BROUILLON - SOMMER LES CARRÉS DES CHIFFRES D'UN NATUREL

CHRISTOPHE BAL

Document, avec son source LaTeX, disponible sur la page
https://github.com/bc-writing/drafts.

Table des matières

1. Faire une tête au carré à tous les entiers naturels

Voici un procédé facile à faire à l'aide d'une calculatrice. Considérons un entier naturel n, puis calculons la somme de ses chiffres élevés au carré. Ceci nous donne un nouveau naturel auquel on peut appliquer le même procédé. Voici deux exemples.

Exemple 1. *Pour $n = 19$, nous obtenons :*

- $1^2 + 9^2 = 82$
- $8^2 + 2^2 = 68$
- $6^2 + 8^2 = 100$
- $1^2 + 0^2 + 0^2 = 1 \to$ *Rien de nouveau à attendre.*

Exemple 2. *Pour $n = 1\,234\,567\,890$, après $1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2 + 8^2 + 9^2 + 0^2 = 285$ nous obtenons :*

- $2^2 + 8^2 + 5^2 = 93$
- $9^2 + 3^2 = 90$
- $9^2 + 0^2 = 81$
- $8^2 + 1^2 = 65$
- $6^2 + 5^2 = 61$
- $6^2 + 1^2 = 37$
- $3^2 + 7^2 = 58$
- $5^2 + 8^2 = 89$
- $8^2 + 9^2 = 145$
- $1^2 + 4^2 + 5^2 = 42$
- $4^2 + 2^2 = 20$
- $2^2 + 0^2 = 4$
- $4^2 = 16$
- $1^2 + 6^2 = 37 \to$ *Dèjà rencontré.*

Dans le 1^er^ cas, au bout d'un moment le procédé ne produit que des 1. Ce sera le cas dès que l'on commence avec une puissance de 10. Le 2^e^ exemple montre que le mieux que l'on puisse espérer c'est que le procédé devienne périodique à partir d'un moment *(on parle de phénomène ultimement périodique)*.

Date: 6 Juin 2018 – 9 Déc. 2018.

On peut explorer le comportement de ce procédé sur plusieurs valeurs grâce à un programme. Voici un code possible non optimisé [1] écrit en Python 3.7 qui prend un peu de temps pour vérifier que pour tous les naturels $n \in [\![1; 10^6]\!]$, le procédé devient ultimement périodique.

```
NMAX    = 10**6
MAXLOOP = 10**20

for n in range(1, NMAX + 1):
    nbloops = 0
    results = []

    while nbloops < MAXLOOP and n not in results:
        nbloops += 1
        results.append(n)
        n = sum(int(d)**2 for d in str(n))

    if n not in results:
        print(f"Test raté pour n = {n}.")

print("Tests finis.")
```

Une fois lancé, le code précédent affiche juste `Tests finis`. Il reste à voir ce qu'il se passe dans le cas général. La section qui suit démontre que pour tout naturel n, le procédé sera toujours ultimement périodique.

2. Une preuve

Pour un naturel $n = [c_{d-1}c_{d-2}\cdots c_1c_0]_{10} \overset{\text{def}}{=} \sum_{k=0}^{d-1} c_k 10^k$ avec $c_{d-1} \neq 0$, on pose $sq(n) = \sum_{k=0}^{d-1} (c_k)^2$ et $\texttt{taille}(n) = d$ qui sera appelé « *taille de n* ».

Pour $(n\,;k) \in \mathbb{N}^2$, on définit $\boxed{n}_{\,0} = n$ et $\boxed{n}_{\,k} = sq^k(n) \overset{\text{def}}{=} sq \circ sq \circ \cdots \circ sq(n)$ avec $(k-1)$ compositions si $k > 0$. Autrement dit, $\boxed{n}_{\,k+1} = sq\left(\boxed{n}_{\,k}\right)$.

On note enfin $V_n = \left\{ \boxed{n}_{\,k} \mid k \in \mathbb{N} \right\}$ l'ensemble des valeurs prises par la suite $\left(\boxed{n}_{\,k}\right)_k$.

Fait 1. *$\forall n \in \mathbb{N}$, $sq(n) \leqslant 81d$ où $d = \texttt{taille}(n)$.*

Preuve. *Si $n = [c_{d-1}c_{d-2}\cdots c_1c_0]_{10}$ alors $sq(n) = \sum_{k=0}^{d-1}(c_k)^2 \leqslant \sum_{k=0}^{d-1} 9^2 = 81d$.*

Fait 2. *$\forall n \in \mathbb{N}$, notant $d = \texttt{taille}(n)$, nous avons les résultats suivants :*

(1) Si $d \geqslant 4$ alors $\texttt{taille}(sq(n)) < \texttt{taille}(n)$.

(2) Si $d \leqslant 3$ alors $\texttt{taille}(sq(n)) \leqslant 3$.

1. Voir la section proposant des graphiques pour découvrir un code qui explore un peu plus efficacement les périodes.

Preuve. *Notons que $n \geqslant 10^{d-1}$. Le comportement des fonctions 10^{x-1} et $81x$ sur $\mathbb{R}_+^*$ assure l'existence d'un naturel D tel que $\forall d \in \mathbb{N}$, $d \geqslant D$ implique $10^{d-1} > 81d \geqslant sq(n)$. On a même beaucoup mieux : si $10^{D-1} > 81D \geqslant sq(n)$ alors $d \geqslant D$ implique $10^{d-1} > 81d \geqslant sq(n)$.*

Comme $10^3 > 81 \times 4$, nous avons sans effort le 1er point (rappelons que $10^k > n$ implique que n admet au plus $(k-1)$ chiffres).

Pour $d \leqslant 3$, le 2nd point découle de $sq(999) = 243$, $sq(99) = 162$ et $sq(9) = 81$.

Fait 3. $\forall n \in \mathbb{N}$, *l'ensemble V_n est fini et donc la suite $\left(\boxed{n}_{\,k}\right)_{k\in\mathbb{N}}$ est ultimement périodique, i.e. périodique à partir d'un certain rang.*

Preuve. *Le 2nd point dépend directement du 1er point via le principe des tiroirs et la définition récursive de la suite $\left(\boxed{n}_{\,k}\right)_k$.*

Pour le 1er point, il suffit de montrer que $V_n \subset \left[0; 10^{\texttt{taille}(n)}\right]$ pour $n \geqslant 4$ via une petite récurrence descendante finie, et pour $n \leqslant 3$ on a directement $V_n \subset [0; 10^3]$.

3. Coder - Étudier la « période » d'un naturel

Quand il ne se fige pas, le code suivant donne la *« période »* d'un naturel auquel on applique le procédé.

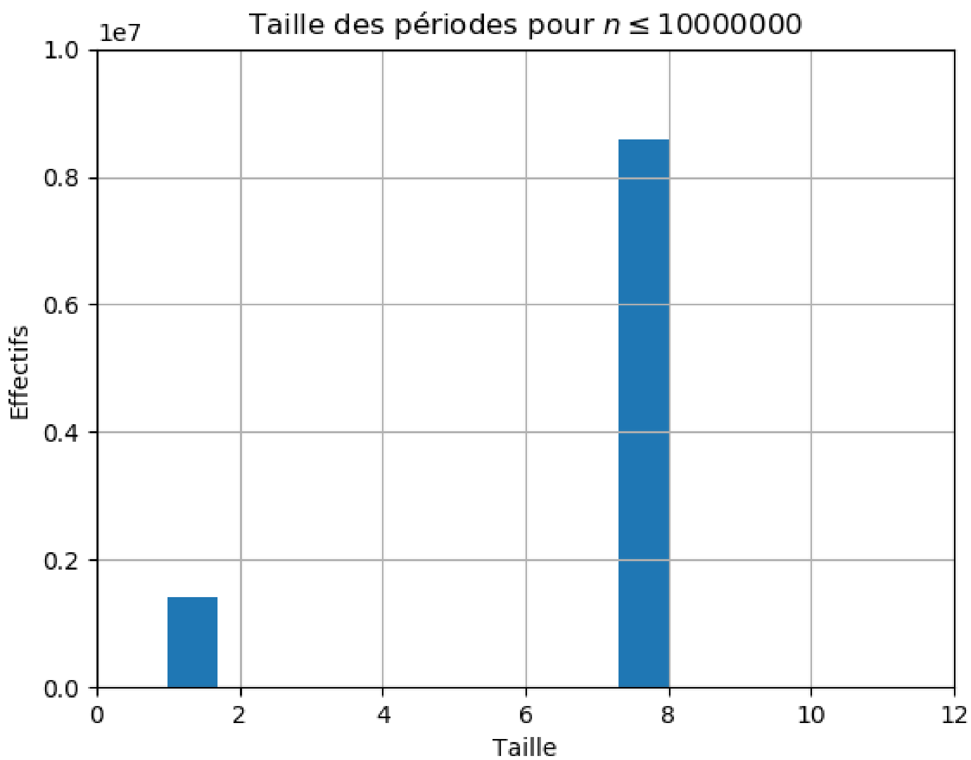
```
n     = 20181209
nmemo = n

results = []

while n not in results:
    results.append(n)
    n = sum(int(d)**2 for d in str(n))

print(f"{nmemo} a la période suivante :")
print(results[results.index(n):])

print()

before = results[:results.index(n)]

if before:
    print("Avant la 1ère période nous avons :")
    print(before)
else:
    print("On commence directement par la période.")
```

Le code précédent, où `n = 20181209`, nous affiche :

```
20181209 a la période suivante :
[16, 37, 58, 89, 145, 42, 20, 4]
```

```
Avant la 1ère période nous avons :
[20181209, 155, 51, 26, 40]
```

Amusons-nous maintenant à représenter un histogramme des effectifs pour la taille des « périodes ». Le code utilisé se trouve à l'adresse `https://github.com/bc-writing/drafts` : voir le fichier `squareint-sizeplots.py` dans le dossier `squares-int`. Le traitement des données a été amélioré pour éviter de refaire des calculs déjà rencontrés *(pour plus de précisions, le lecteur curieux se reportera aux commentaires du code)*. Voici ce que l'on obtient [2].



Voilà quelque chose de frappant ! Il semblerait que l'on ait soit des périodes de taille 1, penser à 0 et 1, soit des périodes de taille 8 comme pour $37 - 58 - 89 - 145 - 42 - 20 - 4 - 16 \ldots$ Magie ou coïncidence ? La section 2 va nous permettre de le savoir. Dans la suite, nous reprenons les notations de la dite section.

Tout d'abord, comme $\texttt{taille}(sq(n)) < \texttt{taille}(n)$ dès que $\texttt{taille}(n) \geqslant 4$ d'après le fait 2, la périodicité n'arrivera que lorsque $\texttt{taille}\left(\boxed{n}_{\,k}\right) \leqslant 3$.

De plus, nous savons aussi que $\texttt{taille}(sq(n)) \leqslant 3$ dès que $\texttt{taille}(n) \leqslant 3$.

Tout ceci nous permet d'analyser brutalement via un programme ce qu'il se passe pour les périodes des naturels appartenant à $[\![0; 999]\!]$. Nous pouvons pour cela utiliser le code suivant, qui n'est absolument pas optimisé mais fait le travail immédiatement.

2. À l'adresse `https://github.com/bc-writing/drafts` dans le dossier `squares-int` vous trouverez l'image `befores.png` qui est un histogramme des effectifs pour la taille des suites de naturels juste avant la 1re « période ».

```
nmax = 999

periodsfound = set()

for n in range(nmax + 1):
    results = []

    while n not in results:
        results.append(n)
        n = sum(int(d)**2 for d in str(n))

    periodsfound.add(
        tuple(results[results.index(n):])
    )

for oneperiod in periodsfound:
    print(oneperiod)
```

Ce code nous fournit les informations suivantes :

```
(20, 4, 16, 37, 58, 89, 145, 42)
(0,)
(1,)
(58, 89, 145, 42, 20, 4, 16, 37)
(16, 37, 58, 89, 145, 42, 20, 4)
(37, 58, 89, 145, 42, 20, 4, 16)
(4, 16, 37, 58, 89, 145, 42, 20)
(89, 145, 42, 20, 4, 16, 37, 58)
(42, 20, 4, 16, 37, 58, 89, 145)
(145, 42, 20, 4, 16, 37, 58, 89)
```

Et là cela devient joli car nous avons non seulement la preuve que les périodes sont de taille 1 ou 8, mais nous pouvons aussi affirmer que soit 0, soit 1, soit 4 apparaîtra à un moment donné du procédé, et qu'ensuite arrivera le phénomène de périodicité. Que c'est beau !