Entre abaque, boulier et ordinateur

Dans les années cinquante, la disparition des dix boules de la tige des unités du boulier, opérée par notre instituteur en blouse grise, m'avait paru magique. J'essaye dans cet article, qui s'adresse au niveau secondaire, de faire revivre cela au travers de l'élaboration d'un programme en relation étroite avec un boulier, qui construit les nombres premiers et permet un codage des entiers. Certaines questions qui ont animé et animent encore la communauté mathématique y apparaissent sous des formes différentes. Ce texte illustre d'une certaine façon les propos de D. Tournès[1] : « *...l'utilisation des artéfacts anciens, éventuellement instrumentalisés d'une nouvelle manière en interaction avec les ressources modernes disponibles...* ».

Sommaire

Le programme initial

Le programme suivant fournit les nombres premiers inférieurs ou égaux à n. Il ne fait pas appel aux notions de cribles, de diviseurs ou de multiples. Il est simplement manipulatoire.
Pour l'interpréter, nous utiliserons un abaque composé d'une seule tige verticale et pouvant recevoir un grand nombre de jetons, et un boulier dont les tiges verticales ne peuvent recevoir qu'un nombre limité de boules.
Le programme est écrit à l'aide du logiciel Xcas.

1Tournès D., Perspectives historiques sur les abaques et bouliers, *MathémaTICE*, n° 51, septembre 2016.

```
PremiersInfégaux(n):={
        local a,c,j,x,u,v,L,K;
a:=2; K:=[0];  L:=[2]; //état initial//
c:=1; //on initialise le compteur//

while(c<n-1){

a:=a+1; K:=apply(x->x+1,K);
for(j:=0;j<size(K);j++){u:=K[j]; v:=L[j]; if(u==v){K:=subsop(K,j=0)}};
if(product(K)!=0){ K:=concat(K,0) ; L:=concat(L,a);}

c:=c+1;}
L;}
:;
```

Interprétation

Dans ce programme :

- La variable a compte le nombre de jetons sur la tige de l'abaque et la variable c est un compteur.
- La liste K représente virtuellement le boulier dont le nombre de tiges est égal au nombre d'éléments de K. K[0] est le nombre de boules sur la première tige, K[1] celui sur la seconde tige, etc. Si pour un indice j, K[j] = 0, cela signifie que la tige d'indice j est vide.
- La liste L contient autant d'éléments que la liste K, elle est la mémoire du boulier. De même que 10 boules vident la tige des unités d'un abaque décimal, un nombre L[j] de boules vide la tige d'indice j du boulier.
- La position initiale

```
a:=2; K:=[0]; L:=[2];
```

Il y a deux jetons sur la tige de l'abaque ; le boulier est formé d'une seule tige vide ; le nombre deux est entré en mémoire (ce choix de la position initiale sera expliqué plus loin).

– Dans la boucle introduite par `while(c<n-1){`

À chaque étape :

On ajoute un jeton sur la tige de l'abaque et une boule sur chaque tige du boulier.

```
a:=a+1; K:=apply(x->x+1,K);
```

On vide les tiges du boulier qui doivent l'être. Pour cela on compare le nombre de boules sur une tige K[j] à sa mémoire L[j].

```
for(j:=0;j<size(K);j++){u:=K[j]; v:=L[j]; if(u==v){K:=subsop(K,j=0)}};
```

Enfin, si aucune tige n'est vide, on ajoute une tige au boulier, copie conforme de celle de l'abaque (à cette étape), puis on la vide. On termine en rentrant le nombre a en mémoire.

```
if(product(K)!=0){ K:=concat(K,0) ; L:=concat(L,a);}
```

On passe à l'étape suivante.

```
c:=c+1;}
```

– On demande au programme de renvoyer la mémoire L ;

Remarque : Les nombres entrés en mémoire sont croissants et supérieurs ou égaux à 2.

Retour sur la position initiale

En réalité, on commence avec un seul jeton sur l'abaque et aucune tige sur le boulier. La première étape consiste donc à ajouter un jeton à la tige de l'abaque et, comme le boulier ne contient pas de tiges vides, on ajoute une tige au boulier, copie conforme de la tige de l'abaque et on la vide. Le nombre 2 est rentré en mémoire. On obtient la position initiale choisie pour démarrer le programme, ceci afin de pouvoir appliquer la boucle for sur la liste K qui n'est plus vide.

Par ailleurs, partir avec aucun jeton sur la tige de l'abaque ne donne rien d'autre qu'un état stationnaire pour les listes K et L comme on peut le vérifier facilement.

Application du programme

PremiersInfégaux(61)

[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61]

Je laisse au lecteur le soin d'établir la correspondance entre la liste L et la liste des nombres premiers (au sens usuel).

Codage

Mis à part les nombres 0 et 1, l'algorithme utilisé permet, par l'intermédiaire de la liste K, un codage des nombres entiers.
La liste qui suit s'obtient facilement en modifiant légèrement le programme initial.

CodeNombreInfégaux(11) (voir en Annexe)

2,[0]
3,[1,0]
4,[0,1]
5,[1,2,0]
6,[0,0,1]
7,[1,1,2,0]
8,[0,2,3,1]
9,[1,0,4,2]
10,[0,1,0,3]
11,[1,2,1,4,0]

Le code du nombre 9 est la liste [1,0,4,2]. La longueur du code est 4, elle correspond au nombre de tiges sur le boulier.

Remarques

- Il résulte de notre algorithme que pour tout nombre (> 1) le boulier possède toujours au moins une tige vide ; on en déduit que tout code contient au moins un zéro.
- Pour tout i, la composante d'indice i du code K[i] est inférieure à L[i] (mémoire de la tige).
- Soit p un nombre premier et q le nombre premier qui le suit immédiatement. La longueur du code du nombre q est supérieure de 1 à celle de p. Les codes de tous les nombres de p à q – 1 ont la même longueur.

Division euclidienne

Si on considère le code d'un nombre a

[K[0], K[1], …, K[j], …, K[m]]

la tige d'indice j du boulier porte K[j] boules (K[j] < L[j]) et s'est vidée au moins une fois (lors de sa création), notons q_j le nombre exact de fois où elle s'est vidée, on a la relation

$a = q_j . L[j] + K[j]$ avec $0 \leq K[j] < L[j]$

qui correspond à la division euclidienne de a par L[j].

Par exemple pour 11, le code est [1,2,1,4,0] ; on a 11 = 5.2 + 1 ; 11 = 3.3 + 2 ; 11 = 2.5 + 1, 11 = 1.7 + 4 ; 11 = 1.11 + 0.

Un tel codage est cependant très éloigné des codes de numération qui n'utilisent qu'un nombre restreint de symboles. De plus, on imagine mal l'utiliser pour effectuer des opérations. Il permet pourtant de visualiser sous un éclairage nouveau certaines grandes questions qui ont agité le monde mathématique pendant des siècles.

Cinq questions

Voici cinq questions que nous allons développer :

- 1) Un code peut-il être très long ?
- 2) Comment évolue le rapport entre un nombre et la longueur de son code ?
- 3) Le code d'un nombre n peut-il commencer par [x_0, x_1, x_2, …] ?
- 4) Un nombre dont le code se termine par …0] est-il premier ?
- 5) Existe-t-il un nombre infini de codes se terminant par …2, 0] ?

Question 1 : Un code peut-il être très long ?

De la remarque qui affirme que le code de tout nombre contient au moins un 0, on déduit immédiatement que pour tout nombre n, on peut trouver un nombre dont le code est plus long que celui de n.
Soit [K[0], K[1], …, K[r]] le code du nombre n, il suffit de considérer le nombre
a = L[0]*L[1]*…*L[r] + 1. L'écriture de a peut s'interpréter comme la division euclidienne de a par une mémoire du boulier et cela de différentes façons.
Donc pour ce nombre a, les r + 1 premières tiges du boulier portent chacune une seule boule, il faut donc un nombre plus grand de tiges pour que, parmi elles, l'une soit vide.
Ainsi le nombre d'éléments de la liste K n'est pas limité, celui de la liste L non plus (K et L ont le même nombre d'éléments).
L'ensemble des nombres premiers n'est pas fini. C'est le résultat d'**Euclide :**

L'ensemble des nombres premiers est infini.

Question 2 *:* Comment évolue le rapport entre un nombre et la longueur de son code ?

La longueur du code d'un nombre n correspond au nombre de nombres premiers inférieurs ou égaux à n, qui est conventionnellement noté $\pi(n)$. Legendre et Gauss ont émis des conjectures concernant le rapport $\frac{\pi(n)}{n}$. À la fin du 19e siècle, il a été prouvé que la limite quand n tend vers

l'infini du rapport $\frac{\pi(n).\ln(n)}{n}$ est 1.
C'est le théorème des nombres premiers d'**Hadamard** et **de La Vallée Poussin.**

Question 3 : Le code d'un nombre n peut-il commencer par [x_0, x_1, x_2, …] ?

Pour cela il est nécessaire que, pour tout i, $x_i < L[i]$ (compatibilité avec la mémoire de la tige).

Le théorème des restes chinois[2] affirme l'existence d'une solution pour les congruences

$n \equiv x_0 \pmod 2$
$n \equiv x_1 \pmod 3$
$n \equiv x_2 \pmod 5$

L'ensemble des solutions correspond aux termes d'une suite arithmétique. Par exemple pour un code commençant par [1, 1, 2, …], la suite est $u_n = 30.n + 7$.
Vers le milieu du 19e siècle a été établi le théorème de la progression arithmétique de **Dirichlet** :

> *Une suite arithmétique $u_n = a.n + b$ avec a et b premiers entre eux contient une infinité de nombres premiers.*

Il existe donc une infinité de nombres premiers dont le code est de la forme [1, 1, 2, …, 0] puisque 30 et 7 sont premiers entre eux.
Ce sont les nombres 7 ; 37 ; 67 ; 97 ; 127 ; 157 ; 277 ; etc.

Voici les codes des premiers nombres premiers .
CodepremiersInfégaux(23) (voir en Annexe 2)

2,[0]
3,[1,0]
5,[1,2,0]
7,[1,1,2,0]
11,[1,2,1,4,0]
13,[1,1,3,6,2,0]
17,[1,2,2,3,6,4,0]
19,[1,1,4,5,8,6,2,0]
23,[1,2,3,2,1,10,6,4,0]

Le code d'un nombre premier se termine par 0.
C'est une conséquence de l'algorithme.

Question 4 : Un nombre dont le code se termine par …0] est-il premier ?

Derrière cette question apparemment anodine se cache le postulat de **Bertrand.**

Pour n > 1, il existe un nombre premier (> n) entre n et 2n.

Soit p un nombre premier, pour ce nombre la dernière tige du boulier vient d'être créée et vidée et le postulat affirme qu'avant qu'elle ne se vide à nouveau, la longueur du code aura changé. Donc les nombres compris strictement entre deux nombres premiers consécutifs ont des codes qui ne se terminent pas par 0.

Si le code se termine par zéro, le nombre est premier.

Question 5 : Existe-t-il un nombre infini de codes se terminant par …2, 0] ?

2 Voir à ce sujet la page de Wikipedia.

Soit p un nombre dont le code est [1, …, 2, 0] ; d'après ce qui précède p est premier et le 0 provient de la création d'une tige ; la première composante du code ne peut ainsi pas être 0, c'est donc 1 (p est impair).
Les codes des nombres p – 1 et p – 2 sont obtenus à partir de celui de p (en remontant).

p – 2, [1, …, 0]
p – 1, [0, …, 1]
p, [1, …, 2, 0]

On en déduit que le nombre p – 2 dont le code se termine par 0 est également premier. Ainsi p – 2 et p forment un couple de nombres premiers séparés par deux unités. On les appelle nombres premiers jumeaux. La question posée revient donc à savoir s'il existe une infinité de nombres premiers jumeaux. C'est la conjecture des nombres premiers jumeaux :

Il existe une infinité de nombres premiers jumeaux.

À ce jour elle n'est pas élucidée bien que de nombreux mathématiciens dont **Yitang Zhang** et **Tao** approchent d'assez près la solution.

Un code binaire ?

Peut-on écrire le code en utilisant seulement deux symboles, 0 et 1, selon la parité du nombre de boules sur chaque tige ?
Par exemple, pour le nombre 5, le code [1,2,0] se transforme en [1,0,0] qui est son code binaire et pour 11, [1,2,1,4,0] donne le code binaire [1,0,1,0,0].
L'information perdue était elle essentielle ? Posons la question autrement :

Deux nombres différents ont-ils des codes binaires différents ?

Deux remarques préliminaires

- Si les deux nombres n'ont pas la même parité, alors les codes binaires diffèrent à partir de la première composante du code.
- Si les codes binaires ont des longueurs différentes, ils sont clairement différents.

Pour répondre à la question, il nous reste donc à considérer deux nombres de même parité et dont les codes ont même longueur, c'est-à-dire compris entre deux nombres premiers consécutifs.
Prenons un exemple : considérons les nombres 23 et 27.

i / n	0 m=2	1 m=3	2 m=5	3 m=7	4 m=11	5 m=13	6 m=17	7 m=19	8 m=23
23	1	0	1	0	1	0	0	0	0
24	0	0	0	1	0	1	1	1	1
25	1	1	0	0	1	0	0	0	0
26	0	0	1	1	0	0	1	1	1
27	1	0	0	0	1	1	0	0	0
28	0	1	1	0	0	0	1	1	1

Nous observons que les codes binaires de 23 et 27 diffèrent par leurs composantes d'indice 2. La présence des deux zéros consécutifs en rouge révèle que lors du passage de 23 à 27 par addition successive d'une boule sur la tige d'indice 2, celle-ci s'est vidée exactement une fois. Le premier zéro traduit le fait qu'il y a quatre boules sur la tige, l'ajout d'une boule vide la tige dont la mémoire est 5, on obtient le deuxième zéro. Cette irrégularité rompt l'alternance des zéros et des uns, et provoque la différence des codes binaires de 23 et de 27.

Montrons que deux nombres de même parité et ayant des codes de même longueur ont des codes binaires différents. Pour cela nous utiliserons le théorème de **Sylvester** :

> *Pour tout nombre n supérieur ou égal à k, le nombre (n + 1)(n + 2)(n + 3)…(n + k) est divisible par un nombre premier strictement supérieur à k.*

Soit deux nombres premiers consécutifs q et q' et deux nombres n_0 et $n_0 + k$ de même parité tels que
$q \leq n_0 \leq n_0 + 1 \leq \ldots \leq n_0 + k < q'$.
Le nombre k vérifie $2 \leq k$, car n_0 et $n_0 + k$ sont de même parité ; il vérifie aussi $k < q$ (postulat de Bertrand) donc $k \leq n_0$.
Pour les nombres $n_0 + 1$, $n_0 + 2$, …, $n_0 + k$, le théorème de Sylvester fournit un nombre premier p ($p > k$) qui divise l'un d'eux.

Comme aucun des nombres $n_0 + 1$, $n_0 + 2$, …, $n_0 + k$ n'est premier, le nombre p est la mémoire d'une des tiges du boulier, appelons j son indice, celle-ci se vide exactement une fois entre n_0 et $n_0 + k$, ce qui entraine que les composantes des codes binaires d'indice j des deux nombres sont différentes. Les deux nombres n_0 et $n_0 + k$ ont des codes différents.

Nous avons établi la proposition :

> Deux nombres différents ont des codes binaires différents.

Terminons avec l'image des longues queues de zéros à la fin des codes binaires des nombres premiers.

2,[0]
3,[1,0]
5,[1,0,0]
7,[1,1,0,0]
11,[1,0,1,0,0]
13,[1,1,1,0,0,0]
17,[1,0,0,1,0,0,0]
19,[1,1,0,1,0,0,0,0]
23,[1,0,1,0,1,0,0,0,0]
29,[1,0,0,1,1,1,0,0,0,0]
31,[1,1,1,1,1,1,0,0,0,0,0]
37,[1,1,0,0,0,0,0,0,0,0,0,0]
41,[1,0,1,0,0,0,1,1,0,0,0,0,0]
43,[1,1,1,1,0,0,1,1,0,0,0,0,0,0]
47,[1,0,0,1,1,0,1,1,1,0,0,0,0,0,0]
53,[1,0,1,0,1,1,0,1,1,0,0,0,0,0,0,0]
59,[1,0,0,1,0,1,0,0,1,1,0,0,0,0,0,0,0]
61,[1,1,1,1,0,1,0,0,1,1,0,0,0,0,0,0,0,0]

Le signal observé en rouge amplifie le postulat de Bertrand.

Dans le code binaire de 43, les nombres en rouge 0,0,0,0,0,0 correspondent aux tiges du boulier dont les mémoires sont respectivement 23, 29, 31, 37, 41, 43, précisément les nombres premiers entre 23 et son double 46.

Pour bien comprendre pourquoi un tel code relève plus de la cryptographie que du reste, considérons le code binaire

[1,1,0,0,0,1,1,1,0,1,0,0,1,1,0,1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

correspondant à un nombre n.

Trouver le code binaire de n + 1 n'est pas évident !

Annexe

Programmes

La différence entre le programme initial et les deux programmes qui suivent tient seulement à la position de la consigne print(a,K).

```
CodeNombresInfégaux(n):={
local a,c,j,x,u,v,L,K;
a:=2; K:=[0]; L:=[2]; //état initial//
print(a,K) ;
c:=1; //on initialise le compteur//
while(c<n-1){
a:=a+1; K:=apply(x->x+1,K);
for(j:=0;j<size(K);j++){u:=K[j]; v:=L[j]; if(u==v){K:=subsop(K,j=0)}};
if(product(K)!=0){ K:=concat(K,0) ; L:=concat(L,a);}
print(a,K) ;
c:=c+1;}
}
:;
```

```
CodePremiersInfégaux(n):={
local a,c,j,x,u,v,L,K;
a:=2; K:=[0]; L:=[2]; //état initial//
print(a,K) ;
c:=1; //on initialise le compteur//
while(c<n-1){
a:=a+1; K:=apply(x->x+1,K);
for(j:=0;j<size(K);j++){u:=K[j]; v:=L[j]; if(u==v){K:=subsop(K,j=0)}};
if(product(K)!=0){ K:=concat(K,0) ; L:=concat(L,a); print(a,K) ;}

c:=c+1;}
}
:;
```

```
CodeBinaire(n):={
local a,c,j,x,y,u,v,l,m,T,L,K,W;
a:=2; K:=[0]; L:=[2]; //état initial//
c:=1; //on initialise le compteur//
T:=K ;
if(n==2){ return 2,T;}
while(c<n-1){
a:=a+1; K:=apply(x->x+1,K);
for(j:=0;j<size(K);j++){u:=K[j]; v:=L[j]; if(u==v){K:=subsop(K,j=0)}};
if(product(K)!=0){ K:=concat(K,0) ; L:=concat(L,a); }
T:=T,K ;
c:=c+1;}
W:=T[size(T)-1];
for(l:=0;l<size(W);l++){if( irem(W[l],2)==0){W:=subsop(W,l=0);}};
for(m:=0;m<size(W);m++){if( irem(W[m],2)==1){W:=subsop(W,m=1);}};
return a,W;
}
:;
```

```
TestUnicitéCode(p):={ //p est un nombre premier > 2//
local j,T;
T:=%{%};
for(j:=p;j<nextprime(p);j++){T:=T union %{codebinaire(j)%};};
if(nextprime(p)-p != size(T)){return p;};
return 1;
}
:;
```