TP: démonstration d'un théorème avec l'ordinateur
Informatique pour tous

On s'intéresse ici à une suite (u_n) définie par :

$$u_0 = a \qquad \text{et} \qquad \forall\, n \in \mathbb{N},\ u_{n+1} = f(u_n)$$

où a est un entier naturel et f la fonction qui à tout entier naturel associe la somme des cubes de ses chiffres.

Le but de la première partie est de démontrer informatiquement le théorème suivant:

Théorème: si a est un multiple de 3 non nul, alors u_n est, à partir d'un certain rang, constante et égale à 153.

La seconde partie aura pour but de s'intéresser au cas où a n'est pas un multiple de 3.

I Vérification du théorème concernant les multiples de 3

1. Calculer $f(153)$. Que peut-on dire de 153?
2. Écrire f en Python sous forme de fonction **récursive**. Vérifier que $f(9999)$ vaut 2916.
3. Écrire une fonction **récursive** `test` permettant de vérifier le théorème pour une valeur de a.
   Indice: on pourra utiliser `n % 10` et `n // 10` pour obtenir le chiffre des unités et le nombre de dizaines, respectivement.
4. Écrire une fonction **récursive** `verif` de paramètre entier naturel n et qui rend le booléen correspondant à la vérification du théorème pour tout entier naturel non nul a inférieur ou égal à n et divisible par 3. Vérifier que `verif(9999)` est `True`.
   Remarque: Python limite le nombre d'appels récursifs d'une fonction. Si vous obtenez une erreur RecursionError, vous pouvez augmenter cette limite (par exemple à 5000) en exécutant `import sys` puis `sys.setrecursionlimit(5000)`.

On a donc démontré le théorème pour tout $a \leq 9999$.

Soit $a \geq 10000$ et p le nombre de chiffres de a.

5. Justifier que $a \geq 10^{p-1}$.
6. Montrer que $f(a) \leq 729 \times p$.
7. Soit $g(x) = 10^{x-1} - 729x$. Montrer que g est croissante sur $[5, \infty[$.
8. En déduire que $f(a) < a$.
9. Montrer que si a est multiple de 3 alors $f(a)$ est multiple de 3.
   Indice: écrire $a = a_{p-1}...a_0$, c'est à dire $a = \sum a_k 10^k$ et montrer que $a_k^3 \equiv a_k$ modulo 3.
10. En déduire que le théorème est vrai pour tout a multiple de 3.

On a ici un exemple de démonstration à l'aide de l'ordinateur. L'humain s'est chargé d'une partie du raisonnement et a laissé la main à l'ordinateur pour la tâche fastidieuse des dernières vérifications nécessaires.

II Etude générale

Dans cette partie, on cherche un théorème similaire pour des valeurs initiales a non multiples de 3.

1. Déterminer à l'aide de Python les points fixes de la fonction f, c'est à dire les x tels que $f(x) = x$.
2. Écrire une fonction `iteres` qui rend la liste des itérés du paramètre a par la fonction f arrêtée lorsqu'on retombe sur un élément déjà rencontré (l'élément déjà rencontré ne faisant pas partie de la liste).
   Par exemple, `iteres(14)` doit renvoyer `[14, 65, 341, 92, 737, 713, 371]` (l'élément terminal est 371).
3. Déterminer tous les éléments terminaux possibles obtenus par un appel à `iteres`.
4. Parmi les éléments de la question précédente, lesquels sont congrus à 2 modulo 3?
5. En déduire que si a est congru à 2 modulo 3 alors u_n est constante égale à 371 ou 407, à partir d'un certain rang.
6. Quels sont les comportements possibles de u_n si $a \equiv 1$ modulo 3?