D'Euclide à Fibonacci

par P. GAGNAIRE

Problème : On recherche le PGCD des deux naturels a et b par l'algorithme d'Euclide.
Combien faut-il effectuer, au plus, de divisions ?

1 Recherche d'une solution

L'algorithme d'Euclide s'écrit ainsi :

$$\begin{array}{ll} a = b\,q_1 + r_1 & 0 \leqslant r_1 < b \\ b = r_1\,q_2 + r_2 & 0 \leqslant r_2 < r_1 < b \\ \dots\dots\dots\dots & \dots\dots\dots\dots \\ r_{n-2} = r_{n-1}\,q_n + r_n & 0 \leqslant r_n < \dots < r_1 < b \end{array}$$

Si $r_n = 0$, la nième division est la dernière, et le pgcd de a et b est r_{n-1}, reste de la division précédente.

r_{n-1} étant le pgcd de a et b, il divise le reste r_1 de leur division euclidienne, donc aussi r_2, ..., r_{n-2}. Il en résulte qu'en remplaçant a et b par leurs quotients respectifs a' et b' par r_{n-1}, on obtient les n divisions euclidiennes suivantes :

$$\begin{array}{ll} a' = b'q_1 + r'_1 & 0 \leqslant r'_1 < b' \\ b' = r'_1\,q_2 + r'_2 & 0 \leqslant r'_2 < r'_1 < b \\ \dots\dots\dots\dots & \dots\dots\dots\dots \\ r'_{n-2} = r'_{n-1}\,q_n + r'_n & 0 \leqslant r'_n < \dots < r'_1 < b \end{array}$$

dans lesquelles r'_1, r'_2, ..., r'_n sont les quotients respectifs de r_1, r_2, ..., r_n par r_{n-1}.

De cette remarque, il résulte que l'on peut toujours se ramener, pour traiter le problème posé, au cas où a et b sont étrangers (premiers entre eux).

Au besoin, on les remplace par leurs quotients par leur pgcd, c'est-à-dire par des naturels plus petits et cela, sans modifier le nombre de divisions de l'algorithme d'Euclide.

On a donc

$$r_{n-1} = 1$$

$$0 < 1 < r_{n-2} < r_{n-3} < \dots < r_1 < b$$

A priori, il est évident qu'il ne peut exister que b naturels *au plus* dans l'ensemble $\{0, 1, r_{n-2}, \dots, r_1\}$.

Ces b naturels seraient obtenus pour :

$$r_{n-1} = 1 \; ; r_{n-2} = 2 \; ; \dots \; ; r_{n-k} = k \; ; \dots \; ; r_1 = b - 1$$

Or voyons si cela est possible, en utilisant les divisions de l'algorithme d'Euclide, à partir de la dernière.

Remarquons que les quotients q_1, $q_2, \dots, q_n$ sont tous au moins égaux à 1 (si $a > b$, ce que l'on peut toujours supposer), puisque la suite :

$$a, b, r_1, \dots, r_{n-}\ , r_{n-1}$$

est strictement décroissante.

On a donc, pour chaque division euclidienne :

$$r_{k-2} > r_{k-1} + r_k$$

Il en résulte :

$$\begin{aligned} r_{n-1} &= 1 \\ r_{n-2} &\geqslant 2 \\ r_{n-3} &\geqslant 2 + 1 = 3 \\ r_{n-4} &\geqslant 3 + 2 = 5 \\ r_{n-5} &\geqslant 5 + 3 = 8 \end{aligned}$$

On voit alors apparaître les termes de la suite de Fibonacci :

$$1 , 2 , 3 , 5 , 8 , 13 \dots$$

pour lesquels on a la propriété, à partir du troisième :

$$u_p = u_{p-1} + u_{p-2}$$

Cela suggère de prendre le problème à l'envers en posant la question :

Sachant que la recherche du PGCD de a et b (a > b) par l'algorithme d'Euclide nécessite n divisions, quelle est la plus petite valeur possible pour b ?

Comme b est le dernier terme de la suite croissante

$$0 , 1 , r_{n-2} , r_{n-3} , \dots , r_1 , b$$

il sera le plus petit possible dans le seul cas où chaque terme de cette suite sera le plus petit possible, c'est-à-dire dans le cas où (à partir de r_{n-3})

$$r_{k-2} = r_{k-1} + r_k$$

Cela entraîne que les naturels

$$r_{n-1} , r_{n-2} , \dots , r_{n-p} , \dots , r_1 , b$$

sont respectivement égaux aux éléments de la suite de Fibonacci :

$$u_1 , u_2 , \dots , u_p , \dots , u_{n-1} , u_n$$

avec $u_1 = 1$ $u_2 = 2$

et $u_p = u_{p-1} + u_{p-2}$

Conclusion

Si la recherche du PGCD de a et b par l'algorithme d'Euclide nécessite n divisions, alors b est au moins égal au nième terme de la suite de Fibonacci de premiers termes

$$u_1 = 1 \quad \text{et} \quad u_2 = 2 .$$

a et b étant donnés, pour trouver combien de divisions nécessitera *au plus* l'algorithme d'Euclide, on peut donc encadrer b par deux termes consécutifs de la suite de Fibonacci. Le rang du plus petit d'entre eux donne la solution.

2 Intérêt pratique de ce problème

1°) La suite de Fibonacci est *pratiquement* exponentielle : le quotient $\frac{u_n}{u_{n-1}}$ diffère très peu, à partir d'un certain rang, du *nombre d'or :*

$$\frac{1+\sqrt{5}}{2} \approx 1{,}618$$

Or il existe une suite exponentielle usuelle, c'est la suite des puissances de dix :

$$1 , 10 , 100 , 1\,000 , 10\,000 \dots$$

Il est donc intéressant de comparer la suite de Fibonacci et la suite des puissances de dix.

Pour cela, écrivons les premiers termes de la suite de Fibonacci en changeant de ligne chaque fois que l'on dépasse une puissance de 10, c'est-à-dire chaque fois que l'écriture décimale nécessite un chiffre de plus que précédemment :

1	2	3	5	8
13	21	34	55	89
144	233	377	610	987
1597	2584	4181	6765	
10946	17711	28657	46368	75025
121393	....	....	....	....

On s'aperçoit que chaque ligne contient au plus cinq éléments de la suite. Cela peut se justifier par le fait que dix est compris entre

$$\left(\frac{1+\sqrt{5}}{2}\right)^4 \quad \text{et} \quad \left(\frac{1+\sqrt{5}}{2}\right)^5$$

Il en résulte une règle d'application très simple :

Le nombre de divisions nécessité par l'algorithme d'Euclide pour deux naturels a et b est au plus égal au produit par 5 du nombre de chiffres du plus petit des naturels a et b.

Par exemple, pour trouver un majorant du nombre de divisions nécessité par l'algorithme d'Euclide appliqué à 1597 et 987, il suffit de multiplier 3 (987 s'écrit avec 3 chiffres) par 5.

Effectivement, on contrôle que, dans ce cas (où les naturels sont deux termes consécutifs de la suite de Fibonacci) l'algorithme d'Euclide comporte 15 divisions.

Notons qu'à partir de 10946, la règle pratique donnée ici est trop "*pessimiste*" d'une division puisque l'intervalle [1000 ; 10000] ne contient que quatre termes de la suite de Fibonacci. Cette règle deviendrait encore plus "pessimiste" pour des naturels plus grands mais ce n'est pas grave puisque le problème posé est un problème de majoration :

la règle fait espérer un résultat assez bon ;
le calcul effectif donne un résultat meilleur encore.

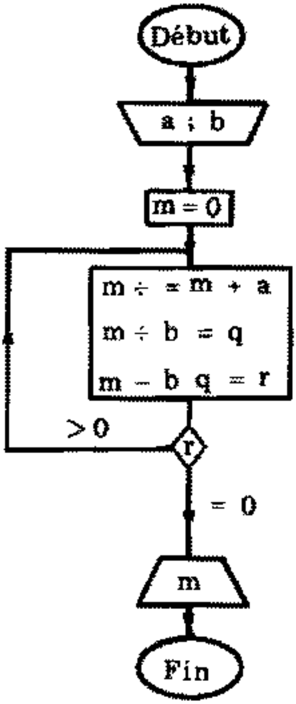
2°) Les élèves de certaines classes travaillent actuellement sur de petites machines à calculer programmables (mini-ordinateurs).

Parmi les activités qu'on peut leur proposer, il y a la programmation du calcul du PGCD et celle du PPCM de deux naturels.

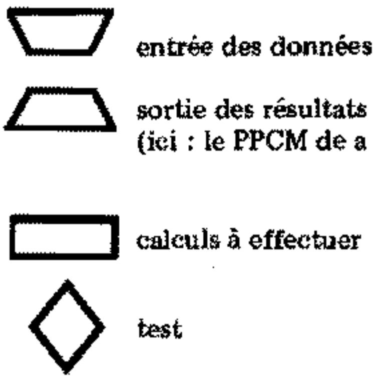
Le programme du calcul du PPCM est très simple à établir :

on demandera à la machine d'écrire la suite des multiples de a jusqu'à ce qu'elle trouve un multiple de b. Celui-ci sera alors le PPCM de a et b.

Voici l'ordinogramme correspondant :



Légende

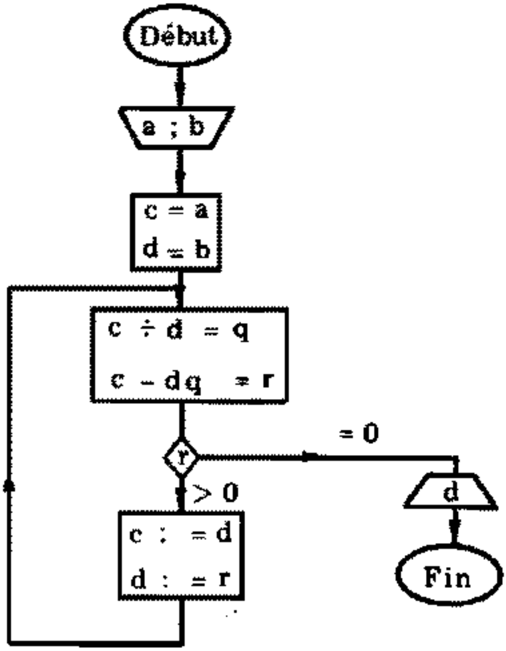


m ÷ b désigne le *quotient euclidien* de a par b

m : = m+a signifie : à la place de m, écrire m + a

Le programme du calcul du PGCD, quoique pas tellement plus compliqué, nécessite la connaissance de l'algorithme d'Euclide.

Voici l'ordinogramme correspondant :

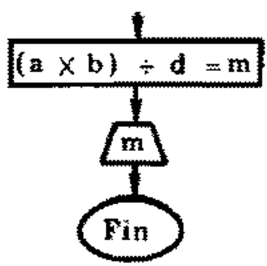


Le deuxième programme peut être utilisé aussi pour calculer le PPCM de a et b. En effet, il est bien connu qu'entre a, b, leur PGCD, d, et leur PPCM, m, il existe la relation :

$$a \times b = m \times d$$

Il suffit donc de changer, dans le second ordinogramme

par



Il est bien évident que le dernier ordinogramme (calcul du PPCM par l'intermédiaire du PGCD) donnera un programme plus long que le premier (calcul du PPCM par les multiples successifs de a).

Mais, est-il sûr que le temps de calcul (par l'ordinateur) du PPCM sera plus long par la dernière méthode que par la première ? Autrement dit : la minimisation du temps de recherche du PPCM correspond-elle à la minimisation de la longueur du programme ?

Il est facile de voir qu'il n'en est rien, dans le cas présent.

En effet, le calcul du PPCM de 1597 et 987 par la première méthode nécessite 987 boucles du programme, puisque 987 est premier avec 1597, alors que par la dernière méthode, il suffit de 15 boucles du programme suivies d'une multiplication et d'une division : le calcul de plus courte durée est obtenu (et de loin !) au moyen du programme le plus long !

On voit sur cet exemple qu'il est possible de motiver par une recherche d'ordre pratique (optimisation d'un programme) un aspect beaucoup plus théorique (algorithme d'Euclide et suite de Fibonacci) de la question posée.

Bibliographie

1. *Elementary number theory*, Uspensky et Heaslet, Mc Graw Hill, 1939.

2. *The analysis of algorithms*, D.E. Knuth, Actes du Congrès international des mathématiciens, Nice, 1970.