La persistance des nombres

Année 2015-2016

Auteurs : ABDOULI Ismaël et LECONTE Alexis, élèves de Terminale S.
Établissement : Lycée Condorcet, Saint-Quentin (02).
Encadrés par : Fabien AOUSTIN.
Chercheur : Fabien DURAND, Laboratoire Amiénois de Mathématique Fondamentale et Appliquée, LAMFA, UMR CNRS 7352, Université de Picardie Jules Verne.

1. Introduction :

Chaque nombre entier naturel a une persistance multiplicative. Celle-ci consiste à multiplier les chiffres du nombre entre eux et de répéter ce procédé avec le produit obtenu. Ce procédé prend fin lorsque le produit obtenu est un nombre à un chiffre (aucune multiplication possible). Le nombre de produits alors obtenus correspond à la persistance du nombre de départ.
Prenons en exemple le nombre 47 :

$4\times 7 = 28$ (1^{er} produit)
$2\times 8 = 16$ ($2^{ème}$ produit)
$1\times 6 = 6$ ($3^{ème}$ produit)

Or 6 est inférieur à 10 donc **la persistance multiplicative du nombre 47 est de 3**.
Nous nous sommes alors demandé quel nombre possède la plus grande persistance multiplicative, s'il existe.

Au cours de nos recherches, nous avons démontré que la persistance multiplicative d'un nombre ne pouvait pas être infinie. De plus, nous avons pu conjecturer que la plus grande persistance était de 11. Enfin, nous avons travaillé dans d'autres bases avec le même problème et nous sommes parvenus à une conjecture en base 3 : la persistance ne dépasse pas 3.

2. La persistance n'est pas infinie :

Au début de nos recherches, nous avons essayé de calculer la persistance de plusieurs nombres pour avoir une meilleure vision du problème. Nous nous sommes alors demandé si un nombre pouvait ne pas avoir de persistance. Grâce à la démonstration qui suit, nous avons montré qu'un tel nombre n'existait pas.
Démonstration :
Soit $n = a_m a_{m-1} a_{m-2} \ldots a_2 a_1 a_0$ un nombre écrit en base 10 avec a_k un chiffre, pour k entier de l'intervalle $[0\,;m]$
La décomposition de ce nombre donne :

$$n = \sum_{k=0}^{m} a_k 10^k$$

La première étape de la persistance de ce nombre donnerait

$$\prod_{k=0}^{m} a_k$$

On veut montrer que,

$$n > \prod_{k=0}^{m} a_k$$

c'est-à-dire que le procédé est strictement décroissant.

On peut écrire : $a_m \times 10^m > \prod_{k=0}^{m} a_k$ car $10 > a_k$ pour k compris entre 0 et $m-1$.

On a donc :

$$a_m \times 10^m > \prod_{k=0}^{m} a_k \Rightarrow a_m \times 10^m + a_{m-1} \times 10^{m-1} + \cdots + a_0 \times 10^0 > \prod_{k=0}^{m} a_k$$

$$\Rightarrow \sum_{k=0}^{m} a_k > \prod_{k=0}^{m} a_k$$

$$\Rightarrow n > \prod_{k=0}^{m} a_k .$$

Le procédé est strictement décroissant donc il s'arrête.

3. Approche algorithmique :

3.1. Nos premiers algorithmes :

Pour mieux visualiser le problème et générer beaucoup de nombres, nous avons utilisé des algorithmes qui nous permettaient de générer des nombres et de trouver leur persistance.
Nous avons utilisé le logiciel Xcas.

L'algorithme ci-dessous calcule la persistance d'un nombre donné.

```
Persistance(y):={ //on demande un nombre
  local c;
  c:=0; // "c" est le nombre d'étapes
  tantque y>9 faire //le programme s'arrête quand le nombre n'a plus qu'un seul chiffre
    y:=product(convert(y,base,10));// on sépare chaque chiffre du nombre en le
                                   // convertissant en base 10 puis on les multiplie
    c:=c+1; //on ajoute une étape car le produit est fait
  ftantque;
  retourne(c);//on affiche le nombre d'étapes
};;
```

Nous avons utilisé la commande « convert » pour transformer le nombre en une liste de chiffres afin de pouvoir les multiplier entre eux grâce à la commande « product ».
Ensuite, nous avons utilisé cet algorithme dans une boucle pour calculer les persistances de tous les nombres de 10 jusqu'à $10^n - 1$ et garder le meilleur résultat obtenu.

```
The_best(n):={
  local a,c,y;
    c:=0;
       pour y de 10 jusque (10^n)-1 faire
         si Persistance(y)>c alors c:=Persistance(y)
         a:=y;
         fsi;
       fpour;
    afficher(a);
  retourne(c);
};
```

Grâce à cet algorithme, nous obtenons les résultats suivants :

Nombre de chiffres	Meilleure persistance
2	4 (pour 77)
3	5 (pour 679)
4	6 (pour 6788)
5	7 (pour 68889)

3.2. Quelques simplifications :

Cependant, la capacité de notre somptueux et talentueux ordinateur était limitée. Nous avons donc dû alléger notre algorithme en usant de petites astuces. Cela nous permettait de ne pas prendre en compte les nombres inintéressants et ainsi de pouvoir tester plus de nombres intéressants.
Nos petites astuces consistaient donc à éliminer tous les nombres inintéressants, c'est-à-dire presque tous les nombres avec une persistance inférieure à 2 ou les nombres « décomposables » (*id est* ceux où on applique les mêmes étapes lors du procédé multiplicatif).
Voici les simplifications possibles :

- Si un nombre possède un 0, la persistance de ce nombre est de 1.
- La persistance d'un nombre possédant un ou plusieurs 1 est la même que ce même nombre sans le ou les 1.
- Si un nombre possède au moins un 2 et un 5, sa persistance sera inférieure ou égale à 2, puisqu'au bout d'une étape, le nombre obtenu comportera au moins un 0.
- Les 4 et les 8 sont obtenus avec des 2 car $4=2\times2$ et $8=2\times2\times2$.
- Les 6 sont obtenus avec des 2 et des 3 car $6=2\times3$.
- Les 9 sont obtenus avec des 3 car $9=3\times3$.

Finalement, on en déduit que tous les nombres intéressants peuvent être remplacés par un autre nombre ne comportant que des 2, 3, 5 ou 7 sans que le 2 et le 5 ne soient présents en même temps.
De plus, différents nombres comportant les mêmes chiffres mais dans un ordre différent ont la même persistance multiplicative. Par exemple, 224, 242 et 422 vont tous les trois donner 16 à l'étape suivante.
Nous avons utilisé ces astuces dans un nouvel algorithme pour pouvoir tester des nombres encore plus grands.

```
Meilleur():={
  local w,x,f,z,c,y,n,m;
  n:=0;//"n" est le nombre d'étapes du nombre testé
  c:=0;//"c" est le nombre d'étapes du nombre testé avec une persistance de "n" (pour garder un
       // nombre avec une persistance plus grande
  pour w de 0 jusque 20 faire //"w" est le nombre de 2
    pour x de 0 jusque 20 faire // "x" est le nombre de 3
      pour f de 0 jusque 20 faire //"f" est le nombre de 5
        pour z de 0 jusque 20 faire //"z" est le nombre de 7
          si w*f=0 alors //on veut que w=0 ou f=0 donc w*f=0
            y:=(2^w)*(3^x)*(5^f)*(7^z);//on effectue la première étape pour donne un nombre à
                                       // l'algorithme de persistance
            c:=Persistance(y);
            si n<c ou n=c alors//on cherche à connaître les nombres avec une plus grande persistance
              n:=c;//"n" prend toujours la valeur de la persistance calculée le plus récemment
              m:=[w,x,f,z];
              afficher(m);//l'algorithme nous renvoie une liste avec le nombre de 2, 3, 5 et 7
            fsi;
          fsi;
        fpour;
      fpour;
    fpour;
  fpour;
  n:=n+1;//on doit rajouter une étape qui correspond au changement de valeur de "y"
  retourne(n);//on obtient la persistance la plus grande
};
```

Cet algorithme nous donne les résultats suivants :

Affichage	Nombre associé	Persistance
[0,0,0,0]	1	0
[0,0,0,1]	7	0
[0,0,0,2]	77	4
[0,0,0,3]	777	4
[0,0,0,6]	777 777	4
[0,0,0,11]	77 777 777 777	4
[0,0,1,3]	5 777	4
[0,0,2,1]	557	4
[0,1,0,2]	377	4
[0,1,2,0]	355	4
[0,1,5,0]	355 555	4
[0,2,0,7]	337 777 777	4
[0,2,2,1]	33 557	5
[0,2,2,3]	3 355 777	5
[0,3,0,4]	3 337 777	7
[1,2,0,12]	233 777 777 777 777	9
[4,20,0,5]	22 223 333 333 333 333 333 333 377 777	11
[19,4,0,6]	22 222 222 222 222 222 223 333 777 777	11

La plus grande persistance que nous ayons trouvée est de 11 ! Nous avons ensuite appris que c'était la persistance la plus grande trouvée à ce jour avec les ordinateurs les plus puissants.
Les deux nombres que nous avons obtenus se simplifient (grâce aux astuces) en 2777778999999999 et 277777788888899. Ce dernier nombre est en fait le plus petit qui a une telle persistance.

4. D'autres bases :

À la suite de ces travaux, nous nous sommes intéressés au même procédé mais dans d'autres bases. Notre base de numération est la base 10 car nous avons 10 doigts, mais les babyloniens par exemple comptaient en base 60. Aujourd'hui encore on peut observer des traces de cette base dans le système temporel (60 secondes ou minutes).

On s'est dans un premier temps intéressé à la base 2 couramment appelée langage binaire. Nous nous sommes vite aperçus que dans cette base la persistance ne pouvait pas dépasser 1 car le résultat que nous pouvions obtenir lors de la première étape est toujours un nombre à un chiffre (1 ou 0) donc le procédé s'arrête dès la première étape.

Puis nous sommes passés à l'étude de la base 3. Nous avons élaboré un nouvel algorithme nous permettant d'entrer un nombre et d'avoir sa persistance.

```
Persistance_base3(y):={
  local c;
  c:=0;
  y:=convert(convert(y,base,10),base,3);
  tantque y>2 faire
         y:=product(convert(y,base,3));
         c:=c+1;
    ftantque
   retourne(c);
};;
```

Les seuls nombres intéressants sont ceux qui sont uniquement composés de 2. La persistance la plus grande trouvée grâce à notre algorithme est de 3 pour les nombres 222 et 222 222 222 222 222. Ce sont aussi les meilleurs résultats connus actuellement.

On pourrait aussi s'intéresser à la persistance multiplicative dans d'autres bases ou tenter de trouver dans la base 10 une persistance plus grande que 11… peut-être gagnerions-nous un gros chèque à la clé !!!!!

Note de l'édition

Les auteurs ne le disant pas explicitement dans l'article, insistons sur le fait surprenant que l'on ne sait toujours pas démontrer qu'il existe (ou non) un entier de plus grande persistance. Ainsi, le nombre trouvé par les auteurs correspond vraiment à l'état de l'art, et est potentiellement le meilleur possible !