Suites itérées

1 Principe des suites itérées

On s'intéresse à une suite itérée $(u_n)_{n\in\mathbb{N}}$, définie par son premier terme u_0 et une relation de récurrence $u_{n+1} = f(u_n)$. La suite de Syracuse, par exemple, est définie par la récurrence $u_{n+1} = f(u_n) = u_n/2$ lorsque u_n est pair, et $u_{n+1} = f(u_n) = 3u_n + 1$ si u_n est impair.

Pour représenter une suite itérée, on définit le type « enregistrement » suivant :

```
type 'a suite = { u0: 'a; f: 'a -> 'a };;
```

Ce type permettra de décrire une suite itérée travaillant sur des u_n de type `'a`. Par exemple, la suite définie par

$$\texttt{seq}\begin{cases} u_0 = 4.9 \\ u_{n+1} = \sin(u_n) \quad \text{pour tout } n \geqslant 0 \end{cases}$$

pourra être déclarée en OCaml par

```
let seq = { u0 = 4.9, f = sin }
```

Pour accéder au premier terme de la suite `seq`, on écrira « `seq.u0` ». Pour accéder à la fonction gouvernant la récurrence de cette même suite, on écrira « `seq.f` ». Ainsi, on peut obtenir u_1 en écrivant « `seq.f seq.u0` » et u_2 en écrivat par exemple « `seq.f (seq.f seq.u0)` ».

1. Définir une fonction `f_syracuse` de signature `int -> int` calculant le passage de u_n à u_{n+1} pour la suite de Syracuse.

2. Y a-t-il un risque d'erreur dans le calcul de u_{n+1} en Caml ?

3. Proposer une fonction `nth` de signature `'a suite -> int -> 'a` qui prend en paramètre une suite $(u_n)_{n\in\mathbb{N}}$ itérée, et un entier n et retourne le terme u_n.

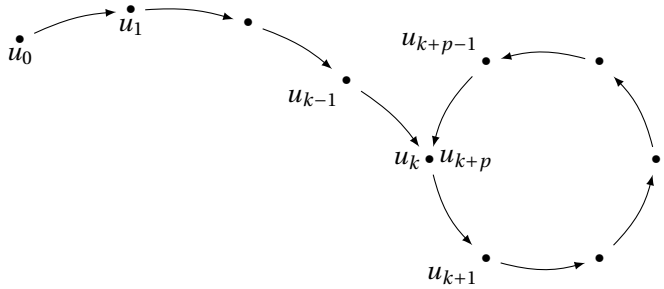
4. Déterminer u_{41} pour la suite de Syracuse dans le cas où $u_0 = 1023$.

5. Écrire une fonction `premiers` de signature `'a suite -> int -> 'a list` qui prend en paramètre une suite itérée et un entier n et retourne la liste des n premiers éléments de la suite $[u_0, u_1, \ldots, u_{n-2}, u_{n-1}]$. Quelle est la complexité de cette fonction ?

6. Construire la liste des 70 premiers termes de la suite de Syracuse pour $u_0 = 1023$. Que constate-t-on ? Essayer avec d'autres valeurs initiales.

2 Suites cycliques

On dit qu'une suite (u_n) est (ultimement) cyclique avec une période $p \in \mathbb{N}^\star$ lorsqu'il existe un entier k tel que, pour tout $n \geqslant k$, $u_{n+p} = u_n$. Dans la suite, on suppose la suite (u_n) cyclique, et on cherche à déterminer les plus petits k et p qui conviennent (que l'on notera simplement dans la suite k et p).



7. Justifier que toute suite itérée d'entiers en OCaml sera cyclique.

8. Montrer que si l'on dispose de la liste des n premiers termes dans l'ordre inverse, $[u_{n-1}, u_{n-2}, \ldots, u_1, u_0]$, on peut vérifier simplement et en temps linéaire si $n - 1 \geqslant k + p$.

9. Écrire une fonction `index` de signature `'a -> 'a list -> int` prenant en argument un élément `el` et une liste d'éléments de même type, et retournant l'index du premier élément de la liste égal à `el`, ou -1 si aucun élément de la liste ne vérifie cette égalité.

10. Proposer une fonction `periode` de signature `'a suite -> int * int` prenant en argument une suite itérée et retournant le couple (k, p) en un temps de l'ordre de $O\left((k+p)^2\right)$.

11. Écrire une fonction `decompose` de signature `'a suite -> 'a list * 'a list` prenant en argument une suite et retournant le couple de listes $[u_0, u_1, \ldots, u_k]$ et $[u_{k+1}, \ldots, u_{k+p}]$.

12. Utiliser la fonction précédente pour différentes valeurs de u_0 pour la suite de Syracuse.

13. Quelle hypothèse peut-on faire ? Il s'agit de la conjecture de Collatz, qui n'a pas encore pu être démontrée.

14. Qu'en est-il en remplaçant $3u_n + 1$ par $3u_n - 1$ (ou d'autres fonctions linéaires de u_n) ?

3 Algorithme de Floyd

Lorsque la prépériode k et la période p sont grands, une recherche de ces valeurs en un temps proportionnel au carré de $(k+p)$ n'est plus suffisamment efficace. Il existe un algorithme linéaire en temps, l'algorithme de Floyd (parfois appelé algorithme du lièvre et de la tortue), qui permet de déterminer ces deux valeurs en temps linéaire.

On définit la suite itérée $(v_n)_{n\in\mathbb{N}}$ par $v_0 = u_0$ et $v_{n+1} = f(f(v_n))$ (suite du « lièvre », car elle « progresse » deux fois plus vite que (u_n))

15. Justifier que pour une suite (u_n) cyclique de période p, il existe une infinité d'entiers i vérifiant $u_i = v_i$, et que le plus petit $i > 0$ pour lequel $u_i = v_i$ est le plus petit multiple (non nul) de p supérieur ou égal à k.

16. Proposer une fonction `floyd` de signature `'a suite -> int` qui prend en argument une suite itérée et retourne le plus petit $i > 0$ vérifiant $u_i = v_i$.

On définit la suite $(u'_n)_{n\in\mathbb{N}}$ par $u'_0 = u_i$ (où i est le plus petit entier strictement positif vérifiant $u_i = v_i$) et $u'_{n+1} = f(u'_n)$.

17. Quel résultat obtient-on en utilisant la fonction `floyd` sur la suite $(u'_n)_{n\in\mathbb{N}}$?

18. Justifier que, si i est le plus petit entier strictement positif vérifiant $u_i = v_i$, alors la prépériode k est le plus petit entier vérifiant $u_k = u_{i+k}$.

19. À partir des questions précédentes, proposer une fonction `periode` dont la signature serait `'a suite -> int` qui, lorsqu'on lui fournit en argument une suite itérée, retourne la plus petite période p.

20. De même, proposer une fonction `preperiode` dont la signature serait `'a suite -> int-> int` qui retourne, lorsqu'on lui fournit une suite itérée et sa période p, la valeur de k.

L'algorithme de Floyd présente l'avantage de ne nécessiter un nombre d'opérations de l'ordre de $O(k+p)$, soit moins que l'approche précédente.

4 Algorithme de Prabekhar

On s'intéresse à présent à la suite sur $\mathbb{N}$ telle que u_{n+1} soit la somme des carrés des chiffres de u_n (algorithme de Prabekhar).

21. Écrire la fonction `f_prabekhar` qui à tout u_n associe u_{n+1}.

22. Y a-t-il un risque d'erreur dans le calcul de u_{n+1} en Caml ?

23. Déterminer k, p et le contenu du cycle pour différentes valeurs de u_0. Qu'en penser ?

24. Justifier qu'un ordinateur peut vérifier la conjecture en testant un nombre fini de u_0 (on pourra notamment justifier que pour tout u_0 ne figurant pas dans la liste, on atteint un nombre de cette liste après une ou plusieurs étapes).

5 Fractions rationnelles

Tout rationnel a/b possède une écriture décimale ultérieurement périodique. Par exemple,

$$19/112 = 0.169642857142857142857\ldots = 0.1696\underline{428571}$$

$$1566/7735 = 0.\underline{2024563671622495151906916612798965740142210730446}$$

Pour simplifier, dans la suite, on suppose $0 < a < b$.

Pour calculer tous les chiffres de a/b, on pourra construire une suite $(u_n)_{n\in\mathbb{N}}$ dans $\mathbb{N}\times\mathbb{N}$, définie par :

- $u_0 = (0, a)$;
- si $u_n = (c_n, a_n)$, alors $u_{n+1} = (\lfloor 10a_n/b \rfloor, (10a_n \mod b))$.

Les c_n ainsi construits correspondent alors à l'écriture décimale de a/b, le calcul de la pré-période et de la période dans l'écriture décimale de a/b revenant à calculer la pré-période et la période de la suite (u_n).

25. Proposer une fonction de signature `int -> int -> int list * int list` prenant en argument deux entiers a et b vérifiant $0 < a < b$ et retournant les chiffres de l'écriture décimale de a/b sous la forme de deux listes d'entiers, la première donnant les chiffres avant la partie périodique (dans l'ordre), la seconde la partie périodique.

26. Vérifier la fonction pour $19/112$ et $1566/7735$

27. Déterminer l'entier b donnant la période la plus longue pour les fractions de la forme $1/b$ avec $1 < b < 10000$.

6 Autres suites ultimement périodiques

Soient a et b deux entiers naturels non nuls. On considère la fonction f définie par

$$f_{a,b} : \begin{cases} [\![0\,..\,a-1]\!] \times [\![0\,..\,b-1]\!] \mapsto [\![0\,..\,a-1]\!] \times [\![0\,..\,b-1]\!] \\ (p,q) \mapsto (pq+1 \mod a, p+q \mod b) \end{cases}$$

28. Proposer une fonction `gen_f` de signature `int -> int -> int*int -> int*int` qui, pour deux paramètres entiers a et b, retourne $f_{a,b}$.

29. Écrire une fonction `cycles` de signature `int -> int -> (int*int) list list` qui prend en argument deux entiers a et b et retourne la liste des cycles des suites itérées utilisant $f_{a,b}$.