

I Mots de Lyndon

On suppose que l'alphabet Σ est $\{0,1\}$.

On rappelle que $\preceq$ est une relation d'ordre où $u \preceq v$ si u est un préfixe de v

ou s'il existe un entier $k \leq \min\{|u|, |v|\}$ tel que, en notant $u = x_1x_2 \cdots x_n$ et $v = y_1y_2 \cdots y_m$, $x_i = y_i$ pour $1 \leq i < k$ et $x_k < y_k$.

Question 1 Écrire une fonction `inferieur : int list -> int list -> bool` qui prend pour deux mots u et v et renvoie pour résultat le booléen `true` si $u \prec v$ et `false` sinon.

```
let rec inferieur l1 l2 =
  match l1, l2 with
  | _, [] -> false
  | [], _ -> true
  | t1::q1, t2::q2 when t1 < t2 -> true
  | t1::q1, t2::q2 when t1 = t2 -> inferieur q1 q2
  | t1::q1, t2::q2 -> false;;
```

Définition 1

Conjugué Un conjugué d'un mot u est un mot de la forme $v = u''u'$ avec $u = u'.u''$, $u' \neq \varepsilon$ et $u'' \neq \varepsilon$.

Un mot non vide est un **mot de Lyndon** si $u \prec v$ pour tout conjugué v de u .

Question 2 Écrire la fonction `conjugue : int list -> int -> int list` telle que `conjugue u i` renvoie le conjugué $w.v$ de $u = v.w$ avec $|v| = i$.

```
let rec decoupe liste i =
  if i = 0
  then [], liste
  else match liste with
    | [] -> failwith "La liste est trop courte"
    | t::q -> let l1, l2 = decoupe q (i-1) in (t::l1), l2;;
```

```
let conjugue liste i =
  let l1, l2 = decoupe liste i in l2@l1;;
```

Question 3 Écrire une fonction `lyndon : int list -> bool` qui teste si un mot est un mot de Lyndon.

```

let lyndon liste =
  let n = List.length liste in
  let reponse = ref true in
  for i = 1 to (n-1) do
    if inferieur liste (conjugue liste i))
    then reponse := false done;
  !reponse;;

```

Question 4 Soit u un mot. Démontrer que s'il existe un mot qui est à la fois un préfixe et un suffixe de u , alors u n'est pas un mot de Lyndon.

Soit $u = w.v = v.w'$.

- Si $w = w'$, u est égal à un de ses conjugués : il n'est pas un mot de Lyndon.
- Si $w \neq w'$, comme w et w' sont de même longueur, w n'est pas un préfixe de w' et w' n'est pas un préfixe de w ; ils sont comparables par une lettre à la position i .
Si on a $w \prec w'$ alors $v.w \prec v.w' = u$ car il diffèrent en $|v| + i$.
Si $w' \prec w$ alors $w'.v \prec w.v = u$ car il diffèrent en $|v| + i$.
Dans les deux cas on a trouvé un conjugué strictement plus petit.

Dans les deux cas u n'est pas un mot de Lyndon.

Question 5 Démontrer qu'un mot u est un mot de Lyndon si et seulement si pour tout suffixe propre h de u , on a $u \prec h$.

- Si $u \prec h$ pour tout suffixe propre alors, comme $h \prec h.w$, on a $u \prec h.w$ pour $u = w.h$, u est de Lyndon.
- Si u est un mot de Lyndon on a $u \prec h.w$ pour $u = w.h$. On sait que h ne peut pas être un préfixe donc une lettre au moins diffère entre u et h ; dans ce cas on doit avoir $u \prec h$.

Question 6 Démontrer que si un mot u de longueur au moins 2 est un mot de Lyndon alors il existe deux mots de Lyndon f et g tels que $u = f.g$ et $f \prec g$ et, réciproquement, si f et g sont des mots de Lyndon tels que $f \prec g$ alors $f.g$ est un mot de Lyndon.

- On suppose que u est un mot de Lyndon.
Soit g le plus long suffixe propre de u qui soit un mot de Lyndon ; on a $u \prec g$.
On a alors $u = f.g$ et $f \prec f.g = u \prec g$.
Soit h un suffixe de f ; $h.g$ n'est pas un mot de Lyndon en raison de la maximalité de g donc il existe un suffixe h' de $h.g$ telle que $h' \prec h.g$.
La remarque essentielle est que h ne peut pas être un préfixe de h' car sinon on peut écrire $h' = h.t$ et $h' \prec h.g$ implique $t \prec g$ alors que t est un suffixe du mot de Lyndon g : c'est impossible.
 $h' \prec h.g$ avec h' non préfixe de h implique soit h' préfixe de h soit il existe une lettre distincte à la position k pour h et h' donc $h \ll h'$: dans les deux cas on a $h' \prec h$.
 g est un suffixe de u , mot de Lyndon, et h' est un suffixe de g , mot de Lyndon, donc on a $f \prec f.g = u \prec g \prec h' \prec h$ pour tout suffixe de f donc f est un mot de Lyndon.
- On suppose que f et g sont deux mots de Lyndon tels que $f \prec g$.
 - Si f est un préfixe de g alors $g = f.f'$ donc $g \prec f'$. On en déduit $f.g \prec f.f' = g$
 - Sinon $f \ll g$, ce qui implique $f.g \ll g$.

On a toujours $f.g \ll g$.

h est un suffixe de $u = f.g$.

- Si h est un suffixe propre de g alors $g \prec h$ donc $u = f.g \prec g \prec h$.
- Si $h = g$ on a prouvé $u = f.g \prec g = h$.
- Si g est un suffixe propre de h alors $h = h'.g$ avec h' suffixe propre de f . On a alors $f \ll h'$ donc $u = f.g \ll h'.g = h$.

Dans tous les cas on a $u \prec h$: u est un mot de Lyndon.

Question 7 Donner la liste de tous les mots de Lyndon de longueur 5.

On écrit les mots de Lyndon de taille 1 à 4

```
[0] [1]
[0; 1]
[0; 0; 1] [0; 1; 1]
[0; 0; 0; 1] [0; 0; 1; 1] [0; 1; 1; 1]
```

On combine ensuite pour les mots de taille 5

```
[0; 0; 0; 1; 1] [0; 0; 1; 1; 1]
[0; 1; 0; 1; 1] [0; 0; 1; 0; 1] [0; 1; 1; 1; 1]
```

Question 8 Écrire la fonction `Lyndon n` qui calcule, dans un tableau de listes, tous le mots de Lyndon de longueur inférieure ou égale à n . Les mots de Lyndon de longueur k devront être ordonnés dans la composante d'indice k du tableau.

On commence par l'insertion ordonnée

```
let rec insere_mot u liste =
  match liste with
  | [] -> [u]
  | t::q when t = u -> liste
  | t::q when inferieur u t -> u::liste
  | t::q -> t :: (insere_mot u q);;
```

On généralise à une liste

```
let rec insere_liste liste1 liste2 =
  match liste1 with
  | [] -> liste2
  | t::q -> insere_liste q (insere_mot t liste2);;
```

On fusionne un mot à une liste en ne gardant que les mots de Lyndon

```
let rec fusionne_mot u liste =
  match liste with
  | [] -> []
  | t::q when inferieur u t -> (u@t) :: (fusionne_mot u q)
  | t::q -> fusionne_mot u q;;
```

On peut alors passer aux listes

```

let rec fusionne_listes liste1 liste2 =
  match liste1 with
  | [] -> []
  | t::q -> insere_liste (fusionne_mot t liste2)
                    (fusionne_listes q liste2) ;;

```

```

let lyndon n =
  let lynd = Array.make (n+1) [] in
  lynd.(1) <- [[0]; [1]];
  for i = 2 to n do
    for k = 1 to (i - 1) do
      let liste = fusionne_listes lynd.(k) lynd.(i - k) in
      lynd.(i) <- insere_liste liste lynd.(i)
    done;
  done;
  lynd;;

```

Définition 2

Factorisation de Lyndon Une factorisation de Lyndon d'un mot u est une suite $u^{(1)}, u^{(2)}, \dots, u^{(p)}$ de mots de Lyndon telle que $u = u^{(1)}.u^{(2)}. \dots .u^{(p)}$ et $u^{(p)} \preceq u^{(p-1)} \preceq \dots \preceq u^{(1)}$.

Par exemple, pour $u = 10100101100100$, une factorisation de Lyndon est $1, 01, 001011, 011, 0, 0$.

Question 9 Prouver qu'un mot admet au plus une décomposition de Lyndon.

Par récurrence sur la longueur de u .

Si u est de longueur 1, la seule décomposition possible est u .

On suppose que tout mot de longueur au plus n admette au plus 1 décomposition de Lyndon.

Soit u de longueur $n + 1$ admettant les décomposition de Lyndon

$u = u^{(1)}. \dots .u^{(p)}$ avec $u^{(i)}$ mot de Lyndon et $u^{(i)} \preceq u^{(i-1)}$,

$u = v^{(1)}. \dots .v^{(q)}$ avec $v^{(j)}$ mot de Lyndon et $v^{(j)} \preceq v^{(j-1)}$.

On suppose $|u^{(1)}| < |v^{(1)}|$ donc $u^{(1)}$ est un préfixe propre de $v^{(1)}$: $u^{(1)} \prec v^{(1)}$.

De plus $v^{(1)} = u^{(1)}. \dots .u^{(k)}.w$ avec w préfixe non vide de $u^{(k+1)}$.

w est un suffixe propre de $v^{(1)}$ donc $v^{(1)} \prec w$. w est un préfixe de $u^{(k+1)}$ donc $w \preceq u^{(k+1)}$.

On a $u^{(k+1)} \preceq u^{(k)} \preceq \dots \preceq u^{(1)}$. Ainsi $v^{(1)} \prec u^{(1)}$ ce qui est incompatible avec $u^{(1)} \prec v^{(1)}$.

De même $|u^{(1)}| > |v^{(1)}|$ est impossible donc $|u^{(1)}| = |v^{(1)}|$ puis $u^{(1)} = v^{(1)}$.

$u^{(2)}. \dots .u^{(p)}$ et $v^{(2)}. \dots .v^{(q)}$ sont alors deux décompositions de Lyndon d'un même mot de longueur n au plus, elle sont égales d'après l'hypothèse de récurrence donc les deux décompositions de u sont égales.

Pour construire une factorisation de Lyndon d'un mot $u = u_1 u_2 \dots u_n$ on construit une suite de décompositions de u en mots de Lyndon (mais pas encore une décomposition de Lyndon) à partir de la suite $u^{(1)}, u^{(2)}, \dots, u^{(n)}$ avec $u^{(i)} = u_i$.

À chaque étape $v^{(1)}, v^{(2)}, \dots, v^{(p)}$ de mots de Lyndon telle que $u = v^{(1)}.v^{(2)}. \dots .v^{(p)}$ on cherche s'il existe un indice k tel que $v^{(k)} \prec v^{(k+1)}$: si oui, on remplace la suite par $v^{(1)}, \dots, v^{(k-1)}, v^{(k)}.v^{(k+1)}, v^{(k+2)}, \dots, v^{(p)}$.

S'il n'y en a pas, on a fini : la séquence obtenue est une factorisation de Lyndon.

Question 10 Écrire la fonction `factorisation` qui calcule une factorisation de Lyndon du mot u passé en paramètre et renvoie le résultat sous forme de la liste des facteurs.

```

let factorisation liste =
  let init = List.map (fun x -> [x]) liste in
  let rec diminuer liste =
    match liste with
    | [] -> [], false
    | [u] -> [u], false
    | u::v::reste when inferieur u v -> (u@v)::reste, true
    | t::q -> let q', b = diminuer q in t::q', b in
  let rec traiter liste =
    let liste', b = diminuer liste in
    if b
    then traiter liste'
    else liste' in
  traiter init;;

```

Solutions