(Extrait de X/ENS MP PC 2013))

Points fixes de fonctions à domaine fini

Dans ce problème, on s'intéresse aux points fixes des fonctions $f : E \to E$, où E est un ensemble fini. Le calcul effectif et efficace des points fixes de telles fonctions est un problème récurrent en informatique (transformation d'automates, vérification automatique de programmes, algorithmique des graphes, etc.), et admet différentes approches selon la structure de E et les propriétés de f.

Pour $n \in \mathbb{N}$, on pose $E_n = \{0, \dots, n-1\}$. On représente une fonction $f : E_n \to E_n$ par une liste `L` de taille n, autrement dit $f(x) =$ `L[x]` pour tout $x = 0, \dots, n-1$. Ainsi la fonction f_0, qui à $x \in E_{10}$ associe $2x + 1$ modulo 10 (ie $f(x)$ est le reste de la division euclidienne de $2x + 1$ par 10, donc le chiffre des unités), est représentée par la liste `L` suivante :

`i`	0	1	2	3	4	5	6	7	8	9
`L[i]`	1	3	5	7	9	1	3	5	7	9

I Recherche de point fixe : cas général

On rappelle que x est un point fixe de la fonction f si et seulement si $f(x) = x$.

1. Écrire une fonction `admet_point_fixe(L:list)->bool` qui prend en argument une liste `L` et renvoie `True` si la fonction f représentée par `L` admet un point fixe, `False` sinon. Par exemple, `admet_point_fixe` devra renvoyer `True` pour la liste donné en introduction, puisque 9 est un point fixe de la fonction f_0 qui à x associe $2x+1$ modulo 10.

2. Écrire une fonction `nb_points_fixes(L:list)->int` qui prend en argument une liste `L` et renvoie le nombre de points fixes de la fonction f représentée par `L`. Par exemple, `nb_point_fixes` devra renvoyer 1 pour la liste donné en introduction, puisque 9 est le seul point fixe de f_0.

On note f^k l'itérée k-ième de f, autrement dit

$$\begin{aligned} f^k : E_n &\to E_n \\ x &\mapsto \underbrace{f(f(\dots f}_{k \text{ fois}}(x)) \dots) \end{aligned}$$

3. Écrire une fonction `itere(L:list,x:int,k:int)->int` qui prend en premier argument une liste `L` représentant une fonction f, en deuxième et troisième arguments des entiers x, k de E_n et renvoie $f^k(x)$.

4. Écrire une fonction `point_fixe(L:list)->int` qui prend en argument une liste `L` représentant une fonction f admettant au moins un point fixe, et renvoie un point fixe de f.

Un élément $z \in E_n$ est dit *attracteur principal* de $f : E_n \to E_n$ si et seulement si z est un point fixe de f, et pour tout $x \in E_n$, il existe un entier $k \geqslant 0$ tel que $f^k(x) = z$.

Afin d'illustrer cette notion, on pourra vérifier que la fonction f_1 représentée par la liste ci-dessous admet 2 comme attracteur principal.

`i`	0	1	2	3	4	5	6
`L[i]`	5	5	2	2	0	2	2

En revanche, on notera que la fonction f_0 donnée en introduction n'admet pas d'attracteur principal puisque $f_0^k(0) \neq 9$ quel que soit l'entier $k \geqslant 0$.

5. Écrire une fonction `admet_attracteur_principal(L:list)->bool` qui prend en argument une liste `L` et renvoie `True` si et seulement si la fonction f représentée par `L` admet un attracteur principal, `False` sinon. On ne requiert pas ici une solution efficace mais vous expliquerez clairement la démarche employée.

On suppose aux question **6** et **7** que f admet un attracteur principal. Le *temps de convergence* de f en $x \in E_n$ est le plus petit entier $k \geqslant 0$ tel que $f^k(x)$ soit un point fixe de f. Pour la fonction f_1 ci-dessus, le temps de convergence en 4 est 3. En effet, $f_1(4) = 0$, $f_1^2(4) = 5$, $f_1^3(4) = 2$, et 2 est un point fixe de f_1. On note $\texttt{tc}(f, x)$ le temps de convergence de f en x.

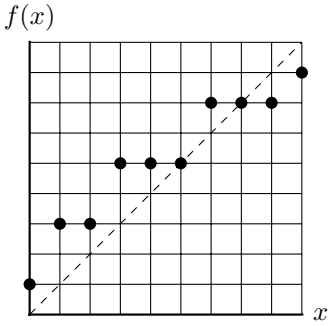
6. Écrire une fonction `temps_cv(L:list)->dico` qui prend en premier argument une liste `L` représentant une fonction f qui admet un attracteur principal, et renvoie un dictionnaire `temps` dont les clefs sont les valeurs de `x` et la valeur associée est $\texttt{tc}(f, x)$. On pourra admettre que $\texttt{tc}(f, x)$ vaut 0 si x est un point fixe de f, et $1 + \texttt{tc}(f, f(x))$ si x n'est pas un point fixe de f. Pour optimiser le temps de calcul, on remplira le dictionnaire de la façon suivante :

- On commence par déterminer le point fixe et on l'ajoute avec son temps de convergence au dictionnaire.
- Si $\texttt{tc}(f, f(x))$ a déjà été calculé et x n'est pas le point fixe, on ajoute au dictionnaire la valeur de $\texttt{tc}(f, x)$
- Sinon, on cherchera le plus petit entier h tel que $\texttt{tc}(f, f^h(x))$ soit connu et on ajoutera au dictionnaire les valeurs de $\texttt{tc}(f, f^j(x))$ pour $j \in [\![0, h]\!]$.

7. Écrire une fonction `temps_cv_max(L)` qui prend en argument une liste `L` représentant une fonction f qui admet un attracteur principal et renvoie $\max_{i=0\dots n-1} \texttt{tc}(f, i)$. On impose un temps de calcul **linéaire** en la taille n de `L`. Justifier la complexité linéaire de cette fonction.

II Recherche efficace de points fixes

Toute fonction `point_fixe(L)` retournant un point fixe d'une fonction arbitraire est de complexité au mieux linéaire en n. On s'intéresse maintenant à des améliorations possibles de cette complexité lorsque la fonction considérée possède certaines propriétés spécifiques.
Le premier (et dernier pour aujourd'hui) cas que nous considérons est celui d'une fonction croissante de E_n dans E_n. On rappelle qu'une fonction $f : E_n \to E_n$ est croissante si et seulement si pour tous $x, y \in E_n$ tels que $x \leqslant y, f(x) \leqslant f(y)$.
On admet qu'une fonction croissante de E_n dans E_n admet toujours un point fixe.
À titre d'exemple, la fonction dont la liste et le graphe sont donnés ci-dessous est croissante. Elle a deux points fixes, à savoir les entiers 5 et 7.



`i`	0	1	2	3	4	5	6	7	8	9
`L[i]`	1	3	3	5	5	5	7	7	7	8

8. Écrire une fonction `est_croissante(L:list)->bool` qui prend en argument une liste `L` et renvoie `True` si la fonction représentée par `L` est croissante, et `False` sinon. On impose un temps de calcul **linéaire** en la taille n de `L`. On ne demande pas de démonstration du fait que le temps de calcul de la solution proposée est linéaire.

9. Écrire une fonction `point_fixe_croissant(L:list)-int` qui prend en argument une liste `L` représentant une fonction croissante f, et retourne un entier $x \in E_n$ tel que $f(x) = x$. On impose un temps de calcul au plus **logarithmique** en la taille n de `L`. On ne demande pas ici de démonstration du fait que le temps de calcul de la solution proposée est logarithmique, ceci étant le sujet de la question suivante.

10. Justifier que le temps de calcul de la fonction de la question **9** est au pire logarithmique en la taille n de `L`. *On pourra admettre que le temps de calcul est une fonction croissante par rapport à la variable n.*