Devoir maison n°6 : Nombres heureux

Jules Charlier, Thomas Diot, Pierre Gallois, Jim Garnier
1E1

Partie A - Description de l'algorithme

1) En exécutant l'algorithme pour $n = 12\,345$, on trouve que son image est de 55.

En l'exécutant encore, on confirme bien que $157 \to 75 \to 74 \to 65$.

---

2) A chaque itération, l'algorithme calcule le dernier chiffre r en base 10 du nombre d, ajoute à la somme p le carré de r et pose d comme le d précédant, sans son dernier chiffre en base 10.

L'algorithme calcule donc la somme des carrés des chiffres en base 10 du nombre n.

---

3) a) Les nombres 157, 175, 517, 571, 715 et 751 ont les mêmes chiffres en base 10 : leurs images par la fonction karma sont donc identiques.

b) Soit $p \in \mathbb{N}^*$. Alors le nombre $n_0 = \overline{1...1}$ avec p chiffres 1 est un antécédent de p par la fonction karma. De plus, si $k \in \mathbb{N}^*$, alors le nombre $10^k n_0$ a p chiffres 1 et k chiffres 0 : c'est donc aussi un antécédent de p par karma.

Donc p a une infinité d'antécédents par la fonction karma.

c) Montrons par l'absurde que 157 n'a pas d'antécédant à 3 chiffres par la fonction karma, en supposant que $\overline{abc}$ soit un antécédant de 157. Comme $3 \cdot 7^2 = 147 < 157$, au moins l'un des chiffres de $\overline{abc}$ est supérieur ou égal à 8. Comme l'image par la fonction karma de $\overline{abc}$ est indépendant de l'ordre, supposons sans perte de généralité que $a \geq 8$.

Si $\overline{abc}$ est un antécédant de 157, alors $a^2 + b^2 + c^2 \equiv 7$ modulo 10. Un carré vaut $0, 1, 4, 5, 6$ ou 9 modulo 10, et les sommes de deux carrés modulo 10 sont reportées dans le tableau ci-dessous.

$+$	0	1	4	5	6	9
0	0	1	4	5	**6**	9
1	1	2	5	**6**	7	0
4	4	5	8	9	0	**3**
5	5	**6**	9	0	1	4
6	**6**	7	0	1	2	5
9	9	0	**3**	4	5	8

Si $a = 8$, alors $a^2 \equiv 4$ modulo 10. Il faut donc que $b^2 + c^2 \equiv 3 \bmod 10$ et que b et c soient 9 et 4 (l'ordre n'importe pas). Mais $8^2 + 9^2 + 4^2 = 161 \neq 157$. Donc $a \neq 8$

Si $a = 9$, $a^2 \equiv 1 \bmod 10$, et $b^2 + c^2 \equiv 6 \bmod 10$. Donc b et c sont 0 et 6, ou 1 et 5. Mais $9^2 + 0^2 + 6^2 = 117 \neq 157$ et $9^2 + 1^2 + 5^2 = 107 \neq 157$. Donc $a \neq 9$.

Donc $\overline{abc}$ n'est pas un antécédant de 157, ce qui est une contradiction.

Donc 157 n'a pas d'antécédent à 3 chiffres. Cependant, 8852 est un antécédent à 4 chiffres de 157.

Partie B - Trajectoire des nombres inférieurs à 100

1) Réimplémentation de la fonction `karma` en python et en typst.

```python
def karma(n):
    """Calcule la somme des carrés des chiffres de n"""
    d = n
    p = 0
    while d != 0:
        r = d%10
        d //= 10
        p += r**2
    return p
```

```typst
#let karma(n) = {
  let d = n
  let p = 0
  while d != 0 {
    let r = calc.rem-euclid(d, 10)
    d = calc.div-euclid(d, 10)
    p += calc.pow(r, 2)
  }
  p
}
```

Création de la fonction `heureux` permettant de déterminer si un nombre `n` est heureux ou non :

```python
def heureux(n):
    """
    Renvoie True si le nombre converge dans le puits 1,
    sinon False
    """
    image = karma(n)
    while image != 1:
        # Teste si le nombre est arrivé dans la boucle
        if image == 89:
            return False
        image = karma(image)
    return True
```

```typst
#let heureux(n) = {
  let image = karma(n)
  while image != 1 {
    if image == 89 {
      return false
    }
    image = karma(image)
  }
  true
}
```

Seul un nombre de la boucle (89) est testé pour simplifier le programme, la perte de vitesse engendrée est négligeable.

Pour obtenir la liste des nombreux heureux strictement inférieurs à 100, il suffit d'exécuter le code python `[n for n in range(1, 101) if heureux(n)]`.

On récapitule le résultat dans le tableau suivant :

	0	1	2	3	4	5	6	7	8	9
0	X	**H**	m	m	m	m	m	**H**	m	m
1	**H**	m	m	**H**	m	m	m	m	m	**H**
2	m	m	m	**H**	m	m	m	m	**H**	m
3	m	**H**	**H**	m	m	m	m	m	m	m
4	m	m	m	m	**H**	m	m	m	m	**H**
5	m	m	m	m	m	m	m	m	m	m
6	m	m	m	m	m	m	m	m	**H**	m
7	**H**	m	m	m	m	m	m	m	m	**H**
8	m	m	**H**	m	m	m	**H**	m	m	m
9	m	**H**	m	m	**H**	m	m	**H**	m	m

Il n'existe pas de nombre strictement inférieur à 100 qui ne soit ni heureux ni malheureux.

---

2) On utilise l'algorithme suivant pour calculer la probabilité qu'un nombre choisi au hasard entre 1 et 100 soit heureux :

```python
def probabilite(Nmax):
    """
    Calcule la probabilite qu'un nombre inferieur à Nmax soit heureux
    """
    return (
        [
            heureux(n) for n in range(1, Nmax+1)
        ].count(True)
        / Nmax
    )
```

On obtient une probabilité de 20%.

Partie C - Trajectoires des nombres de n chiffres

Pour des raisons pratiques, nous appelerons par la suite $f(n)$ l'image de $n \in \mathbb{N}$ par la fonction karma.

1) a) Pour un nombre à n chiffres, le nombre ayant la plus grande image est celui composé de n neufs. Soit, pour un nombre N à 3 chiffres, 999. L'image de 999 par la fonction karma étant 243, $f(N)$ est majoré par 243.

Pour $f(f(N)) \in [\![100, 199]\!]$, $f(f(N))$ est majoré par $2 \times 9^2 + 1$, soit 163.

Pour $f(f(N)) \in [\![200, 243]\!]$, $f(f(N))$ est majoré par $2^2 + 3^2 + 9^2$, soit 94.

$f(N)$ étant majoré par 243, on en déduit que $f(f(N))$ est majoré par 163.

b) Soit $n \in [\![100, 163]\!]$. $f(n)$ est majoré par 107.

Soit $n \in [\![100, 107]\!]$. $f(n)$ est majoré par 50.

On a donc montré que tout nombre à trois chiffres, lorsqu'on lui applique la fonction karma suffisamment de fois, finit par donner un nombre à deux chiffres.

Tout nombre à deux chiffres étant toujours soit heureux soit malheureux, et tout nombre à trois chiffres retombant toujours à un nombre à deux chiffres, on en déduit que tout nombre à trois chiffres est ou bien heureux, ou bien malheureux, c'est-à-dire, $P(3)$ est vraie.

---

2) a) Montrons par récurrence que pour tout entier $n \geqslant 4$, on a $n \times 9^2 \leqslant 10^{n-1} - 1$.

- Initialisation, pour $n = 4$:

  $4 \times 9^2 = 324$ et $10^3 - 1 = 999$. L'inégalité est donc vérifiée pour $n = 4$.

- Hérédité : Soit $n \in \mathbb{N}$, avec $n \geqslant 4$. On suppose que :

$$n \times 9^2 \leqslant 10^{n-1} - 1 \Longleftrightarrow 10^{n-1} - 81n - 1 \geqslant 0$$

Montrons que $(n+1) \times 9^2 \leqslant 10^n - 1$.

$$\begin{aligned}
& (n+1) \times 9^2 \leqslant 10^n - 1 \\
\Longleftrightarrow\ & 9 \times 10^{n-1} + 10^{n-1} - 1 - 81n - 81 \geqslant 0 \\
\Longleftrightarrow\ & 10^{n-1} - 81n - 1 + 9(10^{n-1} - 9) \geqslant 0 \\
\Longleftrightarrow\ & 10^{n-1} - 81n - 1 \geqslant -9(10^{n-1} - 9)
\end{aligned}$$

Or $n \geqslant 4$, donc $10^{n-1} \geqslant 1000$. Alors $10^{n-1} \geqslant 9$, i.e. $10^{n-1} - 9 \geqslant 0$.

Donc $-9(10^{n-1} - 9) \leqslant 0$. Or d'après l'hypothèse de récurrence, $10^{n-1} - 81n - 1 \geqslant 0$. Donc l'inégalité $10^{n-1} - 81n - 1 \geqslant -9(10^{n-1} - 9)$ est vérifiée, et par équivalence, l'inégalité $(n+1) \times 9^2 \leqslant 10^n - 1$ l'est aussi.

On a donc montré par récurrence que pour tout entier $n \geqslant 4$, on a $n \times 9^2 \leqslant 10^{n-1} - 1$.

---

3) On réutilise l'agorithme ci-dessus, et on trouve une probabilité de 14.42% pour qu'un nombre entre 1 et 10000 soit heureux.

Partie D - Typst

1) Coder le dernier DM sur la méthode de Cardan, avec de nombreux polynômes donc, en LaTeX nous a définitivement dégouté de sa syntaxe peu lisible et datée de nombreuses années. Nous avons cherché des alternatives et avons trouvé Typst (https://typst.app).

Typst permet de faire la même chose que LaTeX, avec un rendu proche et des options de mise en page assez puissantes. Le language de programmation se rapproche *beaucoup* plus des standards (mélange de Rust et Markdown) et est *beaucoup* plus lisible. Les syntaxes des formules mathématiques sont aussi grandement améliorées. De plus l'exécutable Typst permettant de compiler vers PDF pèse une dizaine de Mo contre plusieurs Go pour LaTeX et compiler un document Typst est *instantané* ; les modifications s'affichent en temps réel.

Nous n'avons donc trouvé que des avantages à migrer vers Typst.

---

2) Quelques exemples de la puissance de Typst utilisée dans ce DM (des exemples plus orientés sur les formules mathématiques sont disponibles sur https://typst.app)

Mise en page complète : l'ensemble des titres avec les numéros sont bien sûr générés automatiquement. Ils sont adaptés au format du DM grâce à notre template (cf code du template sur GitHub).

Programmation puissante : Il est possible de coder des algorithmes directement dans le language Turing-complet de Typst. Nous avons mis l'équivalent en Typst des algorithmes de ce devoir à côté de ceux en Python. Créer automatiquement le tableau récapitulatif est alors une tâche très simple qui est remplie par ce code :

```
table(
    align: center,
    columns: 20pt,
    [X],
    ..array.range(1, 100)
      .map((n) => {
        if heureux(n) [*H*] else [m]
      })
  )
```

Typst a une très bonne intégration avec les logiciels d'écriture du code (VS Code) et de gestion de version (git) grâce à de puissants plugins créés par la communauté : par exemple cliquer sur la preview du document nous amène directement sur l'èlément correspondant dans le code, ou encore créer une nouvelle version du devoir (commit) compile automatiquement tous les documents modifiés (grâce à un script de notre création).

Enfin lorsque Typst ne possède pas nativement une fonctionnalité (le support du français et les encadrés sont natifs par exemple), des packages de base existent déjà. Par exemple, il a suffi d'une ligne d'import (pas d'installation compliquée) pour créer ce QR-code.

Tous nos DMs, code et PDF, sont disponibles sur
github.com/PierreGallois/DMs-LLG