Nombres de Harshad

Année 2017 – 2018

Élèves chercheurs : Alain Michalopoulos, Florian Guiheux, Théo Leroy , Edouard Ly, Mathis Denis (classe de seconde), Guillaume Alexis et Agathe Moreaux (terminale S)

Enseignants : Driss Badaoui et Jérémie Saint-Blanquet

Etablissements : Lycée de Carquefou et Lycée La Herdrie (Basse Goulaine)

Chercheuse : Caroline Robet, Laboratoire Jean Leray, Université de Nantes.

Présentation du sujet :

Des nombres un peu particuliers…

Un entier naturel est un nombre H s'il est divisible par la somme de ses chiffres. Par exemple 18 : $1 + 8 = 9$ et 18 est divisible par 9 donc 18 est un nombre H. En revanche, pour 17, on a $1 + 7 = 8$ et 17 n'est pas divisible par 8 donc n'est pas un nombre H.

— Recherche de ces nombres

— Propriétés sur ces nombres ?

— Algorithme permettant de trouver ces nombres

— Nombres H premiers ?

— Nombres H consécutifs ?

Introduction

Définition :

Les nombres de Harshad (ou nombre H) sont caractérisés par leur divisibilité par la somme de leurs chiffres. Autrement dit (1)

$$\sum_{k=0}^{n} a_k 10^k \equiv 0 \left[\sum_{k=0}^{n} a_k\right]$$

avec k entier naturel, désignant le rang du chiffre a. a_0 est le chiffre des unités, a_n est le chiffre le plus à gauche.

Exemple :

18 est un nombre H. La somme des chiffres fait 9, et 18 est divisible par 9.

Recherche d'algorithmes

Détermination des nombres H :

```
VARIABLES
  H EST_DU_TYPE NOMBRE
  S EST_DU_TYPE NOMBRE
  P EST_DU_TYPE NOMBRE
  J EST_DU_TYPE NOMBRE
  N EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE J
  LIRE N
  P PREND_LA_VALEUR 1
  S PREND_LA_VALEUR 0
  POUR H ALLANT_DE J A N
    DEBUT_POUR
    TANT_QUE (P<N) FAIRE
      DEBUT_TANT_QUE
      S PREND_LA_VALEUR S+(floor(H/P)-10*floor(H/(10*P)))
      P PREND_LA_VALEUR P*10
      FIN_TANT_QUE
    P PREND_LA_VALEUR 1
    SI (H%S==0) ALORS
      DEBUT_SI
      AFFICHER H
      FIN_SI
    S PREND_LA_VALEUR 0
    FIN_POUR
FIN_ALGORITHME
```

Algobox

```python
J= int (input ("Entrez le début : "))
N= int (input ("Entrez la fin : "))
P=1
S=0
for H in range (J,N) :
    while P<N :
        S=S+(int(H/P)-10*int(H/(P*10)))
        P=P*10
    P=1
    if H%S==0 :
        print (H)
    S=0
```

Python

Dans ces algorithmes, l'utilisateur peut choisir l'intervalle dans lequel déterminer les nombres H (entre *J* et *N*). La boucle tant que permet d'isoler et d'additionner un par un, les chiffres des nombres à tester. La boucle si permet de poser une condition : il faut que le reste dans la division euclidienne du nombre par la somme de ses chiffres soit nul pour que l'algorithme affiche ce nombre H (2).

Vérification d'un nombre H :

Le principe est ici relativement le même, si ce n'est qu'au lieu de tester une plage de nombres, l'algorithme teste seulement un nombre.

```
VARIABLES
  H EST_DU_TYPE NOMBRE
  S EST_DU_TYPE NOMBRE
  P EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE H
  P PREND_LA_VALEUR 1
  S PREND_LA_VALEUR 0
  TANT_QUE (P<=H) FAIRE
    DEBUT_TANT_QUE
    S PREND_LA_VALEUR S+(floor(H/P)-10*floor(H/(10*P)))
    P PREND_LA_VALEUR P*10
    FIN_TANT_QUE
  SI (H%S==0) ALORS
    DEBUT_SI
    AFFICHER "Oui"
    FIN_SI
    SINON
      DEBUT_SINON
      AFFICHER "Non"
      FIN_SINON
FIN_ALGORITHME
```

Algobox

```python
H= int (input ("Entrez le nombre à vérifier : "))
P=1
S=0
while P<=H :
    S=S+(int(H/P)-10*int(H/(P*10)))
    P=P*10
if H%S==0 :
    print ("Oui")
else :
    print ("Non")
```

Python

Recherche de propriétés des nombres H

Nombres H jusqu'à 1000
Les nombres H consécutifs sont surlignés en jaune

```
***Algorithme lancé***
1 2 3 4 5 6 7 8 9 10 12 18 20 21 24 27 30 36 40 42 45 48 50 54 60 63 70 72 80 81 84 90 100 102 108 110 111 112 114 117 120 126 132 133 135 140 144 150 152 153 156 162 171 180 190 192 195 198 200 201 204 207
209 210 216 220 222 224 225 228 230 234 240 243 247 252 261 264 266 270 280 285 288 300 306 308 312 315 320 322 324 330 333 336 342 351 360 364 370 372 375 378 392 396 399 400 402 405 407 408 410 414 420 423 432 440 441 444 448 450 460 465 468 476 480 481
486 500 504 506 510 511 512 513 516 518 522 531 540 550 552 555 558 576 588 592 594 600 603 605 612 621 624 629 630 640 644 645 648 660 666 684 690 700 702 704 711 715 720 730 732 735 736 738 756 770 774 777 780 782 792 800 801 803 804 810 820 825 828 832
840 846 864 870 874 880 882 888 900 902 910 912 915 918 935 936 954 960 966 972 990 999 1000

***Algorithme terminé***
```

- *Les seuls nombres premiers et H sont 2, 3, 5 et 7*

Démonstration :

Un nombre premier a seulement deux diviseurs distincts dans ℕ : 1 et lui-même. Or, un nombre H est divisible par la somme de ses chiffres. Donc, pour qu'un nombre premier soit H, il faut que la somme de ses chiffres soit égale à lui-même ou 1. Cela n'est possible qu'avec les nombres à un chiffre. 2, 3, 5 et 7 sont donc les seuls nombres H premiers.

- *Le produit d'un nombre H et d'une puissance de 10 est aussi un nombre H*

Démonstration :

En multipliant par une puissance de 10, la somme des chiffres d'un nombre ne change pas. On peut donc affirmer :

$$\sum_{k=0}^{n} a_k 10^k \equiv 0 \left[\sum_{k=0}^{n} a_k\right] \Leftrightarrow 10^l * \sum_{k=0}^{n} a_k 10^k \equiv 0 * 10^l \left[\sum_{k=0}^{n} a_k\right]$$

$$\sum_{k=0}^{n} a_k 10^k \equiv 0 \left[\sum_{k=0}^{n} a_k\right] \Leftrightarrow 10^l * \sum_{k=0}^{n} a_k 10^k \equiv 0 \left[\sum_{k=0}^{n} a_k\right]$$

Conséquence :

Il existe une infinité de nombres H pairs.

- *Quand un nombre H est composé de n chiffres avec n pair, il possède au maximum $n-1$ chiffres impairs.*

Démonstration par l'absurde:

Si H est composé d'un nombre pair de chiffres, alors $\sum_{k=0}^{n} a_k$ est obligatoirement paire (3). De plus, on suppose que H possède n chiffres impairs, donc son dernier chiffre est impair, et H est impair. Or, un nombre impair n'est jamais divisible par un nombre pair. On aboutit ainsi à une absurdité : H ne peut pas posséder n chiffres impairs, seulement $n-1$ maximum, si n est pair.

- *La somme ou le produit de nombres H, consécutifs ou non, entre eux n'est pas toujours un nombre H.*

Démonstration par contre-exemple :

27 et 30 sont des nombres H. Or 27+30=57 n'est pas un nombre H.
12 et 81 sont des nombres H. Or 12+81=93 n'est pas un nombre H.

- *Mettre un nombre H en puissance ne donne pas toujours un nombre H*

Démonstration par contre-exemple :

27 est un nombre H. Or, $27^2=729$ n'est pas un nombre H
63 est un nombre H. Or, $63^3=250047$ n'est pas un nombre H.

- *Si l'on considère deux nombres H comme étant les longueurs de deux côtés d'un triangle rectangle, la longueur du troisième côté n'est pas toujours un nombre H*

Démonstration par contre-exemple :

27 et 12 sont des nombres H. Or $(27^2+12^2)^{1/2}$ n'est pas un nombre entier, donc pas un nombre H (4).

Conjecture :

- Les nombres composés de 3 fois, 9 fois et 11 fois le même chiffre sont des nombres H (5).

Recherche d'une séquence additive permettant de déterminer les nombres H

```
J= int (input ("Entrez le début : "))
N= int (input ("Entrez la fin : "))
P=1
S=0
M=1
I=1
liste = []
liste2 = []
for H in range (J,N) :
    while P<N :
        S=S+(int(H/P)-10*int(H/(P*10)))
        P=P*10
    P=1
    if H%S==0 :
        liste.append(H)
    S=0
    I=I+1
while M<I :
    V=liste[M+1]-liste[M]
    print (V, end=' ')
    M=M+1
```

Algorithme Python qui détermine le « pas » (nombre à additionner) entre chaque nombre H (6)

Résultats :

On conjecture qu'il n'existe pas de séquence additive permettant de déterminer les nombres H.

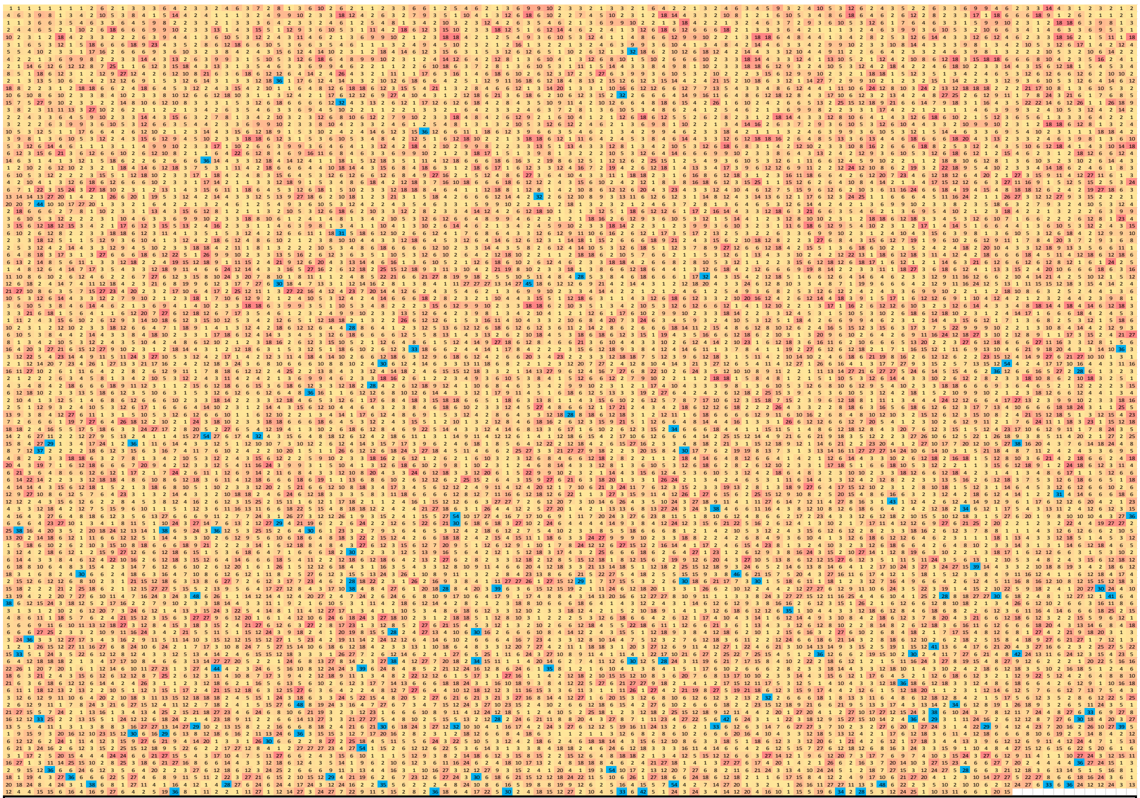

Tableau présentant la liste des pas des nombres H
En dégradé jaune (=1) vers rouge (=27), tous les pas inférieurs à 27
En bleu, tous les pas supérieurs à 27
54 est le pas le plus grand trouvé entre deux nombres H

On conjecture que la densité des nombres H diminue lorsque les nombres tendent vers l'infini. En effet, la moyenne des 500 premiers pas est de 4,97 environ alors que la moyenne des 8000 premiers pas est de 7,60 environ. L'écart entre deux nombres H semble donc s'agrandir.

On pourrait ici essayer de définir une fonction de densité des nombres H, comme cela a été fait pour les nombres premiers…(7)

Conclusion

Les nombres H sont caractérisés par leur divisibilité par la somme de leurs chiffres. On peut les déterminer avec des algorithmes sur Python ou Algobox. Les seuls nombres H premiers sont 2, 3, 5 et 7. De plus, ces nombres possèdent quelques autres propriétés, comme le fait qu'ils forment un ensemble infini. Finalement, il semblerait qu'il y a de plus en plus d'écart entre deux nombres H, cette conjecture est à vérifier avec une fonction de densité…

Notes d'édition

(1) La notation $H \equiv 0[S]$ (H est congru à 0 modulo S) signifie que $H - 0 = H$ est divisible par S.

(2) Dans ces algorithmes, le k-ième chiffre de H (le nombre testé) est obtenu comme la partie entière de $H/10^k$ moins 10 fois la partie entière de $H/10^{k+1}$, les puissances de 10 étant calculées par multiplications successives. Le reste de la division euclidienne de H par S est codé H%S.

(3) On suppose ici par l'absurde que tous les chiffres sont impairs, mais bien sûr cette affirmation n'est pas vraie en général.
Il y a aussi une imprécision car $\sum_{k=0}^{n} a_k$ correspond à la somme des chiffres d'un nombre à $n + 1$ chiffres.

(4) On trouve aussi des exemples où la longueur du 3e côté est un entier mais pas un nombre H :
$5^2 + 12^2 = 13^2$, $9^2 + 12^2 = 15^2$; 5, 9 et 12 sont des nombres H mais 13 et 15 non.

(5) Comme $aaa = a \times 111$ et que la somme de ses chiffres est $a \times 3$, pour vérifier que c'est un nombre H, il suffit de vérifier que 111 est divisible par 3, ce qui est bien le cas. De même 111111111 est divisible par 9 car la somme de ses chiffres est 9, et tous les nombres dont l'écriture décimale consiste en 9 fois le même chiffre sont des nombres H.
Par contre $11111111111 = 11 \times 1010101010 + 1$ n'est pas divisible par 11, donc n'est pas un nombre H (et aucun nombre dont l'écriture décimale consiste en 11 fois le même chiffre n'est un nombre H).
Mais on peut trouver d'autres exemples de nombres H dont tous les chiffres sont égaux…

(6) Il s'agit bien sûr des pas entre chaque nombre H et le suivant. L'algorithme recherche les nombres H entre J et N comme précédemment, dans l'ordre naturel, et les stocke dans une liste ; il calcule ensuite les différences successives entre les nombres consécutifs de la liste et les affiche.

Par ailleurs l'expression *séquence additive permettant de déterminer les nombres H*, dans le titre du paragraphe et dans le résultat, n'est pas entièrement claire. Il s'agit probablement d'un algorithme qui permettrait de prévoir cette suite de différences sans avoir d'abord à déterminer la liste des nombres H.

(7) La *fonction de densité* naturelle est la proportion de nombres H entre 1 et n, qui est fonction de n et qui, d'après les calculs faits, semble décroître lorsque n croît. Il s'agit donc plutôt de donner une estimation de cette densité à l'aide de fonctions connues, comme il en existe une pour la densité des nombres premiers.