Option Informatique

TD $n°04$

Corrigé

1 Mots de Lyndon

1.1 Définitions

1. L'ordre lexicographique sur les chaînes de caractères est déjà implémenté en Caml (c'est l'ordre naturel du type string) mais il ne fait pas partie des "fonctions autorisées" par le sujet, il s'agit donc de le réécrire.

```
let rec ordre m p =
        if m = "" && p= "" then Egal
        else if m= "" then Inferieur
        else if p= "" then Superieur
        else if m.[0]<p.[0] then Inferieur
        else if m.[0]>p.[0] then Superieur
        else ordre (String.sub m 1 (String.length m - 1))
                   (String.sub p 1 (String.length p - 1));;
```

2. `let conjugue m i = (String.sub m i (String.length m - i))^(String.sub m 0 i);;`

3. Pour $k \in \mathbb{Z}$ et $n \in \mathbb{N} \setminus \{0\}$, notons $k\ [n]$ le reste dans la division euclidienne de k par n.
   - Réflexivité : soit $m \in \Sigma^*$. Notons $n = |m|$. Pour $i = 0$ on a $m_i \cdots m_{n-1} m_0 \cdots m_{i-1} = m$ donc on a $m\,C\,m$. D'où la réflexivité.
   - Symétrie : soient $m, p \in \Sigma^*$. Notons $n = |m|$. Supposons $m\,C\,p$. Ainsi il existe $i \in \{0, \ldots, n-1\}$ tel que $m_i \cdots m_{n-1} m_0 \cdots m_{i-1} = p$. Donc $|p| = n$ et $p_0 = m_i, \cdots, p_{n-i-1} = m_{n-1}, p_{n-i} = m_0, \ldots, p_{n-1} = m_{i-1}$ $(\star)$. Si on a $i = 0$ alors $m = p$ et donc en particulier $p\,C\,m$ d'après la réflexivité. Sinon, posons $j = n - i$. On a $p_j \cdots p_{n-1} p_0 \cdots p_{j-1} = p_{n-i} \cdots p_{n-1} p_0 \cdots p_{n-i-1} = m$ d'après $(\star)$.
     D'où la symétrie.
   - Transitivité : soient $m, p, o \in \Sigma^*$. Notons $n = |m|$. Supposons $m\,C\,p$ et $p\,C\,o$. Ainsi il existe $i \in \{0, \ldots, n-1\}$ tel que $m_i \cdots m_{n-1} m_0 \cdots m_{i-1} = p$ et il existe $j \in \{0, \ldots, n-1\}$ tel que $p_j \cdots p_{n-1} p_0 \cdots p_{j-1} = o$. En particulier on a $|o| = |p| = n$ et en reprenant les arguments donnés pour la symétrie on trouve $p_k \cdots p_{n-1} p_0 \cdots p_{k-1} = m$.
     D'où la transitivité.

   La relation C est donc bien une relation d'équivalence.

4. — 0010011 est bien un mot de Lyndon. En effet, il est minimal dans sa classe d'équivalence (le seul mot qui commence par deux 0 dans sa classe est 0011001 qui lui est strictement supérieur, et les autres ne commencent pas par deux 0 donc lui sont strictement supérieurs) et bien apériodique (il est de longueur 7 qui est premier donc s'il était périodique tous ses caractères seraient égaux).
   — 010011 n'est pas un mot de Lyndon. Il n'est pas minimal dans sa classe d'équivalence (001101 lui est strictement inférieur).
   — 001001 n'est pas un mot de Lyndon. Il est périodique (car égal à 001^2).

5. Supposons que m soit un mot de Lyndon. En particulier m est un collier, donc un mot plus petit que tous ses conjugués. Supposons qu'un conjugué de m soit égal à m. Il existe alors $i \in [|0, n-1|]$ tel que $m_0 \cdots m_{i-1} m_i \cdots m_{n-1} = m_i \cdots m_{n-1} m_0 \cdots m_{i-1}$, on a donc $\forall k \in \mathbb{Z},\ m_{i+k} = m_k$, où l'addition est prise modulo n, et donc m est périodique, contradiction. C'est donc que m est strictement plus petit que tout conjugué strict de m.

   Supposons que m soit strictement plus petit que tout conjugué strict de m. En particulier m est plus petit que tout conjugué strict de m donc c'est un collier. Supposons qu'il soit périodique de période $i > 0$, alors $m_0 \cdots m_{i-1} m_i \cdots m_{n-1} = m_i \cdots m_{n-1} m_0 \cdots m_{i-1}$ et donc m est égal à l'un de ses conjugués stricts, contradiction. C'est donc que m est un collier apériodique, c'est-à-dire un mot de Lyndon.

6. On peut tester séparément le fait que l'argument soit un collier d'une part, et soit apériodique d'autre part, mais c'est coûteux. Il semble plus pertinent ici d'utiliser la caractérisation équivalente de la question précédente.

```
let lyndon m = (* En utilisant la caracterisation de stricte minimalite *)
        let n = String.length m in
        let rec strictement_minimal i =
                match ordre m (conjugue m i) with
                |_ when i=n -> true
                |Inferieur -> strictement_minimal (i+1)
                |_ -> false in
        (strictement_minimal 1);;
```

7. La $i^{\text{ième}}$ lettre de q, à l'exception éventuelle de la dernière, est la $(i\ [n])^{\text{ième}}$ lettre de m, où n est la longueur de m. La dernière lettre que q est la $(i\ [n])^{\text{ième}}$ lettre de m si ce n'est pas la plus grande lettre de l'alphabêt, ou la lettre qui suit la $(i\ [n])^{\text{ième}}$ lettre de m dans l'alphabêt sinon.

8. L'étape 1 conduit à initialiser q au mot 001110011.
L'étape 2 conduit à tronquer q en 0011100.
L'étape 3 enfin conduit à changer q en 0011101.
Le mot de Lyndon généré est donc 0011101 (on peut vérifier qu'il s'agit bien d'un mot de Lyndon).

9. La description de l'algorithme de génération des mots de Lyndon par l'énoncé est très absconse.
   - Le premier mot de Lyndon de longueur $\leqslant 4$ est 0 (donné par l'énoncé).
   - On applique l'algorithme à $m = 0$.
     — On applique l'étape 1 de l'algorithme, on obtient $q = 0000$.
     — On applique l'étape 2, q n'est pas modifié.
     — On applique l'etape 3, q devient 0001 qui est donc notre deuxième mot de Lyndon.
   - On applique l'algorithme à $m = 0001$.
     — On applique l'étape 1 de l'algorithme, on obtient $q = 0001$.
     — On applique l'étape 2, q devient 000.
     — On applique l'etape 3, q devient 001 qui est donc notre troisième mot de Lyndon.
   - On applique l'algorithme à $m = 001$.
     — On applique l'étape 1 de l'algorithme, on obtient $q = 0010$.
     — On applique l'étape 2, q n'est pas modifié.
     — On applique l'etape 3, q devient 0011 qui est donc notre quatrième mot de Lyndon.
   - On applique l'algorithme à $m = 0011$.
     — On applique l'étape 1 de l'algorithme, on obtient $q = 0011$.
     — On applique l'étape 2, q devient 00.
     — On applique l'etape 3, q devient 01 qui est donc notre cinquième mot de Lyndon.
   - On applique l'algorithme à $m = 01$.
     — On applique l'étape 1 de l'algorithme, on obtient $q = 0101$.
     — On applique l'étape 2, q devient 010.
     — On applique l'etape 3, q devient 011 qui est donc notre sixième mot de Lyndon.
   - On applique l'algorithme à $m = 011$.
     — On applique l'étape 1 de l'algorithme, on obtient $q = 0110$.
     — On applique l'étape 2, q n'est pas modifié.
     — On applique l'etape 3, q devient 0111 qui est donc notre septième mot de Lyndon.
   - On applique l'algorithme à $m = 0111$.
     — On applique l'étape 1 de l'algorithme, on obtient $q = 0111$.
     — On applique l'étape 2, q devient 0.
     — On applique l'etape 3, q devient 1 qui est donc notre huitième mot de Lyndon.
   - On applique l'algorithme à $m = 1$.
     — On applique l'étape 1 de l'algorithme, on obtient $q = 1111$.

— On applique l'étape 2, q devient ε.

C'est le cas d'arrêt, c'est fini.

On a ainsi obtenu les huits mots 0, 0001, 001, 0011, 01, 011 0111, 1.

10. Lors de la première étape on complète m en un mot q de longueur n, ce qui nécessite moins de n ajouts de caractère, donc une complexité en $O(n)$ pour cette étape. Lors de la seconde étape, on supprime au plus n caractères au mot q, donc une complexité en $O(n)$ pour cette étape. Lors de la troisième étape, on modifie un caractère de q, ce qui coûte 0 suppression/ajout ou 2 suppressions/ajouts suivant comment on voit les choses, dans tous les cas c'est en $O(1)$.
La complexité totale dans le pire des cas est donc en $O(n)$.

1.2 Factorisation de mots de Lyndon

11. On traduit en Caml (remarque : comme je n'aime pas les if, je matche sur "rien" !) :

```
let factorisation m = (* m un mot quelconque *)
  let rec aux j k n l m = match () with
    |_ when j>n -> l
    |_ when j=n || m.[k]>m.[j] -> let p = String.sub m 0 (j-k) in
                                  let m_new = String.sub m (j-k) (n-j+k) in
                                  aux 1 0 (n-j+k) (p :: l) m_new
    |_ when m.[k]=m.[j] -> aux (j+1) (k+1) n l m
    |_ -> aux (j+1) 0 n l m in
  aux 1 0 (String.length m) [] m;;
```

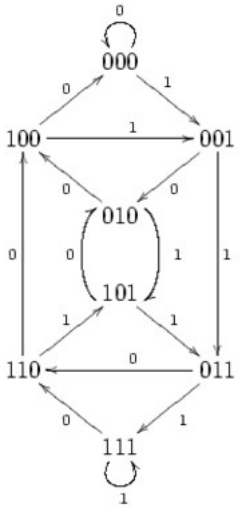
2 Mots de de Bruijn

2.1 Définitions

12. Un tel mot de de Bruijn (s'il en existe un !) est de longueur k^n. En effet, chaque caractère du mot est le début d'un mot différent de longueur n et il en existe k^n. Le dernier caractère, qui correspond donc au dernier de ces k^n mots (ce mot va s'obtenir par circularité), doit donc être le $(k^n)^{ime}$ caractère.

2.2 Graphe de de Bruijn

13. Les $2^3 = 8$ mots de longueur 3 sur l'alphabet $\Sigma = \{0, 1\}$ sont 000, 001, 010, 011, 111, 110, 101, 100.
On en déduit que $B(2, 3)$ peut par exemple se représenter comme suit :



14. Si am est un sommet du graphe alors il y a un arc sortant de am pour chaque lettre $b \in \Sigma$. Il y en a donc k ce qui signifie $d_s(am) = k$. Ceci étant vrai pour chaque sommet am, le degré entrant de chaque sommet est égal à k.

    De même, si mb est un sommet du graphe alors il y a un arc entrant en mb pour chaque lettre $a \in \Sigma$. Il y en a donc k ce qui signifie $d_e(mb) = k$. Ceci étant vrai pour chaque sommet mb, le degré sortant de chaque sommet est également égal à k.

15. On a $|E| = \sum_{m\in\Sigma^n} d_e(m) = \sum_{m\in\Sigma^n} d_s(m)$. Dans les deux cas on trouve $|E| = k \sum_{m\in\Sigma^n} 1 = k^{n+1}$.

16. Notons $m = ub$. Les prédécesseurs de m sont tous les mots de la forme au avec $a \in \Sigma$. Pour tout $a \in \Sigma$, l'ensemble des sucesseurs d'un tel mot est $\{ux, x \in \Sigma\}$ qui ne dépend pas de a. Cela montre bien que tous les prédécesseurs de m ont le même ensemble de successeurs.

17. Un sommet m vérifie $(p, m) \in E$ si et seulement si il existe $a, b \in \Sigma$ tels que $ap = mb$. On a alors nécessairement $a = p_0$, et l'ensemble de ces sommets est donc $\{p_1 \cdots p_{n-1}b,\ b \in \Sigma\}$.

18. Les sommets de $B(k, n+1)$ sont les mots de Σ^{n+1}, alors que les arcs de $B(k, n)$ sont les (am, mb) avec $(a, m, b) \in \Sigma \times \Sigma^{n-1} \times \Sigma$. Tout mot de Σ^{n+1} pouvant s'écrire sous la forme amb avec $(a, m, b) \in \Sigma \times \Sigma^{n-1} \times \Sigma$, on obtient les sommets de $B(k, n+1)$ à partir des arcs (am, mb) de $B(k, n)$ en appliquant la transformation $(am, mb) \mapsto amb$.

    En particulier, le sommet créé dans $B(2, 4)$ à partir de l'arc $(001, 010)$ de $B(2, 3)$ est le mot 0010.

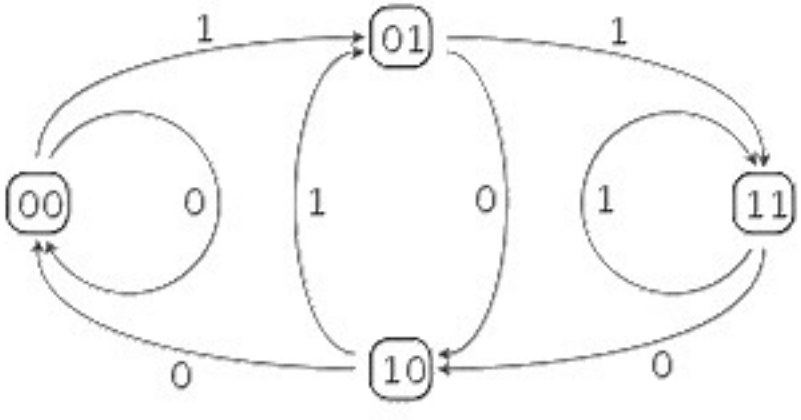
19. Considérons deux arcs ajacents (m, p) et (p, q) de $B(k, n)$. Par définition il existe $a \in \Sigma$, $u \in \Sigma^{k-2}$, $b \in \Sigma$, $c \in \Sigma$, $d \in \Sigma$, tels que $m = acu$, $p = cub$, $q = ubd$. Ainsi $acub$ et $cubd$ sont deux sommets de $B(k, n+1)$ et $(acub, cubd)$ un arc de $B(k, n+1)$ qui les relie. Réciproquement, un arc entre deux sommets de $B(k, n+1)$ est clairement de cette forme.

    Par exemple, l'arc de $B(2, 4)$ créé par les deux arcs adjacents de $B(2, 3)$ $(001, 011)$ et $(011, 110)$, est $(0011, 0110)$.

2.3 Construction des mots de de Bruijn

2.3.1 Construction à l'aide de B(k,n)

20. Un dessin pour $B(2, 2)$ permet de le décrire complètement :



    Un circuit eulérien dans $B(2, 2)$: $00 \xrightarrow{0} 00 \xrightarrow{1} 01 \xrightarrow{0} 10 \xrightarrow{1} 01 \xrightarrow{1} 11 \xrightarrow{1} 11 \xrightarrow{0} 10 \xrightarrow{0} 00$.
    La concaténation des étiquettes donne 01011100 qui est bien conjugué au mot de de Bruijn sur $\{0, 1\}$ d'ordre 3 (les mots obtenus sont, dans l'ordre, 010, 101, 011, 111, 110, 100, 000, 001 qui sont bien tous les mots de longueur 3, obtenus une et une unique fois). Si on veut le vrai mot de de Bruijn, il faut prendre le conjugué qui est un collier (c'est-à-dire démarrer le cycle au sommet 11), on obtient 00010111.

21. Il suffit de voir que $B(k, n) = (V, E)$ a un toujours circuit eulérien, mais on a vu plus haut qu'on avait $\forall m \in V,\ d_s(m) = k = d_e(m)$ donc d'après ii., $B(k, n)$ a bien un circuit eulérien.

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

22. On appliquer l'algorithme au cas $n = 3$ et $\Sigma = \{0, 1\}$.

Initialisation : $m = 000$, STOP=false.
Étape 1 : $m = 0001$.
001 n'a pas été rencontré donc : $m = 00011$.
011 n'a pas été rencontré donc : $m = 000111$.
111 n'a pas été rencontré donc : $m = 0001111$.
111 a été rencontrée donc : $m = 000111$ et on passe à l'étape 2.
Étape 2 : $m = 0001110$.
110 n'a pas été rencontré donc on passe à l'étape 1.
Étape 1 : $m = 00011101$.
101 n'a pas été rencontré donc : $m = 000111011$.
011 a été rencontré donc : $m = 00011101$ et on passe à l'étape 2.
Étape 2 : $m = 000111010$.
010 n'a pas été rencontré donc on passe à l'étape 1.
Étape 1 : $m = 0001110101$.
101 a été rencontré donc : $m = 000111010$ et on passe à l'étape 2.
Étape 2 : $m = 0001110100$.
100 n'a pas été rencontré donc on passe à l'étape 1.
Étape 1 : $m = 0001110101$.
101 a été rencontré donc : $m = 000111010$ et on passe à l'étape 2.
Étape 2 : $m = 0001110100$.
000 a été rencontré donc STOP=true.
On retourne 0001110100**.**

Notons que le mot retourné n'est pas un mot de de Brujin d'ordre 3 sur $\{0, 1\}$ pour la définition qui était donnée jusqu'alors, puisque celle-ci autorisait d'obtenir certains mots de Σ^n par circularité (bien sûr, comme l'algorithme ne tient pas compte de cette circularité, c'était inévitable ; on peut par contre vérifier qu'on obtient bien tous les mots de $\{0, 1\}^3$ une et une seule fois **sans** circularité ici). Pour obtenir un mot de de Bruijn au sens de l'énoncé, il faut retirer les deux zéros finals, ce qui donne 00011101. On vérifie immédiatement qu'on a bien un mot de de Bruijn, mais ce n'est pas le même que celui obtenu dans la question précédente, ce qui montre qu'un mot de de Bruijn n'est en général pas unique (remarque : le mot 00010111 obtenu par circuit eulérien peut s'obtenir avec l'algorithme "prefer opposite" qui est une variante de "prefer one" dans laquelle on donne priorité non pas au caractère 1 mais au caractère opposé au dernier caractère écrit).

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

23. Comme $k = |\Sigma|$ est égal à 2, on a à renommage près $\Sigma = \{0, 1\}$, convention qu'on va utiliser ici.

Indiquons les classes d'équivalences pour $\mathcal{C}$ dans Σ^4 par leur représentant qui est un collier.
- $\overline{0000} = \{0000\}$;
- $\overline{0001} = \{0001, 1000, 0100, 0010\}$;
- $\overline{0011} = \{0011, 1001, 1100, 0110\}$;
- $\overline{0101} = \{0101, 1010\}$;
- $\overline{0111} = \{0111, 1011, 1101, 1110\}$;
- $\overline{1111} = \{1111\}$;

comme $1 + 4 + 4 + 2 + 4 + 1 = 16$, on n'a oublié personne.

24. Les mots de Lyndon de longueur 4 pour $\Sigma = \{0, 1\}$ sont donc les mots apériodiques parmi 0000, 0001, 0011, 0101, 0111, 1111, soit 0001, 0011, 0111 (0000 et 1111 sont 1-péridique et 0101 est 2-périodique). Ce sont bien sûr ceux dont la classe d'équivalence est de cardinal $n = 4$.

25. Les diviseurs de 4 sont 1, 2 et 4. Les mots de Lyndon de longueur 1 sur $\{0, 1\}$ sont : 0 et 1. De longueur 2 : 01. De longueur 4 : 0001, 0011, 0111. Mettons ces 6 mots dans l'ordre lexicographique : 0, 0001, 0011, 01, 0111, 1. Concaténons, on obtient 00000100110101111. Ceci est bien un mot de de Bruijn au sens de l'énoncé (clairement un collier, et on peut vérifier que les 16 mots de $\{0, 1\}^4$ s'obtiennent bien par circularité). Je suis tout disposé à croire que c'est le plus petit dans l'ordre lexicographique.

Remarque : pour $n = 3$, les diviseurs de 3 sont 1 et 3, et les mots de Lyndon de longueur 3 sont 001 et 011, donc

en mettant dans l'ordre lexicographique les mots de Lyndon de longueur 1 ou 3 : 0, 001, 011, 1. En concatérant on obtient 00010111 qui est le mot de de Bruijn d'ordre 3 obtenu par circuit eulérien.

3 Application

26. Nombre de mots de Σ^n : k^n.
    - Si on met bout à bout tous ces mots, qui sont de longueur n, on a un mot de longueur nk^n. C'est ce que le sujet veut que je donne comme borne supérieure.
    - Si on écrit le mot de de Bruijn de longueur n, on a tous les mots de n chiffres une et une seule fois... mais en acceptant la circularité, ce que ne permet pas le digicode (qui n'est même pas capable d'aller dans le passé... encore que, vu son fonctionnement, la question se pose). Il faut donc recopier les $n-1$ premières lettres de ce mot à la fin du mot (c'est ce que fait "prefer one" d'ailleurs). On obtient donc un mot de longueur k^n+n-1. C'est ce que le sujet veut que je donne comme borne inférieure.

27. Avec "prefer one" (ou autrement), on obtient le mot de de Bruijn d'ordre n complété à la fin par ses $n-1$ premières lettres, qui est exactement le mot minimal me permettant d'entrer à coup sûr si je le tape sur le digicode (en moyenne... pour un digicode donné le mot minimal c'est le code du digicode...).

    Donc si $n=2$ et $k=4$ on tape 00000100110101111000 et hop (tu remarqueras que je n'ai pas oublié de recopier les trois zéros à la fin).

28. - Pour $k=4$ et $n=2$ on a $nk^n=32$ et $k^n+n-1=17$.
    - Pour $k=10$ et $n=4$ on a $nk^n=40000$ et $k^n+n-1=10003$.

    La question virée était "que conclure quant à l'efficacité de votre stratégie?" Ma réponse : Il n'y a rien à conclure de cette question quant à l'efficacité de ma stratégie. On a déjà conclu sur l'efficacité de ma stratégie : pour un digicode donné, donner le code du digicode est une meilleure stratégie, et en moyenne il ne peut y avoir mieux que de donner tous les codes possibles, ce qui ne peut pas se faire avec un mot plus court qu'un mot de de Bruijn (complété à la fin par ses $n-1$ premières lettres) **par définition**, ce qui fait que la stratégie est optimale en moyenne **par définition**.