Les perles de Dijkstra

Dijkstra s'est posé la question de savoir s'il est possible de construire un collier de perles, avec des perles bleues, blanches ou rouges, de telle manière à ce qu'il n'existe pas deux fois la même séquence de perles répétée consécutivement dans le collier.

On formalise ce problème par la notion de mots sans facteur carré et on explore des méthodes de construction de tels mots.

1 Mots sans facteur carré

Soit Σ un alphabet et u un mot sur Σ. On dit que u contient un *facteur carré* s'il se décompose en $u = xv^2y$ avec x, y, v des mots sur Σ, avec $v \neq \varepsilon$.

On réalisera ce travail en OCaml. On identifie les lettres avec le type `int` et on considère que les mots sont des listes de lettres.

```
type lettre = int;;
type mot = lettre list;;
```

Question 1

Écrire une fonction `est_carre : mot -> bool` testant si un mot est de la forme $u = v^2$, où v est un mot.

Question 2

Écrire une fonction `prefixe_carre : mot -> bool` testant si un mot est de la forme $u = v^2y$, où v et y sont des mots, avec $v \neq \varepsilon$. Étudier sa complexité.

Question 3

En déduire une fonction `facteur_carre : mot -> bool` testant si un mot contient un facteur carré. Étudier sa complexité.

2 Construction exhaustive

Question 4

Écrire une fonction

`tous_les_mots : int -> int -> mot list`

telle que `tous_les_mots k n` construit la liste de tous les mots de de longueur n sur l'alphabet $\Sigma = \{0, 1, \dots, k-1\}$. Quelle est la longueur du résultat ?

Question 5

En déduire une fonction `exhaustif : int -> int -> mot list` construisant la liste de tous les mots sans facteur carré de longueur n sur l'alphabet $\Sigma = \{0, 1, \dots, k-1\}$.

Question 6

Combien y a-t-il de mots sans facteur carré de longueur 3, de longueur 6 et de longueur 8 sur un alphabet ternaire : $\Sigma = \{0, 1, 2\}$?

3 Construction par backtracking

Question 7

Proposer un algorithme de génération d'un mot de longueur n sans facteur carré sur $\Sigma = \{0, 1, \dots, k-1\}$ par une méthode de backtracking.

Question 8

En déduire une fonction `backtracking : int -> int -> mot` construisant un seul mot sans facteur carré de longueur n sur l'alphabet $\Sigma = \{0, 1, \dots, k-1\}$.

Question 9

Écrire une version alternative de cette fonction pour générer tous les mots sans facteur carré.

Vous vérifierez les résultats obtenus à l'aide de la fonction `exhaustif`.

4 Génération par morphismes

Un *morphisme* est une application $f : \Sigma^* \to \Sigma^*$ telle que $f(\varepsilon) = \varepsilon$ et pour tous mots u et v. $f(u.v) = f(u).f(v)$.

Question 10

Justifier qu'un morphisme est entièrement défini par sa restriction à Σ.

Ainsi on définira un morphisme par le type

```
type morphisme = int -> int list
```

On définit le morphisme de Thue-Morse sur $\Sigma = \{0, 1\}$ ainsi :

$$f(0) = 01 \qquad f(1) = 10$$

et la suite de mots définie par $u_0 = 0$ et $u_{i+1} = f(u_i)$.

Question 11

Calculer à la main $u_0, u_1, \dots, u_6$. Que remarque-t-on ?

Question 12

Écrire une fonction `image_morph : morphisme -> mot -> mot` calculant l'image d'un mot par un morphisme.

Question 13

Écrire une fonction `thuemorse : int -> mot` construisant le mot u_i.

Un mot de Thue-Morse ne contient jamais le facteur 111. En comptant entre chaque paire de 0 le nombre de 1 consécutifs, on obtient un mot sur $\{0, 1, 2\}$ qui est sans facteur carré.

Question 14

Écrire une fonction `perles_thuemorse : int -> mot` qui construit un mot sans facteur carré à partir du morphisme de Thue-Morse.

Un autre morphisme intéressant est le morphisme de Leech g défini par :

$$g(0) = 0121021201210 \qquad g(1) = 1202102012021 \qquad g(2) = 2010210120102$$

qui génère directement des mots sans facteur carré.

Question 15

Écrire une fonction `leech : int -> mot` construisant des mots ternaires sans facteur carré.