant de $m = 0$ et toujours pour $\Sigma = \{0, 1\}$, construire par l'algorithme tous les mots de Lyndon de longueur au plus 4.

   solution : On trouve 0001, 001, 0011, 01, 011, 0111, 1 et $\emptyset$.

4. Donner la complexité de l'algorithme 1 au pire des cas en nombre d'ajouts ou de suppressions de caractères.
   solution :
   - Étape 1 : $O(n)$ ajouts de caractères.
   - Étape 2 : $O(n)$ suppressions de caractères.
   - Étape 3 : $O(1)$.

   La complexité est donc en $O(n)$. Le pire des cas est obtenu lorsque m n'a qu'une seule lettre, qui est la plus grande de l'alphabet. Il y a alors $n-1$ ajouts puis n suppressions.

3 Factorisation de mots de Lyndon

Définition 3.1. *Soit $m \in \Sigma^*$. Une factorisation de Lyndon de m est une suite $m_1, ...m_l$ de mots de Lyndon telle que $m = m_1 \cdots m_l$ et $m_l \preceq ... \preceq m_2 \preceq m_1$.*

On admet le résultat suivant :

Théorème 3.2. *Tout mot $m \in \Sigma^*$ admet une unique factorisation de Lyndon.*

L'algorithme 2 propose une méthode de factorisation d'un mot m (on ne demande pas de le justifier). Le principe est d'itérer sur la chaîne des symboles de m pour trouver le plus grand mot de Lyndon possible. Lorsqu'un tel mot est trouvé, il est ajouté à la liste L des facteurs de m et la recherche est poursuivie sur la sous-chaîne restante.

5. En utilisant l'algorithme 2, écrire une fonction factorisation m qui réalise la factorisation d'un mot m donné. Cette fonction fera appel à une ou des fonction(s) récursive(s).

   solution : Je n'utilise pas d'accu pour L, donc mon programme renverra le résultat dans l'ordre inverse de celui obtenu avec accu. Ceci dit, l'algo décrit dans l'énoncé ne précisait pas de quel côté ajouter p à L.
   Quand on étudie l'algo, on voit que le premier mot créé est le plus grand dans l'ordre alphabétique. Et il sera le premier dans la liste renvoyée par la fonction aux. Se je veux le résultat dans l'ordre décroissant, je dois donc retourner le résultat de aux.

```
42 let factorisation m =
43
44   let rec boucle j k m =
45     (* invariant : j > k ⩾ 0 *)
46     let l = String.length m in
47     if j>l then []
48     else
```

Données : un mot de Lyndon m.
Sortie : la liste L des mots de Lyndon décroissants de la factorisation de m.

$L \leftarrow []$
$j \leftarrow 1$
$k \leftarrow 0$
tant que $j \leq |m|$ faire :

- si $j = |m|$ ou $m_k > m_j$ alors :
  - $p \leftarrow m_0 ... m_{j-k-1}$
  - Ajouter p à L
  - Supprimer p de m
  - $k \leftarrow 0$
  - $j \leftarrow 1$
- sinon :
  - si $m_k = m_j$ alors :
    - $k \leftarrow k + 1$
    - $j \leftarrow j + 1$
  - sinon
    - $k \leftarrow 0$
    - $j \leftarrow j + 1$

Renvoyer L

FIGURE 2 – Factorisation d'un mot de Lyndon

```
49        if j = l || m.[k] > m.[j] then
50          let p = String.sub m 0 (j-k) in
51          let q = String.sub m (j-k) (l-(j-k)) in
52          p::(boucle 1 0 q)
53        else
54          if m.[k] = m.[j] then
55            boucle (j+1) (k+1) m
56          else
57            boucle (j+1) 0 m
58
59    in
60    List.rev (boucle 1 0 m)
61  ;;
```

4 Mots de de Bruijn

4.1 Définition

Définition 4.1. *Un mot de de Bruijn d'ordre n sur Σ est un collier qui contient tous les mots de Σ^n une et une seule fois.*

Par exemple, pour $\Sigma = \{a, b, c\}$ et $n = 2$, $m = abacbbcca$ est un mot de de Bruijn puisqu'il contient une unique fois chaque mot de longueur 2 sur Σ, le mot 'aa' étant obtenu par circularité de m.

solution : Cet exemple semble suggérer que pour la définition ci-dessus « u contient v » doive être interprété en « v est un sous-mot d'un conjugué de u ».

1. Donner la longueur d'un mot de de Bruijn en fonction de n et du nombre k de symboles de Σ.
   solution : Soit Σ un alphabet de cardinal k.
   Le nombre de mots de longueur n sur Σ est k^n, je peux donc affirmer que la longueur d'un mot de Bruijn sur n est au moins k^n car un mot de longueur l contient l sous-mots).

Je n'ai en revanche pas d'idée de la longueur exacte, et vu les parties suivantes je soupçonne que ça n'est pas évident... Sommes-nous sûrs qu'un collier ne peut pas avoir deux fois un même sous-mot à des emplacements différents ?

4.2 Graphe de Bruijn

Définition 4.2. *Le graphe de de Bruijn d'ordre n sur Σ est le graphe orienté $B(k,n) = (V,E)$ où :*

- $V = \Sigma^n$ *est l'ensemble des sommets du graphe ;*
- $E = \left\{(am, mb)\ ;\ a, b \in \Sigma, m \in \Sigma^{n-1}\right\}$ *est l'ensemble des arcs orientés du graphe.*

On value les arcs par le dernier symbole du noeud terminal de chaque arc : ainsi $(am, mb) \in E$ est étiqueté par b.

Dans ce graphe, certains arcs ont pour sommet initial et terminal un même sommet de V. Ces arcs sont appelés des boucles.

1. Donner l'ensemble des mots de longueur 3 sur $\Sigma = \{0,1\}$. En déduire de graphe de de Bruijn $B(2,3)$ associé.
2. Montrer que le degré entrant et le degré sortant de chaque sommet est égal à k.
   Rappel : Le degré entrant d'un sommet s est le nombre d'arêtes qui aboutissent à s, son degré sortant est le nombre d'arêtes qui en partent.
   solution : Soit $u \in V$. Soit m le mot formé des $n-1$ dernières lettres de u et b sa première lettre, donc $u = bm$.
   L'ensemble des arêtes qui aboutissent à u est $\left\{(am, mb)\ ;\ a \in \Sigma\right\}$. Il y en a bien k. (La fonction $a \mapsto (am, mb)$ est une bijection entre Σ et l'ensemble de ces arêtes.)
   Même raisonnement pour les arêtes sortantes.
3. En déduire le nombre d'arcs orientés $|E|$ en fonction de k et n.
   solution : On a $A = \bigcup_{v \in V}(A \cap (\{v\} \times V))$ et cette union est disjointe. Par la question précédente, pour tout $v \in V$, $|A \cap (\{v\} \times V)| = k$.
   Donc $|A| = \sum_{v \in V} k = k^{n+1}$.

Définition 4.3. *(Successeur, prédécesseur).*
Soit $G = (V,E)$ un graphe orienté. $v \in V$ est un successeur (respectivement prédécesseur) de $u \in V$ si $(u,v) \in E$ (resp. $(v,u) \in E$).

4. Soient $B(k,n) = (V,E)$ et $m \in V$. Montrer que tous les prédécesseurs de m ont le même ensemble de successeurs.
   solution : Soit n le mot formé des $n-1$ dernières lettre de m et b sa première lettre. Donc $m = bn$. Et l'ensemble des prédécesseurs de m est $\{am\ ;\ a \in \Sigma\}$.
   Or pour tout $a \in \Sigma$, l'ensemble des successeurs de an est $\{nc\ ;\ c \in \Sigma\}$. On constate que cet ensemble est indépendant de a : c'est bien le même quel que soit le prédécesseur de m considéré.
5. Soit $p = (p_0 \cdots p_{n-1}) \in V$. Donner l'ensemble des sommets m tels que $(p,m) \in E$.
   solution : Vu la définition, il s'agit de $\left\{p_1 ... p_{n-1} x\ ;\ x \in \Sigma\right\}$.
6. Proposer une méthode pour construire les sommets de $B(k,n+1)$ à partir des arcs de $B(k,n)$.
   Donner le sommet créé dans $B(2,4)$ à partir de l'arc $(001,010)$ de $B(2,3)$.
   solution : Soit (u,v) une arête. Soit $(m,a,b) \in \Sigma^{n-1} \times \Sigma \times \Sigma$ tel que $u = am$ et $v = mb$. Alors le mot amb est dans Σ^{n+1}.
   Réciproquement, soit $n \in \Sigma^{n+1}$. Soit a sa première lettre, b sa dernière lettre, et m le mot formé des lettres restantes. Alors (am, mb) est une arête de $B(k,n)$ et la construction précédente appliquée à celle-ci donne n.
   Ainsi, cette construction permet bien d'obtenir tous les sommets de $B(k,n+1)$.

   Dans l'exemple proposé, on obtient le mot 0010.
7. Proposer de même une construction des arcs de $B(k,n+1)$ en fonction des arcs adjacents de $B(k,n)$.
   Donner l'arc de $B(2,4)$ créé par les arcs adjacents de $B(2,3)$ $(001,011)$ et $(011,110)$.
   solution : Soient e_1 et e_2 deux arcs adjacents dans $B(k,n)$. Soit $(m,p,q) \in V^3$ tel que $e_1 = (m,p)$ et $e_2 = (p,q)$. Soit $(a,b,m') \in \Sigma \times \Sigma \times \Sigma^{n-1}$ tel que $e_1 = (am', m'b)$, et soit $(c,d,p') \in \Sigma \times \Sigma \times \Sigma^{n-1}$ tel que $e_2 = (cp', p'd)$.
   Ainsi, $m'b = cp'$. Soit w le mot formé des $n-2$ dernières lettres de m' ; c'est aussi le mot formé des $n-2$ premières lettre de p', et on a $m' = cw$ et $p' = wb$. Ainsi, $e_1 = (acw, cwb)$ et $e_2 = (cwb, wbd)$.
   Or le sommet obtenu à partir de e_1 dans $B(k,n+1)$ est $acwb$ et celui obtenu à partir de e_2 est $cwbd$. Et $(acwb, cwbd)$ est une arête dans $B(k,n+1)$.
   Réciproquement, toute arête de $B(k,n+1)$ peut être obtenue de la sorte : si (xu, uy) est une telle arête, poser $a = x$, $d = y$, c la première lettre de u, b la dernière, w le reste de u, alors (acw, cwb) et (cwb, wbd) sont deux arêtes adjacentes de $B(k,n)$.
   En conclusion, les arêtes de $B(k,n+1)$ sont les couples de sommets obtenus par la construction de la question précédente à partir de deux arêtes adjacentes.

4.3 Construction des mots de Bruijn

On propose ici trois algorithmes de construction de mots de de Bruijn.

4.3.1 Construction à l'aide de $B(k, n)$

Les mots de de Bruijn d'ordre n sur Σ peuvent être construits en parcourant $B(k, n)$.

Définition 4.4. *(Circuit eulérien).*
Soit G un graphe orienté. Un circuit eulérien est un chemin dont l'origine et l'extrémité coïncident et passant une fois et une seule par chaque arête de G.

8. Construire $B(2, 2)$ et trouver dans ce graphe un circuit eulérien. Vérifier que la concaténation des étiquettes lues au fil de ce circuit donne un représentant d'un mot de de Bruijn et donner son ordre sur $\{0, 1\}$.

On admet les résultats suivants :

- un graphe $G = (V, E)$ possède un circuit eulérien si et seulement si il est connexe et si, pour tout $v \in V$, le degré entrant de v et le degré sortant de v sont égaux.
- un circuit eulérien dans le graphe $B(k, n)$ correspond à un mot de de Bruijn.

9. En déduire qu'il existe au moins un mot de de Bruijn pour tout alphabet Σ et tout n.
solution : Soit Σ un alphabet et $n \in \mathbb{N}$. Soit G le graphe de Bruijn associé. Pour pouvoir utiliser les propriétés admises ci-dessus il faut vérifier que G est connexe et que chaque sommet a un degré entrant égal à son sommet sortant.
Le second point a été vu partie 4.2 question 2.
Montrons que G est connexe. Soient $(u, v) \in (\Sigma^n)^2$. Le chemin suivant relie alors u à v :

$$u_0...u_{n-1}, \quad u_1...u_{n-1}v_0, \quad u_2...u_{n-1}v_0v_1, \quad ... \quad , u_{n-1}v_0...v_{n-2}, \quad v_0...v_{n-1}$$

Ainsi, la concaténation des étiquettes lues au fil d'un circuit eulérien de $B(k, n)$ donne un représentant d'un mot de de Bruijn d'ordre $n + 1$ sur k symboles.

4.3.2 Construction à l'aide de l'algorithme Prefer One

Pour construire les mots de de Bruijn d'ordre n sur $\Sigma = \{0, 1\}$, on peut également utiliser l'algorithme 3, dit algorithme Prefer One.

Dans cet algorithme, « les n derniers symboles de m n'ont pas été rencontrés auparavant » signifie que le mot composé des n derniers symboles de m n'est pas un sous-mot de m, situé entre le premier et le $(|m| - 1)$-ième symbole de m.

1. Appliquer l'algorithme au cas $n = 3$ et $\Sigma = \{0, 1\}$. Écrire les valeurs successives de m au cours de l'exécution.

4.3.3 Construction à l'aide de la relation aux mots de Lyndon

La troisième construction utilise les notions de collier et de mots de Lyndon.

1. Donner, pour $k = 2$ et $n = 4$, les classes d'équivalence de tous les mots binaires de longueur 4 pour la relation C. En déduire les colliers correspondants.
2. En déduire les mots de Lyndon de longueur 4 pour $\Sigma = \{0, 1\}$.

Plus généralement, les mots de de Bruijn et de Lyndon sont étroitement liés. On peut montrer que si l'on concatène, dans l'ordre lexicographique, les mots de Lyndon sur k symboles dont la longueur divise un entier n, alors on obtient le mot de de Bruijn le plus petit, pour l'ordre lexicographique, de tous les mots de de Bruijn de longueur n sur k symboles.

3. Déduire de la question précédente le plus petit mot de de Bruijn pour $n = 4$ et $\Sigma = \{0, 1\}$.

4.3.4 Application

La soirée a été longue, vous rentrez chez vous mais, au pied de votre immeuble, vous vous trouvez confronté à un sérieux problème : vous avez complètement oublié le code d'entrée à n chiffres de la porte. On suppose que le digicode de l'appartement est composé de $1 \leq k \leq 10$ chiffres $0, 1, ...k - 1$, qui constituent l'alphabet Σ.

Le digicode fonctionne de la façon suivante : vous tapez successivement sur les chiffres afin de composer un mot. À chaque nouveau symbole entré à partir du n-ième, le digicode teste le mot constitué par les n derniers chiffres pour voir s'il correspond au code secret.

Données : n, Σ
Résultat : m mot de de Bruijn de longueur n sur Σ.
$m \leftarrow$ suite de n zeros
STOP←false
tant que STOP=false faire

> Etape 1
> Ajouter un 1 à la fin de m.
> si les n derniers symboles de m n'ont pas été rencontrés auparavant alors :
>
> > Répeter Etape 1
>
> Sinon :
>
> > Retirer le 1 ajouté à la fin de m.
> > Passer à Etape 2
>
> Etape 2
> Ajouter un 0 à la fin de m.
> si les n derniers symboles de m n'ont pas été rencontrés auparavant alors :
>
> > Aller à l'Etape 1
>
> sinon :
>
> > STOP←true

FIGURE 3 – Algorithme Prefer One

Ainsi, par exemple pour $n = 4$, si vous tapez la séquence 021201, le digicode teste successivement 0212, 2120 et 1201.

Étant pressé de regagner votre lit, vous cherchez à taper un minimum de touches pour ouvrir la porte. On note la longueur de la plus petite séquence de frappe de touches qui vous permet de rentrer dans l'immeuble.

1. Donner un encadrement de :
   - pour la borne supérieure, on considèrera que l'on met bout à bout tous les mots possibles de n chiffres construits sur Σ ;
   - pour la borne inférieure, on cherchera un mot sans redondance, c'est-à-dire qui contient une et une seule fois chaque mot de n chiffres.
2. Expliquer en une phrase en quoi les mots de de Bruijn peuvent vous aider. En vous inspirant de l'algorithme 3, donner une séquence la plus courte de chiffres à taper pour ouvrir à coup sûr la porte de votre immeuble, lorsque $n = 2$ et $k = 4$.
3. Comparer alors le nombre maximum de frappes de touches du digicode à effectuer en utilisant les mots de de Bruijn, avec celui calculé par la méthode brute. Calculer ces nombres pour :
   - $k = 4$ et $n = 2$.
   - $k = 10$ et $n = 4$.

   Que conclure quant à l'efficacité de votre stratégie ?