Les quatre cycles de Conway

1. Énoncé

Soient u_0 et u_1 deux entiers naturels non nuls.
On définit par récurrence la suite $(u_n)_{n\in\mathbb{N}}$, $n \geqslant 2$ par :

$$u_n = \begin{cases} u_{n-1} + u_{n-2} & \text{si } (u_{n-1} + u_{n-2}) \text{ est premier} \\ \dfrac{u_{n-1} + u_{n-2}}{d} & \text{sinon, où } d \geqslant 2 \text{ est le plus petit diviseur de } (u_{n-1} + u_{n-2}) \end{cases}$$

Conjecture. Quelque soit $(u_0, u_1) \in (\mathbb{N}^*)^2$, il existe un rang $M \in \mathbb{N}$ tel que, ou bien la suite est stationnaire (*i.e.* $\forall k \geqslant M$, $u_k = u_M$) ou bien elle boucle sur :

(a) un cycle de 18 termes contenant 13, 61, 37, 49...

(b) un cycle de 19 termes contenant 17, 43, 30, 73...

(c) un cycle de 56 termes contenant 89, 433, 261, 347...

(d) un cycle de 136 termes contenant 7, 31, 19, 25...

Dans cet exercice, on se propose de programmer cette suite pour tester la conjecture avec quelques valeurs particulières.

1. Montrer que pour tout premier p, la suite est stationnaire pour $k \geqslant 2$ avec $u_0 = p^2 - p$ et $u_1 = p$
2. Implémenter la suite ainsi :
   - u_0 et u_1 sont donnés par l'utilisateur au départ (dans le code ou par un `input`).
   - une variable `N`, initialement égale à 100, représente le nombre de termes calculés.
   - un tableau `S` sauvegarde tous les termes de u_0 à u_{N-1}. Ce tableau est affiché à la fin.
3. Programmer la condition d'arrêt suivante. Si pour un $n \geqslant 3$ donné, il existe un rang $0 \leqslant i < n$ tels que $(u_{i-1}, u_i) = (u_{n-1}, u_n)$, arrêter la boucle. En effet, si l'on trouve une paire de termes consécutifs égale à une autre paire, les prochains termes calculés seront identiques : nous nous situons donc dans une boucle. Passer `N` à 500.
4. En supposant que l'algorithme s'est terminé avant 500 termes, donner les quatre premiers termes du cycle et sa longueur.
5. Tester la conjecture sur une dizaine de paires $(u_0, u_1) \in (\mathbb{N}^*)^2$ bien choisies.

Résultat final souhaité à la question 4. pour $u_0 = 14$ et $u_1 = 16$:

Exécution du code

```
[14, 16, 15, 31, 23, 27, 25, 26, 17, 43, 30, 73, 103, 88, 191, 93, 142, 47, 63, 55, 59,
 ↪  57, 58, 23, 27]

Le cycle commence à u_4 par [23, 27, 25, 26] et sa longueur est de 19
```

2. Solutions

1. $u_2 = \frac{p^2 - p + p}{p} = p$; $u_3 = \frac{p+p}{2} = p$; $u_4 = \frac{p+p}{2} = p$ et ainsi de suite.

2. Il existe bien entendu plusieurs manières de coder pour répondre aux questions posées. Voici un exemple.

quatre-cycles-de-Conway.py

```python
S = [14, 16]   # liste des termes de la suite. Définir les deux premiers

N = 100     # Nombre max d'itérations

def diviseurs(n):  # Renvoie le tableau des diviseurs de n
   D = []
   for k in range(1, int(n+1)):
      if (n%k==0):
         D.append(k)
   return D

def u(v, w):          # v := u_{n-2} et w := u_{n-1}
   D = diviseurs(v+w)
   if(len(D)==2):     # si (v+w) est premier, il a 2 diviseurs exactement
      return (v+w)
   else:
      return( (v+w) // D[1] )    # D[1] = 2e indice, plus petit diviseur différent de 1

for k in range(2, N):
   S.append(u(S[k-2], S[k-1]))

print(S)
```

Exécution du code

```
[14, 16, 15, 31, 23, 27, 25, 26, 17, 43, 30, 73, 103, 88, 191, 93, 142, 47, 63, 55, 59,
 57, 58, 23, 27, 25, 26, 17, 43, 30, 73, 103, 88, 191, 93, 142, 47, 63, 55, 59, 57,
 58, 23, 27, 25, 26, 17, 43, 30, 73, 103, 88, 191, 93, 142, 47, 63, 55, 59, 57, 58,
 23, 27, 25, 26, 17, 43, 30, 73, 103, 88, 191, 93, 142, 47, 63, 55, 59, 57, 58, 23,
 27, 25, 26, 17, 43, 30, 73, 103, 88, 191, 93, 142, 47, 63, 55, 59, 57, 58, 23]
```

3. On ajoute la fonction `dans(S)` qui parcourt les indices pour trouver éventuellement une paire égale à celle formée par les deux derniers termes calculés.

quatre-cycles-de-Conway.py

```python
def dans(S):          # Fonction qui cherche s'il existe déjà deux termes consécutifs
   dedans = False     # ayant la même valeur que les deux derniers termes
   for i in range(3, len(S)):    # on parcourt les indices
      if(S[-2:]==S[i-3:i-1]):    # S[-2:] donne les deux derniers termes
         dedans = True           # S[i-3:i-1] est une paire de termes consécutifs
   return dedans         # Si aucune égalité n'a été trouvée, dedans est resté à False
```

On ajoute une condition d'arrêt dans la boucle. Les dernières lignes de code deviennent :

quatre-cycles-de-Conway.py

```
for k in range(2,N):
   S.append(u(S[k-2], S[k-1]))

   if(dans(S)):
      break

print(S)
```

4. L'affichage demandé peut se coder ainsi.

quatre-cycles-de-Conway.py

```
for i in range(3, len(S)):     # on parcourt les indices
   if(S[-2:]==S[i-3:i-1]):     # S[-2:] donne les deux derniers termes
      break

print("\nLe cycle commence par", S[i-3:i+1], "et sa longueur est de", (len(S)-i))
```

3. D'où vient cette suite ?

John Horton Conway (1937-2020) est un mathématicien britannique de génie, ayant réalisées des avancées majeures dans des branches variés des mathématiques : théorie des jeux, des nœuds, des nombres et même mécanique quantique.

Il est notamment célèbre pour le « Jeu de la Vie », un algorithme aux des règles simples et aux résultats spectaculaires, qui a ouvert un nouveau champ de recherches : les automates cellulaires.

La suite présentée est citée dans sa biographie *Genius at play*[1].

Pour des travaux universitaire, voir : *Conway's subprime Fibonacci sequences* - Richard K. Guy, Tanya Khovanova, Julian Salazar - 2012 - arXiv:1207.5099

John Conway - Aquarelle par le Chat (Mistral) d'après Wikipedia

Vous pouvez tester la suite en ligne à l'adresse
`https://joliesmaths.fr/wp-content/uploads/quatre-cycles-de-Conway.html`

Pour en savons plus sur Conway, voir les excellentes vidéos suivantes :

- « Deux (deux ?) minutes pour John Conway » sur la chaine d'El JJ pour découvrir un panorama de ses travaux. `https://www.youtube.com/watch?v=9Hpy6MKM-J8`
- « Le Jeu de la Vie » sur la chaine de ScienceEtonnante pour une explication spécifique à celui-ci. `https://www.youtube.com/watch?v=S-W0NX97DB0`

1. Siobhan Roberts, *Genius at play : The curious mind of John Horton Conway*, Bloomsbury, 2015 (en anglais)