Programmation Python
TP n° 7 : Points fixes de fonctions à domaine fini

Dans ce problème[1], on s'intéresse aux points fixes des fonctions $f : E \to E$, où E est un ensemble fini. On rappelle que x est un *point fixe* de la fonction f si et seulement si $f(x) = x$. 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.
On suppose par la suite un entier strictement positif $n > 0$ fixé et rangé dans une variable globale de même nom, et on pose $\mathsf{E}_n = \{0, \ldots, n-1\}$. On représente une fonction $f : \mathsf{E}_n \to \mathsf{E}_n$ par une liste `t` de taille n, autrement dit $f(x) = \texttt{t[x]}$ pour tout $x = 0, \ldots, n-1$. Ainsi la fonction f_0 qui à $x \in \mathsf{E}_{10}$ associe $2x + 1 \mod 10$ est représentée par la liste `t = [1,3,5,7,9,1,3,5,7,9]`.

1. Recherche de point fixe : cas général

Question 1. Écrire une fonction `admet_point_fixe(t)` qui prend en argument une liste `t` de taille n et renvoie `True` si la fonction $f : \mathsf{E}_n \to \mathsf{E}_n$ représentée par `t` admet un point fixe, `False` sinon. Par exemple, `admet_point_fixe` devra renvoyer `True` pour la liste donnée en introduction, puisque 9 est un point fixe de la fonction f_0 qui à x associe $2x + 1$ modulo 10.

Question 2. Écrire une fonction `nb_points_fixes(t)` qui prend en argument une liste `t` de taille n et renvoie le nombre de points fixes de la fonction $f : \mathsf{E}_n \to \mathsf{E}_n$ représentée par `t`. Par exemple, `nb_point_fixes` devra renvoyer 1 pour la liste donnée en introduction, puisque 9 est le seul point fixe de f_0.
On note f^k l'itérée k-ième de f, autrement dit

$$\begin{aligned} f^k : \mathsf{E}_n &\to \mathsf{E}_n \\ x &\mapsto \underbrace{f(f(\ldots f}_{k \text{ fois}}(x))\ldots) \end{aligned}$$

Question 3. Écrire une fonction `itere(t,x,k)` qui prend en premier argument une liste `t` de taille n représentant une fonction $f : \mathsf{E}_n \to \mathsf{E}_n$, en deuxième et troisième arguments des entiers x, k de E_n, et renvoie $f^k(x)$. On suppose que la valeur de k peut dépasser la profondeur de récurrence maximale, et on impose donc d'implémenter cette fonction de façon *itérative*.

Question 4. Écrire une fonction `nb_points_fixes_iteres(t,k)` qui prend en premier argument une liste `t` de taille n représentant une fonction $f : \mathsf{E}_n \to \mathsf{E}_n$, en deuxième argument un entier $k \geqslant 0$ et renvoie le nombre de points fixes de f^k.
Un élément $z \in \mathsf{E}_n$ est dit *attracteur principal* de $f : \mathsf{E}_n \to \mathsf{E}_n$ si et seulement si z est un point fixe de f, et pour tout $x \in \mathsf{E}_n$, il existe un entier $k \geqslant 0$ tel que $f^k(x) = z$.
Afin d'illustrer cette notion, on pourra vérifier que la fonction $f_1 : \mathsf{E}_7 \to \mathsf{E}_7$, représentée par la liste `[5,5,2,2,0,2,2]`, admet 2 comme attracteur principal. 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 $k \geqslant 0$.

Question 5. Écrire une fonction `admet_attracteur_principal(t)` qui prend en argument une liste `t` de taille n et renvoie `True` si et seulement si la fonction $f : \mathsf{E}_n \to \mathsf{E}_n$ représentée par `t` admet un attracteur principal, `False` sinon. On ne requiert pas ici une solution efficace, mais il est impératif d'expliquer clairement pourquoi votre fonction est correcte.

1. Source : Composition d'informatique, concours d'admission X 2013.

On suppose aux questions 6 et 7 que f admet un attracteur principal. Le *temps de convergence* de f en $x \in \mathsf{E}_n$ 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 $\mathtt{tc}(f, x)$ le temps de convergence de f en x.

Question 6. Écrire une fonction `temps_de_convergence(t,x)` qui prend en premier argument une liste `t` de taille n représentant une fonction $f : \mathsf{E}_n \to \mathsf{E}_n$ qui admet un attracteur principal, en deuxième argument un entier x de E_n, et renvoie le temps de convergence de f en x.

Question 7. Écrire une fonction `temps_de_convergence_max(t)` qui prend en argument une liste `t` de taille n représentant une fonction $f : \mathsf{E}_n \to \mathsf{E}_n$ qui admet un attracteur principal et renvoie $\max_{i=0\ldots n-1} \mathtt{tc}(f, i)$. On impose un temps de calcul linéaire en la taille n de la liste. À titre d'indication, on pourra au besoin créer des listes intermédiaires au cours du calcul. Démontrer que la solution proposée est linéaire.

2. Recherche efficace de points fixes

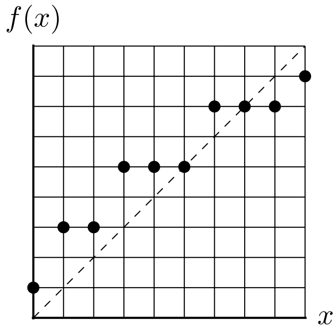
Toute fonction `point_fixe(t)` retournant un point fixe d'une fonction arbitraire est de complexité au mieux linéaire en n. On s'intéresse maintenant à des améliorations possibles de cette complexité lorsque la fonction considérée possède certaines propriétés spécifiques. Nous examinons deux cas.

2.1 Premier cas

Le premier cas que nous considérons est celui d'une fonction croissante de E_n dans E_n. On rappelle qu'une fonction $f : \mathsf{E}_n \to \mathsf{E}_n$ est *croissante* si et seulement si pour tous $x, y \in \mathsf{E}_n$ 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
`t[i]`	1	3	3	5	5	5	7	7	7	8

Question 8. Écrire une fonction `est_croissante(t)` qui prend en argument une liste `t` de taille n et renvoie, en temps linéaire, `True` si la fonction représentée par `t` est croissante, et `False` sinon. Prouver que le temps de calcul de la solution proposée est linéaire.

Question 9. Écrire une fonction `point_fixe(t)` qui prend en argument une liste `t` de taille n représentant une fonction croissante $f : \mathsf{E}_n \to \mathsf{E}_n$, et retourne un entier $x \in \mathsf{E}_n$ tel que $f(x) = x$. On impose un temps de calcul logarithmique en la taille n de la liste.

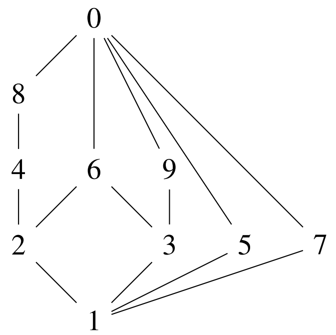
Question 10. Démontrer que la fonction de la question 9 termine, et que le temps de calcul est logarithmique en la taille n de la liste.

2.2 Deuxième cas

On peut généraliser la notion de fonction croissante comme suit. On rappelle qu'une relation binaire $\preccurlyeq$ sur un ensemble E est une relation d'ordre si et seulement si elle est réflexive ($x \preccurlyeq x$ pour tout $x \in E$), anti-symétrique (pour tous $x, y \in E$, si $x \preccurlyeq y$ et $y \preccurlyeq x$, alors $x = y$), et transitive (pour tous $x, y, z \in E$, si $x \preccurlyeq y$ et $y \preccurlyeq z$ alors $x \preccurlyeq z$). Soit $\preccurlyeq$ une relation d'ordre sur E. Une fonction $f : E \to E$ est *croissante au sens de* $\preccurlyeq$ si et seulement si pour tous $x, y \in E$, $x \preccurlyeq y$ implique $f(x) \preccurlyeq f(y)$.
Ceci généralise la notion de fonction croissante de E_n dans E_n, que l'on retrouve en prenant $E = \mathsf{E}_n$ et $\preccurlyeq$ la relation d'ordre $\leqslant$. On s'intéresse dorénavant à d'autres relations d'ordre sur E_n.
On dit qu'un élément m de E est un *plus petit élément* de E au sens de $\preccurlyeq$ si et seulement si, pour tout $x \in E$, $m \preccurlyeq x$. On admet que pour tout ensemble fini E, muni d'une relation d'ordre $\preccurlyeq$ et qui admet un plus petit élément m au sens de $\preccurlyeq$, pour toute fonction croisante $f : E \to E$ au sens de $\preccurlyeq$, il existe un entier $k \geqslant 0$ tel que $f^k(m)$ est un point fixe de f dans E.

Question 11. Soit E un ensemble fini quelconque muni d'une relation d'ordre $\preccurlyeq$ et admettant un plus petit élément m au sens de $\preccurlyeq$. Soit $f : E \to E$ une fonction croissante au sens de $\preccurlyeq$, et soit $k \geqslant 0$ un entier tel que $f^k(m)$ soit un point fixe de f dans E. Démontrer que $f^k(m)$ est en fait le plus petit point fixe de f au sens de $\preccurlyeq$, autrement dit que pour tout autre point fixe x de f dans E, on a $f^k(m) \preccurlyeq x$.

Nous nous intéressons maintenant à un choix particulier d'ordre $\preccurlyeq$, appelé *ordre de divisibilité* et noté $|$. Précisément, on note $a|b$ la relation d'ordre « a divise b » sur les entiers positifs, vraie si et seulement s'il existe un entier $c \geqslant 0$ tel que $ca = b$. Ainsi l'ensemble E_{10} ordonné par divisibilité peut se représenter graphiquement comme suit :



D'après la définition donnée précédemment, une fonction $f : \mathsf{E}_n \to \mathsf{E}_n$ *croissante au sens de l'ordre de divisibilité* est une fonction telle que pour tout x, y dans E_n, si $x|y$ alors $f(x)|f(y)$. Par exemple la fonction représentée par la liste ci-dessous est croissante au sens de l'ordre de divisibilité.

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

On remarque que, par la question 11, toute fonction de E_n dans E_n croissante au sens de l'ordre de divisibilité a un plus petit point fixe au sens de l'ordre de divisibilité.
On rappelle que le pgcd de deux entiers $x \geqslant 1$ et $y \geqslant 1$ est le plus grand entier non nul qui divise x et y. On étend cette définition à des entiers naturels quelconques, en convenant de définir le pgcd d'un entier $x \geqslant 0$ et de 0 comme valant x.

Question 12. Soit f une fonction de E_n dans E_n, croissante au sens de l'ordre de divisibilité, et notons $x_1, \ldots, x_m$ les points fixes de f dans E_n. Montrer que le plus petit point fixe de f au sens de l'ordre de divisibilité est exactement le pgcd de $x_1, \ldots, x_m$.

Question 13. Écrire une fonction `pgcd_points_fixes(t)` qui prend en argument une liste `t` de taille n représentant une fonction de E_n dans E_n, croissante au sens de la divisibilité, et renvoie le pgcd de ses points fixes, en temps logarithmique en la taille n de la liste.

Question 14. Montrer que la fonction de la question 13 a un temps de calcul logarithmique en la taille n de la liste.