Samedi 9 avril 2022

OPTION : SCIENCES DU NUMÉRIQUE

MP / PC / PSI / PT / TSI

Durée : 2 heures

Conditions particulières

Calculatrice interdite
Indiquer uniquement votre code candidat sur le QCM et insérer le dans votre copie d'examen

Concours CPGE EPITA-IPSA-ESME 2022
Option Sciences du Numérique

Consignes Python

Tout code doit être écrit dans le langage Python, à l'exception des requêtes SQL de la section E.

- Veuillez indenter votre code correctement.
- Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué ci-dessous.
- Vous pouvez écrire vos propres fonctions, dans ce cas elles doivent être documentées (on doit savoir ce qu'elles font). Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.

Fonctions et méthodes autorisées

- Itérateurs fournis par la fonction `range`.
- Slicing (extraction de portions) sur les listes.
- La fonction `len` donne la taille d'une liste.
- Les méthodes utilisables sur les listes sont `append` et `pop`, tout autre méthode (`insert`, `sort`...) est exclue.
- Fonctions pour les fichiers :
  - `f=open(fichier, mode)` ouvre un « wrapper » vers un fichier. Le mode peut-être `'r'` (lecture), `'w'` (écriture) ou `'a'` (ajout).
  - `f.close()` ferme le « wrapper ».
  - `f.write(s)` : avec `f` un tel « wrapper » vers un fichier ouvert en écriture ou ajout, écrit dans le fichier la chaîne de caractères `s`.
  - `+` : concaténation de chaînes de caractères.
  - `'\n'` : caractère spécial de saut de ligne.

Introduction

Le sujet de l'option informatique contient six sections, reliées par un thème commun. Les trois dernières sont assez largement indépendantes des trois premières.

Jeu de la vie sur un univers fini

Le *jeu de la vie* est un automate cellulaire imaginé par John Horton Conway en 1970. Malgré sa définition via des règles très simples, il recelle une incroyable complexité. Ce sujet en présente une variante, où l'univers est fini.

L'univers considéré est un quadrillage de dimension $N \times N$, mais les cases de la première ligne sont voisines de celles de la dernière, de même que celles de la colonne de droite sont voisines de celles de la colonne de gauche. Ainsi, chaque case de l'univers possède exactement 8 voisines (voir figure 1), adjacentes horizontalement, verticalement, ou diagonalement.

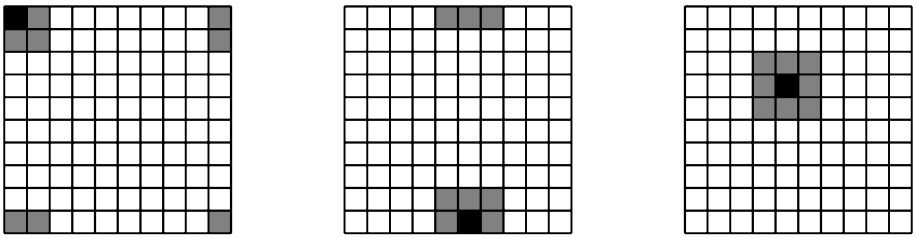

FIGURE 1: Voisins dans le jeu de la vie sur une grille de taille 10 : en noir une case donnée, en gris ses 8 voisines.

Chaque case du jeu de la vie est soit *vivante*, soit *morte*. À partir d'une situation initiale au temps $t = 0$, la situation du jeu évolue entre les instants t et $t + 1$ en suivant les deux règles suivantes.

— une cellule morte possédant exactement trois cellules voisines vivantes devient vivante (elle naît) ;

— une cellule vivante possédant deux ou trois cellules voisines vivantes le reste, sinon elle meurt.

Représentation en Python. Un univers de taille $N \times N$ est représenté par une liste comportant N listes toutes de tailles N. La i-ème liste de `U` (c'est-à-dire `U[i]`, pour $0 \leq i < N-1$) représente la i-ème ligne de la grille en partant du haut, en faisant démarrer l'indexation à zéro. Les colonnes sont également indexées de 0 à $N-1$. La case indicée par (i, j) (c'est-à-dire `U[i][j]`) contient un booléen indiquant la présence d'une cellule vivante. La figure suivante montre un univers de taille 4×4 et sa représentation sous forme de liste de listes.

```
[[True, False, False, False],
 [False, False, True, True],
 [False, False, False, False],
 [True, False, True, False]]
```

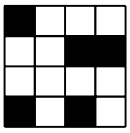

FIGURE 2: Représentation d'un univers par une liste de listes. Les cellules vivantes sont en noir.

Dans toute la suite, on désigne par *univers* une telle liste de listes `U` contenant des booléens, qui sera supposée carrée : avec N la taille de `U`, toutes les listes de `U` ont elles-mêmes une taille N. L'entier N sera supposé toujours au moins égal à 3.

A. Évolution de l'univers

La fonction suivante, renvoyant un univers de taille $N \times N$ contenant uniquement des cases mortes, pourra être utilisée dans la suite, si nécessaire.

```
def univers_mort(N):
    L=[]
    for i in range(N):
        L.append([False]*N)
    return L
```

Question 1. La fonction `random`, sans argument, supposée importée, renvoie un réel aléatoire de l'intervalle $[0, 1[$. Écrire une fonction `init(N)` prenant en entrée un entier strictement positif, et renvoyant un univers $N \times N$ (sous forme de liste de listes), chaque case contenant une cellule vivante avec probabilité 1/3. Par exemple :

```
>>> U = init(4) ; U
[[True, False, False, False], [False, False, True, True], [False, False, False, False],
[True, False, True, False]]
```

Question 2. Écrire une fonction `nb_cellules_vivantes(U)` prenant en entrée un univers `U` et comptant le nombre de cellules vivantes dans l'univers. Cette fonction ne sera pas utilisée dans la suite.

```
>>> nombre_cellules_vivantes(U) #U est la liste prise en exemple en question 1
5
```

Question 3. Écrire une fonction `voisins(i,j,N)` prenant en entrée 3 entiers avec $0 \leq i < N$ et $0 \leq j < N$, et renvoyant la liste des listes de taille 2 d'indices des 8 voisins de (i, j) dans un univers $N \times N$. Remarque : on fera usage du modulo `%` pour ne pas avoir à traiter à part le cas des cases d'un bord. Par exemple :

```
>>> voisins(2,3,4)
[[1, 2], [1, 3], [1, 0], [2, 2], [2, 0], [3, 2], [3, 3], [3, 0]]
```

Votre fonction pourra renvoyer ces listes de taille 2 dans un ordre quelconque.

Question 4. Écrire une fonction `evolue(U)` qui prend en argument un univers à un instant t, et qui renvoie un nouvel univers, résultat de l'évolution de `U` entre les instants t et $t + 1$. *Attention :* vous ne devez pas modifier `U`.

```
>>> evolue(U)
[[True, False, True, True, False], [False, False, False, True], [False, True, True, False],
[False, True, False, True]]
```

Question 5. Quelle est la complexité de `evolue(U)`, en fonction de N, la taille de `U` ?

B. Période et temps d'attraction

Question 6. Combien y a-t-il d'univers possibles (c'est-à-dire de grilles différentes) en taille $N \times N$?

À N fixé, l'ensemble des univers possibles est gros, mais fini. Il s'en suit que lorsqu'on fait évoluer l'univers, à partir d'un certain temps on retombe sur un état déja atteint. À partir de ce moment, l'évolution est périodique. Autrement dit, en notant U_0 l'univers initial, il existe un plus petit r tel que l'univers U_r réapparaîtra, et un plus petit $p > 0$ tel que $U_r = U_{r+p}$, qu'on appelle respectivement temps d'attraction et période (voir figure 3).

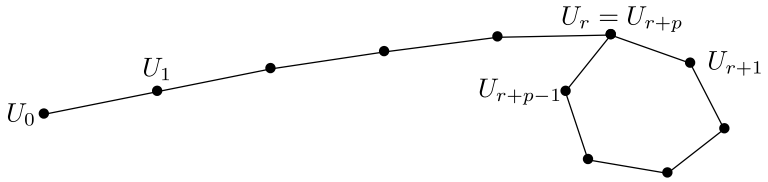

FIGURE 3: Évolution de l'univers : période p et temps d'attraction r

On souhaite calculer temps d'attraction et période dans la suite. La situation se généralise comme suit : on se donne un ensemble fini S et une fonction $f : S \to S$. Un exemple de telle fonction avec $S = [\![0, 8]\!]$ est donné figure 4, avec le *graphe* de f représenté également (un autre exemple est la fonction `evolue`, et S l'ensemble des univers possibles en taille fixée).

x	$f(x)$
0	6
1	6
2	0
3	1
4	4
5	3
6	3
7	4
8	0

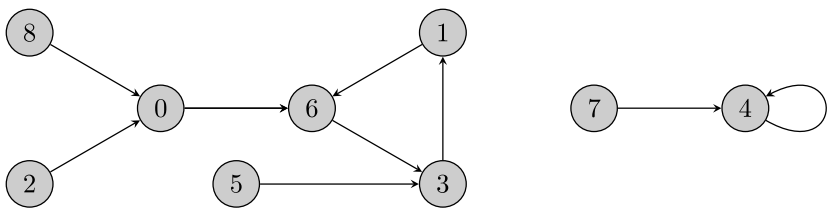

FIGURE 4: Une fonction d'un ensemble fini vers lui-même et sa représentation