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

Jeu de la vie sur un univers fini

A. Évolution de l'univers

Question 1.

```
def init(N):
    U=univers_mort(N)
    for i in range(N):
        for j in range(N):
            if random()<=1/3:
                U[i][j]=True
    return U
```

Question 2.

```
def nombre_cellules_vivantes(U):
    n=len(U)
    c=0
    for i in range(n):
        for j in range(n):
            if U[i][j]:
                c+=1
    return c
```

Question 3.

```
def voisins(i,j,N):
    L=[]
    for di in range(-1,2):
        for dj in range(-1,2):
            if (di!=0 or dj!=0):
                L.append([(i+di)%N, (j+dj)%N])
    return L
```

Question 4. On écrit d'abord une fonction qui donne le nombre de voisins vivants d'une cellule :

```
def nb_voisins_vivants(U,i,j):
    N=len(U)
    L=voisins(i,j,N)
    c=0
    for i in range(len(L)):
        a,b=L[i]
        if U[a][b]:
            c+=1
    return c
```

On fait ensuite évoluer l'univers, en suivant les règles :

```
def evolue(U):
    N=len(U)
    U2=univers_mort(N)
    for i in range(N):
        for j in range(N):
            U2[i][j]=U[i][j]
            c=nb_voisins_vivants(U,i,j)
            if U[i][j]:
                if c!=2 and c!=3:
                    U2[i][j]=False
            elif c==3:
                U2[i][j]=True
    return U2
```

Question 5. Les fonctions `voisins` et `nb_voisins_vivants` sont de complexité constante, donc la complexité de `evolue(U)` est en $O(N^2)$.

B. Période et temps d'attraction

Question 6. Il y a N^2 cases ayant deux états possibles, il y a donc 2^{N^2} univers.

Question 7. 1.

```
def indice(L,x):
    for i in range(len(L)):
        if L[i]==x:
            return i
    return -1
```

2. La complexité est $O(N)$.

Question 8. Par exemple :

```
def rang_periode(f, u0):
    termes=[u0]
    u=f(u0)
    while indice(termes, u)==-1:
        termes.append(u)
        u=f(u)
    r=indice(termes, u)
    return r, len(termes)-r
```

Explication : lorsque la boucle `while` s'arrête, `termes` contient exactement les termes $u_0, u_1, \dots, u_{r+p-1}$, avec r et p les entiers cherchés, et u vaut $u_{r+p} = u_r$. $r + p$ est donc la longueur de `termes`, et `indice(termes,u)` vaut r.

Question 9. Il faut faire $O(r + p)$ tours de boucle. Chaque appel à `indice` a une complexité $O(r + p)$ d'après la question précédente, d'où la complexité totale $O((r + p)^2)$.

C. Calcul efficace : algorithme de Floyd

Question 10. On note $B = \{0\} \cup \{n \in \mathbb{N} \mid n \geq r$ et n est un multiple de $p\}$, et on veut montrer que $A = B$.

- $B \subset A$. Déja, $0 \in A$, et de plus si n est un multiple de p supérieur à r, et s'écrit $n = qp$, on a immédiatement $u_n = u_{n+p} = \cdots = u_{n+qp} = v_n$. d'où $n \in A$. Ainsi $B \subset A$.
- $A \subset B$. Il suffit de montrer qu'un élément qui n'est pas dans B n'est pas dans A. Déja, par définition de r, $[\![1, r-1]\!] \cap A = \emptyset$. De plus, si $n \geq r$ n'est pas un multiple de p, alors sa division euclidienne de n par p s'écrit $n = qp + k$, avec $0 \leq k < p$. On a donc comme précédemment $u_n = u_{n+qp}$, qui ne peut être égal à u_{n+qp+k}, sinon k serait une période de la suite $(u_n)_{n\geq r}$, absurde par définition de p.

On a bien montré que $A = B$.

Question 11. Une condition qui convient est `t==0 or u!=v`.

Question 12. On peut compléter ainsi :

```
p=1
u=f(u)
while u!=v:
    u=f(u)
    p+=1
```

Question 13. 1. Le code suivant convient :

```
for i in range(p):
    w=f(w)
```

2. Enfin, on calcule r :

```
r=0
while u!=w:
    r+=1
    u=f(u)
    w=f(w)
```

Question 14. — La première boucle `while` est en $O(t)$, mais d'après la propriété sur A, $t \leq r+p$. Elle est donc en $O(r+p)$.

— Le code de la question 12 est en $O(p)$.

— Celui de la question 13.1 également.

— Enfin, le code de la question 13.2 est en $O(r)$.

Par suite, la complexité est $O(r+p)$.

D. Les univers ayant les plus longues périodes

Question 15. Par *slicing* par exemple :

```
def separation(L):
    m=len(L)//2
    return [L[:m], L[m:]]
```

Question 16.

```
def fusion(L1, L2):
    L=[]
    i1, i2 = 0, 0
    n1, n2 = len(L1), len(L2)
    for i in range(n1+n2):
        if i1<n1 and (i2==n2 or L1[i1][1]>=L2[i2][1]):
            L.append(L1[i1])
            i1+=1
        else:
            L.append(L2[i2])
            i2+=1
    return L
```

Question 17. Elle est en $O(n \log n)$, avec n la taille de `L`.

E. SQL

Question 18. 1. Nombre d'entrées dans la table `univers` :

```
SELECT COUNT(*) FROM univers
```

2. Liste des couples (i, j) d'indices de cellules vivantes de l'univers appelé `'canon'` :

```
SELECT i,j FROM cellules JOIN univers
ON id = idu
WHERE nom = 'canon'
```

3. Pour chaque période présente dans `univers`, le nombre d'univers de la table ayant cette période :

```
SELECT p, COUNT(*) FROM univers
GROUP BY p
```

4. Liste des couples d'identifiants d'univers différents de même période et même rang d'attraction :

```
SELECT u1.id, u2.id FROM univers u1 JOIN univers u2
ON u1.p = u2.p AND u1.r = u2.r
WHERE u1.id <> u2.id
```

F. Stockage dans un fichier

Question 19. Il est commode d'introduire une fonction prenant en entrée une des listes de `U` et renvoyant la chaîne de caractères à écrire :

```
def ligne(L):
    s=''
    for i in range(len(L)):
        if L[i]:
            s=s+'T'
        else:
            s=s+'F'
        if i<len(L)-1:
            s=s+';'
        else:
            s=s+'\n'
    return s
```

On en déduit la fonction `imprime`.

```
def imprime(U,nom):
    f=open(nom, 'w')
    for i in range(len(U)):
        f.write(ligne(U[i]))
    f.close()
```