DM - Le lièvre et la tortue // Éléments de correction

I. Aspects mathématiques

On fixe un entier $n \in \mathbf{N}$ et on considère une fonction $f : E_n \to E_n$ où l'on a posé $E_n = [\![0, n-1]\!]$.

On pose enfin un entier $a \in E_n$ et on considère la suite récurrente (u_n) définie par :

$$u_0 = a \qquad \forall \in \mathbf{N},\ u_{n+1} = f(u_n)$$

Question 1 *Montrer qu'il existe des entiers μ et $\lambda \geq 1$ tels que :*

$$\forall\, k \in \mathbf{N},\ k \geq \mu \implies u_{k+\lambda} = u_k$$

et que les $x_0, x_1, \dots, x_{\mu+\lambda-1}$ sont tous distincts.

Puisque l'on a une infinité de termes prenant un nombre fini de valeurs, il existe deux termes égaux. L'ensemble

$$\{n \in \mathbf{N} \mid \exists\, p \in \mathbf{N}^\star : u_{n+p} = u_n\}$$

est non vide, et on peut définir son minimum μ, ainsi que

$$\lambda = \min\{p \in \mathbf{N}^\star \mid u_{\mu+p} = u_\mu\}$$

On a alors :

$$\begin{aligned} \forall k \in \mathbf{N} k \geq \mu \implies u_{k+\lambda} &= f^{k-\mu}(x_{\mu+\lambda}) \\ &= f^{k-\mu}(x_\mu) \\ &= x_k \end{aligned}$$

De plus, $x_{\mu+\lambda}$ est le premier terme de la suite ayant une valeur déjà présente auparavant, les termes $x_0, \dots, x_{\mu+\lambda-1}$ sont tous distincts.

Question 2 *Montrer qu'il existe un entier $n \in \mathbf{N}^\star$ tel que $x_{2n} = x_n$ et que l'on a alors*

$$\mu = \min\{k \in \mathbf{N} \mid x_{n+k} = x_k\}$$

Pour avoir $x_{2n} = x_n$ avec $n \geq 1$, il faut avoir $n \geq \mu$ et $n - \mu \equiv 2n - \mu\,[\lambda]$, soit $n \equiv 0\,[\lambda]$. Ainsi, tout multiple non nul de λ et supérieur ou égal à λ convient.

Pour un tel n, pour avoir $x_{n+k} = x_k$, il faut et il suffit d'avoir $k \geq \mu$ et $k - \mu \equiv n + k - \mu\,[\lambda]$. La deuxième condition étant vérifiée, cela revient à avoir $k \geq \mu$. En particulier, μ est bien la plus petite valeur vérifiant l'égalité.

Ce résultat nous fournit un algorithme pour déterminer la valeur μ. Il est alors facile de déterminer λ. Il est attribué à Robert Floyd et est parfois appelé algorithme du lièvre et de la tortue.

Question 3 *Écrire une fonction* `floyd : ('a -> 'a) -> 'a -> int * int` *telle qu'étant donnés* `f` *est une fonction sur un ensemble fini E et* `a` *un élément de E, « `floyd f a` » retourne le couple d'entiers (μ, λ) défini précédemment.*

On peut programmer cela de façon procédurale comme fonctionnelle.

```
let floyd_procedural f a =
  let x = ref (f a)
  and y = ref (f (f a)) (* ou f !x *)
  in
  while !x <> !y do
    x := f !x ;
    y := f (f !y)
  done ;
  let mu = ref 0 in
  x := a ;
  while !x <> !y do
    x := f !x ;
    y := f !y ;
    incr mu
  done ;
  let lambda = ref 1 in
  x := f !x ;
  while !x <> !y do
    x := f !x ;
    incr lambda
  done ;
  !mu, !lambda
```

```
let floyd_fonctionnel f a =
  let rec phase1 x y =
    if x <> y
    then phase1 (f x) (f (f y))
    else x
  in
```

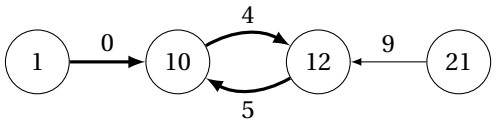
```
let rec phase2 cnt x y =
  if x = y
  then cnt, x
  else phase2 (cnt + 1) (f x) (f y)
in
let rec phase3 cnt x y =
  if x = y
  then cnt
  else phase3 (cnt + 1) (f x) y
in
let xn = phase1 (f a) (f (f a)) in
let mu, xmu = phase2 0 a xn in
let lambda = phase3 1 (f xmu) xmu in
mu, lambda
```

Une qualité remarquable de cet algorithme est que sa complexité temporelle est linéaire et sa complexité spatiale est constante. Une implémentation naïve pourrait nécessiter une complexité spatiale linéaire (on conserve la liste des termes déjà calculés) et une complexité quadratique (à chaque nouveau terme, on regarde s'il a déjà été calculé).

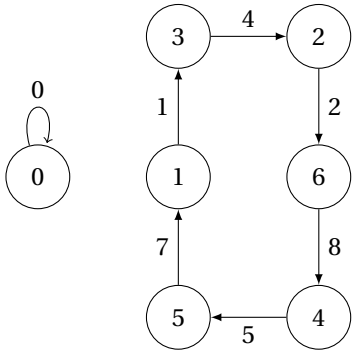
Notons au passage qu'il est également coûteux de calculer x_{2n} à partir de x_n comme $f^n(x_{2n})$ alors que si l'on avait x_{n-1} et x_{2n-2}, il suffit d'appliquer f 3 fois et non n fois.

II. Écriture développée des rationnels

Question 4 *Quel est le développement décimal de* 1/22 *?* Il suffit de partir de 1 et de suivre les flèches.

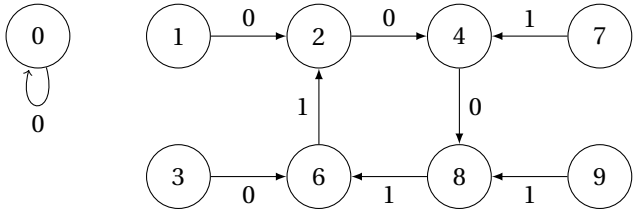


Question 5 *Construire le graphe de division de* 7 *en base* 10.



Question 6

1. *Construire le graphe de division de* 10 *en base* 2.



2. *En déduire le développement en base* 2 *de* 1/10, 2/10 *et* 3/10. On a :

$$\frac{1}{10} = \texttt{0.0(0011)}, \quad \frac{2}{10} = \texttt{0.(0011)}, \quad \frac{3}{10} = \texttt{0.0(1001)}$$

III. Détermination du développement d'un rationnel

Question 7 *En se basant sur l'algorithe de Floyd, écrire une fonction*

```
dvlpt : int -> int -> int -> int list * int list
```

Une manière très simple de procéder est de réutiliser la fonction `floyd` pour récupérer les valeurs μ et λ, puis de construire les listes.

```
let dvlpt a d b =
  (* Calcul de mu et lambda *)
  let f x = (x * b) mod d in
  let mu, lambda = floyd f a in
  (* Calcul de la liste de décimales *)
  let prelude = ref []
  and boucle = ref []
  and x = ref a in
```

```
  for i = 1 to mu do
    let y = !x * b in
    prelude := (y / d) :: !prelude ;
    x := y mod d
  done ;
  for i = 1 to lambda do
    let y = !x * b in
    boucle := (y / d) :: !boucle ;
    x := y mod d
  done ;
  (* Les listes ont été construites à l'envers *)
  List.rev !prelude, List.rev !boucle
```

Question 8 *Donner une suite d'instructions permettant de déterminer la 1 000 000 décimale de* 2020/2021.

Il faut tester si l'on est dans le prélude ou dans la boucle (ou justifier que ce n'est pas nécessaire... [1]), et ne pas oublier que le première décimale est en position 0, et donc que la 1 000 000 ème est en position 999 999.

```
let prelude, boucle = dvlpt 2020 2021 10 in
let pos = 1000000 - 1 in
if pos < List.length prelude
then List.nth prelude pos
else List.nth boucle
  ((pos - List.length prelude) mod (List.length boucle))
```

Question 9 *Quelle est cette décimale ?* Il s'agit de 6.

1. La somme des longueurs du prélude et de la boucle ne peuvent excéder la valeur du dénominateur. Pourquoi ?