TEST DE PRIMALITÉ
ET MÉTHODES DE FACTORISATION

Par Jean-Louis NICOLAS

Introduction.

Dans ce colloque d'algèbre "effective", le but de cet exposé est de montrer que certaines structures algébriques servent à résoudre des problèmes concrets de factorisation. Ces structures ne sont pas très compliquées : le groupe multiplicatif des éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$, le groupe des classes d'un corps quadratique, le groupe des points d'une courbe elliptique, mais en revanche il n'apparait pas du tout évident que ces structures puissent être utilses pour trouver des diviseurs d'un nombre entier.

Pourquoi des groupes ?

Rappelons d'abord le petit théorème de Fermat : si p premier ne divise pas a, alors

$$a^{p-1} \equiv 1 \bmod p.$$

Ce théorème est l'application au groupe $(\mathbb{Z}/p\mathbb{Z})^*$ du résultat classique : si G est un groupe fini à n éléments et d'élément neutre e, on a pour tout $a \in G$:

$$a^n = e .$$

On déduit immédiatement du théorème de Fermat un test garantissant qu'un nombre N est composé (c'est-à-dire non premier)

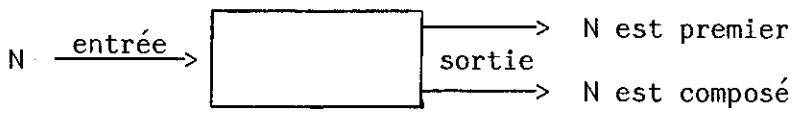
(1) $\qquad (a,N) = 1 \quad \text{et} \quad a^{N-1} \not\equiv 1 \bmod N \Longrightarrow N$ composé.

Ce test est très rapide à exécuter en ordinateur car les deux opérations : calcul de (a,N) et calcul de $a^{N-1} \bmod N$ ont un temps d'exécution proportionnel au nombre de chiffres de N. De même dans n'importe quel groupe G, le calcul de a^n est inférieur à : constante × (log n) × (temps d'éxécution d'une multiplication dans G).

Multiprécision.

Les calculettes, et les ordinateurs calculent en général sur des nombres d'une dizaine de chiffres maximum. L'intérêt des méthodes de factorisation actuelles est de travailler sur des nombres de quelques dizaines de chiffres. Il faut donc posséder d'abord un logiciel de multiprécision, c'est-à-dire de traitement de grands nombres, avant de programmer une méthode de factorisation.

Test de primalité (cf. [Dix], [Nic], [Rie]) :



C'est une machine, qui, à partir d'un entier N, indique s'il est premier ou composé. Si N est composé, la machine n'indique pas forcément un diviseur de N.

Nous avons vu le test (1) certifiant qu'un nombre est composé. Un test meilleur est

(2) $\qquad (a,N) = 1 \quad \text{et} \quad a^{(N-1)/2} \not\equiv \pm 1 \bmod N \implies N$ est composé.

En effet, si N est premier $a^{(N-1)/2}$ est une racine carrée de 1 dans le corps $\mathbb{Z}/N\mathbb{Z}$, donc ± 1.

Si N n'est pas premier, la proportion de a compris entre 1 et N et tels que $a^{(N-1)}/2 \equiv \pm 1$ est inférieure à 50 %. Si l'on a trouvé plusieurs valeurs de a au hasard telles que $a^{(N-1)/2} \equiv \pm 1 \bmod N$, il y a de grandes chances que N soit premier.

Un test récent (cf. [Adl], [Coh 2], [Coh 3]) garantit qu'un nombre de 100 à 200 chiffres est premier en quelques minutes d'ordinateur.

Méthode de factorisation (cf. [Guy], [Pom 1], [Mon])

C'est une machine qui fournit un diviseur de l'entier N, qui est garanti composé.

Les algorithmes que nous allons exposer ne sont pas démontrés. Nous donnerons seulement quelques remarques montrant que l'on peut en espérer un résultat. Mais dans une méthode de factorisation, ce qui compte est d'obtenir D, et il est facile de vérifier que c'est un diviseur de N.

Par contre la fiabilité d'un test de primalité est plus discutable : il y a peu d'indices pour s'assurer qu'il n'y a pas eu de faute dans le programme et que la sortie binaire : (N premier, N composé) n'a pas été permutée.

Nous présenterons 4 algorithmes de factorisation. Les 3 premiers utilisent une structure de groupe. Le quatrième, le crible quadratique est actuellement le plus performant. Mais d'abord, il faut parler d'une fonction essentielle dans l'évaluation de ces algorithmes, la fonction Ψ.

La fonction Ψ de De Bruijn.

Soit P(n) le plus grand facteur premier de n. On définit pour x réel plus grand que 1, et pour y réel inférieur à x :

$$\Psi(x,y) = \sum_{\substack{n \leqslant x \\ P(n) \leqslant y}} 1 .$$

Cette fonction a été étudiée par N.G. De Bruijn ([De B]), E.R. Canfield, P. Erdös et C. Pomerance ([Can]) et plus récemment par H. Maier ([Mai]), puis par A. Hildebrand et G. Tenenbaum ([Hil]).

On pose $u = (\log x)/(\log y)$, et le résultat suivant est suffisant pour étudier les méthodes de factorisation : pour $u \to +\infty$, et $y \geqslant (\log x)^{1+\varepsilon}$, on a :

$$\Psi(x,y) = x/(u^{u(1+o(1))}) . \tag{3}$$

On peut dire également que la probabilité qu'un nombre entier voisin de x ait tous ses diviseurs premiers $\leqslant y$ est environ u^{-u}.

Cette estimation de Ψ sert à évaluer plusieurs algorithmes de factorisation. Cette évaluation se fait à l'aide de la quantité L(N), où l'on définit

$$L(x) = \exp(\sqrt{(\log x)(\log\log x)})$$

et l'on déduit de (3), lorsque α et v sont des constantes :

$$\Psi(x^{\alpha},(L(x))^{v}) = x^{\alpha}(L(x))^{-\alpha/2v+o(1)} \tag{4}$$

On utilisera en fait la fonction

$$\tilde{\Psi}(x,y) = \sum_{\substack{n<x \\ n \mid y!}} 1 .$$

On a évidemment $\tilde{\Psi}(x,y) \leqslant \Psi(x,y)$, mais les deux fonctions sont assez voisines et les relations (3) et (4) sont encore valables pour $\tilde{\Psi}$.

La méthode (p-1) de Pollard (cf. [Pol]).

On veut factoriser N. Pour $j = 1,2,\ldots,k$, on calcule :

$$\text{pgcd}(a^{j!}-1,N).$$

Si, pour une valeur de j cette quantité est $\neq 1,N$, on a évidemment factorisé N.

Maintenant supposons que $p|N$ et que $p-1|k!$. Alors le théorème de Fermat montre que

$$a^{k!} \equiv 1 \mod p$$

et que : p divise $(a^{k!}-1,N)$.

Par conséquent, cette méthode permet de détecter les facteurs premiers de N, tels que les diviseurs premiers de p-1 soient petits.

La méthode de Schnorr et Lenstra (cf. [Sch]).

Comme l'a observé Gauss, l'ensemble des classes d'équivalence de formes quadratiques $AX^2+BXY+CY^2$, de discriminant fixé $B^2-4AC = \Delta<0$ est un groupe $\mathcal{H}(\Delta)$. Dans chaque classe d'équivalence il existe une forme réduite, et Gauss a donné un bon algorithme de réduction, ainsi qu'un bon algorithme pour calculer la composition de deux formes pour cette loi de groupe (cf. [Sha] et [Coh 1]. L'élément neutre est :

$$I = (1,0,\Delta/4) \quad \text{si} \quad \Delta \equiv 0 \mod 4$$

$$I = (1,1,(1-\Delta)/4) \quad \text{si} \quad \Delta \equiv 1 \mod 4$$

et l'inverse de la forme (A,B,C) est (A,-B,C).

On appelle ambige une forme H telle que $H^2 = I$. Il en existe trois espèces :

1) $B = 0$
2) $A = B$
3) $A = C$.

On observera que la connaissance d'une forme ambige fournit une factorisation de Δ.

On désigne l'ordre du groupe par $h(\Delta)$. La formule :

$$h(\Delta) = \frac{\sqrt{-\Delta}}{\pi} \prod_{p=2}^{\infty} \frac{p}{p-(\frac{\Delta}{p})}$$

où $(\frac{\Delta}{p})$ est le symbole de Kronecker montre que $h(\Delta)$ est de l'ordre de grandeur de $\sqrt{-\Delta}$.

Pour factoriser N, on pose $\Delta = -N$ si $N \equiv 3 \bmod 4$ et $\Delta = -4N$ si $N \equiv 1 \bmod 4$. On choisit ensuite une forme au hasard $F = (A,B,C) \in \mathcal{H}(\Delta)$ et pour $j = 1,2,\ldots,k$ on calcule $F^{j!}$ en espérant trouver I.

Si $h(\Delta) | k!$, alors on a $F^{k!} = I$. Il reste à trouver une forme ambige : on décrit $k! = 2^{\alpha}k'$, avec k' impair. On calcule $F^{k'} = G$. Si $G = I$, on recommence avec une autre forme F_1. Si $G \neq I$, on calcule $G^2, G^4, G^8, \ldots$ jusqu'à $G^{2^{\beta}} = I$, alors $H = G^{2^{\beta-1}}$ est ambige et fournit une factorisation de Δ, et en général de N.

Cette méthode est efficace si $h(\Delta)$ n'a que des petits facteurs premiers. On décide donc d'essayer simultanément plusieurs groupes $\mathcal{H}(\Delta_s)$ pour $1 \leqslant s \leqslant S$, s sans facteur carré et premier avec N, et $\Delta_s = -sN$ si $sN \equiv 3 \bmod 4$ et $\Delta_s = -4sN$ si $sN \equiv 1 \bmod 4$. On fait ensuite l'hypothèse que les nombres $h(\Delta_s)$ sont des nombres au hasard voisin de $\sqrt{N}$, et donc que la probabilité que $h(\Delta_s)$ divise $k!$ est environ $\frac{1}{\sqrt{N}}\Psi(\sqrt{N},k)$. Il est donc raisonnable de choisir $S = \sqrt{N}/\Psi(\sqrt{N},k)$. Le nombre d'opérations nécessaires au calcul de $F^{k!}$ est $O(\log k!) = O(k \log k)$ et le nombre total d'opérations est $O(Sk \log k)$.

Le choix de $k = L^{1/2}$, avec $L = \exp(\sqrt{\log N \log\log N})$ donne $S = L^{1/2+o(1)}$, et une estimation heuristique du temps d'exécution de l'algorithme en $L^{1+o(1)}$.

La méthode des courbes elliptiques de Lenstra (cf. [Len]).

La méthode p-1 est basée sur le groupe $(\mathbb{Z}/p\mathbb{Z})^*$, la méthode précédente

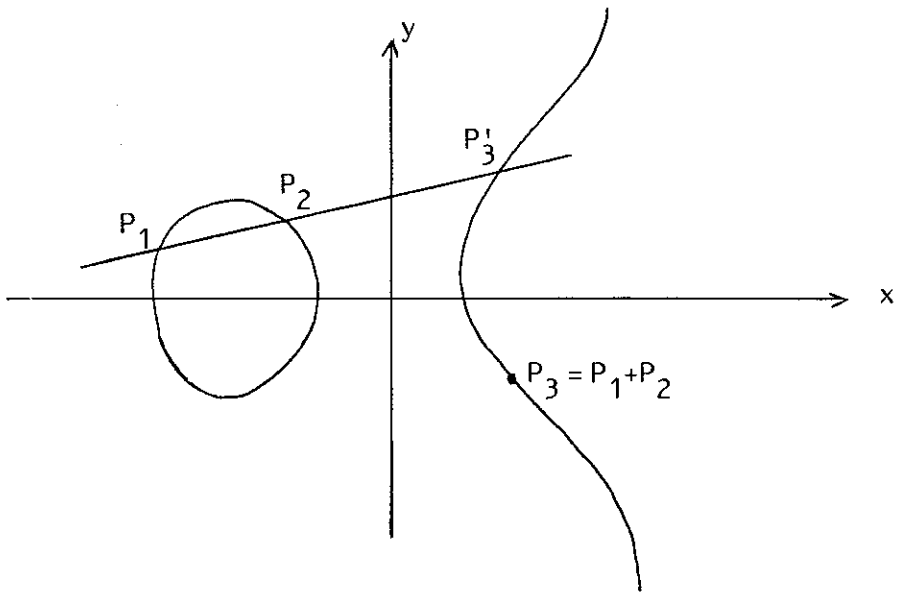
sur le groupe $\mathcal{H}(\Delta)$; il est donc assez naturel de chercher une méthode basée sur le groupe des points d'une courbe elliptique.

Rappelons d'abord quelques propriétés des courbes elliptiques que l'on pourra trouver par exemple dans [Kob] ou [Sil].

Soit la courbe elliptique (E), mise sous la forme de Weierstrass :

$$y^2 = 4x^3 - g_2x - g_3$$

que l'on suppose non décomposée, c'est-à-dire $g_2^3 + 27\,g_3^2 \neq 0$. Soient $P_1(x_1,y_1)$ $P_2(x_2,y_2)$ deux points de cette cubique. Le point $P_3 = P_1+P_2$ est obtenu géométriquement comme suit : la droite P_1P_2 recoupe la courbe en un troisième point P_3'. P_3 et P_3' sont symétriques par rapport à l'axe des x.



On observe (par exemple par le calcul, mais la preuve de l'associativité est technique) que cette addition est une loi de groupe. L'élément neutre 0 est le point à l'infini sur l'axe des y. Deux éléments opposés sont symétriques par rapport à l'axe des x. Trois points alignés ont une somme nulle.

Supposons $x_1 \neq x_2$ et soit $\lambda = \dfrac{y_2-y_1}{x_2-x_1}$ la pente de la droite P_1P_2. Si $y=\lambda x+\mu$ est l'équation de P_1P_2, x_3 sera la troisième racine de l'équation :

$$4x^3 - g_2x - g_3 = (\lambda x+\mu)^2 .$$

Les trois racines ont pour somme $-\lambda^2/4$. Donc

$$\begin{cases} x_3 = -x_1 - x_2 + \frac{1}{4}\left(\dfrac{y_2-y_1}{x_2-x_1}\right)^2 \\ y_3 = -\left(y_1 + \dfrac{y_2-y_1}{x_2-x_1}(x_3-x_1)\right) . \end{cases}$$

Supposons $x_1 = x_2$. Si $y_1 = -y_2$ alors $P_1+P_2 = 0$. Si $y_1 = y_2$, la pente de P_1P_2 est la pente de la tangente à (E),

$$\lambda = (12x_1^2 - g_2) / 2y_1$$

et l'on a :

$$\begin{cases} x_2 = -2x_2 + \lambda^2/4 \\ \dfrac{y_3'-y_1}{x_3-x_1} = \lambda \quad ; \quad y_3 = -y_3' . \end{cases}$$

Formules projectives :

(E) : $y^2t = 4x^3 - g_2xt^2 - g_3t^3$, g_2 pair, $(x_3,y_3,t_3) = (x_1,y_1,t_1) * (x_2,y_2,t_2)$.

a) $\boxed{\text{Si } x_1t_2 - x_2t_1 \neq 0,}$

on pose

$$X = (y_1t_2 - y_2t_1)^2t_1t_2 - 4(x_1t_2+x_2t_1)(x_1t_2-x_2t_1)^2$$

$$x_3 = X(x_1t_2 - x_2t_1)$$

$$y_3 = -X(y_1t_2 - y_2t_1) - 4(x_1y_2 - y_1x_2)(x_1t_2 - x_2t_1)^2t_1t_2$$

$$t_3 = 4(x_1t_2 - x_2t_1)^3t_1t_2 .$$

b) $\boxed{\text{Si } x_1 - t_2 - x_2 t_1 = 0 \text{ et } y_1 t_2 - y_2 t_1 \neq 0}$

alors $(x_3, y_3, t_3) = (0,1,0)$ = élément neutre.

c) $\boxed{\text{Si } x_1 t_2 - x_2 t_1 = y_1 t_2 - y_2 t_1 = 0}$

(donc, projectivement, $(x_1, y_1, t_1) = (x_2, y_2, t_2)$), on pose :

$$X = \left(6x_1^2 - \frac{g_2}{2} t_1^2\right)^2 - 8x_1 y_1^2 t_1$$

$$x_3 = X y_1 t_1$$

$$y_3 = -\left(6x_1^2 - \frac{g_2}{2} t_1^2\right)(X - 4x_1 y_1^2 t_1) - 4y_1^4 t_1^2$$

$$t_3 = 4y_1^3 t_1^3 .$$

Réduction modulo p

La loi de groupe précédente marche si l'on considère x et y réels, ou complexes. On peut également considérer le cas où x et $y \in \mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. On peut prévoir que pour les p valeurs de x, $4x^3 - g_2 x - g_3$ sera une fois sur deux un carré, et donc qu'il y aura environ p points sur la courbe. En fait, un théorème dû à Hasse, montre que ce nombre de points est compris entre $p+1 - 2\sqrt{p}$ et $p+1 + 2\sqrt{p}$.

Algorithme

On part d'un point $P = (x_0, y_0, 1)$ au hasard, et d'une courbe E passant par P (mod N), c'est-à-dire vérifiant

$$y_0^2 \equiv 4x_0^3 - g_2 x_0 - g_3 \mod N .$$

On calcule ensuite pour $j = 1,2,\ldots,k$:

$$P_{j!} = j!P = (x_{j!}, y_{j!}, t_{j!}) \mod N .$$

On utilise pour cela les formules projectives. On calcule également $\text{pgcd}(t_{j!}, N)$.

Supposons que $p|N$ et que E_p, le nombre de points de (E) mod p, divise k!. Alors $P_{k!} = 0$, $t_{k!}$ sera un multiple de p, et $(t_{k!}, N)$ fournit un diviseur de N, en général non trivial.

Comme dans la méthode précédente, on suppose que pour différentes courbes elliptiques E, E_p est un nombre au hasard voisin de p, et donc que la probabilité que E_p divise k! est voisine de $\frac{1}{p}\Psi(p,k)$. On peut donc espérer raisonnablement qu'en essayant S courbes elliptiques avec $S = p/\Psi(p,k)$, on réussira à factoriser N.

On pose $L = \exp(\sqrt{\log N \log\log N})$. On suppose que $p = N^a$, $0 < a < 1$. On choisit $k = L^b$, ce qui entraine $S = L^{a/2b+O(1)}$. Pour une courbe, le calcul de k!P nécessite $O(k \log k)$ opérations, soit, pour S courbes, $L^{a/2b+b+O(1)}$ opérations. Pour détecter un facteur premier voisin de N^a, on choisit $b = \sqrt{a/2}$, et l'algorithme est en $L^{\sqrt{2a}+O(1)}$.

Par rapport à la méthode de Schnorr, cette méthode a comme avantage que la loi de groupe est plus facile à calculer, et qu'elle détecte plus rapidement les petits facteurs premiers. Les deux méthodes ont en commun d'utiliser très peu de mémoires, et d'autoriser le calcul en parallèle : on peut imaginer 1 000 micro-ordinateurs choisissant chacun des courbes elliptiques au hasard, et faisant la course.

Du point de vue algorithmique, cette méthode soulève le problème du calcul rapide de la loi de groupe d'une courbe elliptique.

Le crible quadratique de C. Pomerance (cf. [Pom 2]).

On veut factoriser N. Le paramètre théorique qui servira à la mesure de la vistesse de l'algorithme est :

$$L = \exp(\sqrt{\log N \log\log N}) .$$

En pratique, on choisira des bornes voisines des bornes théoriques de façon à minimiser le temps sur des exemples.

1) On considère le polynôme du second degré en A :

$$Q(A) = ([\sqrt{N}] + A)^2 - N$$

où [t] est la partie entière de t.

Pour les valeurs de $A = 0, 1, \ldots, A\max \sim L^{\sqrt{9/8}}$ on observe que $Q(A) \sim 2A\sqrt{N}$, et n'a guère plus de la moitié des chiffres de N.

2) On choisit une "base" de nombres premiers, par exemple tous les nombres premiers $\leqslant B \sim L^{1/\sqrt{8}}$:

$$p_1 = 2,\ p_2 = 3, \ldots, p_k \leqslant B .$$

3) On recherche une famille $A_1, A_2, \ldots, A_{k+1}$ dans l'intervalle $(-A\max, A\max)$ telle que $Q(A_i)$ se factorise complètement sur la base de nombres premiers. On verra plus tard comment s'effectue la recherche des A_i. Comme tous les facteurs de $Q(A_i)$ appartiennent à la base, on peut écrire :

$$Q(A_i) = p_1^{a_{i,1}} . p_2^{a_{i,2}} \ldots . p_k^{a_{i,k}} \quad \text{avec} \quad a_{i,j} \geqslant 0 .$$

4) Soit $K = \mathbb{Z}/2\mathbb{Z}$ le corps à deux éléments. Pour chaque i, on note $\vec{v}_i$ le vecteur de K^k dont les coordonnées sont :

$$\vec{v}_i = (a_{i,1} \bmod 2,\ a_{i,2} \bmod 2, \ldots, a_{i,k} \bmod 2) .$$

La famille de vecteurs $\vec{v}_1,\ldots,\vec{v}_{k+1}$ est linéairement dépendante. Il existe une combinaison linéaire du type :

$$\vec{v}_{i_1} + \vec{v}_{i_2} + \ldots + \vec{v}_{i_s} = 0 .$$

On obtient une telle combinaison par la méthode du pivot de Gauss en un temps $O(L^{\sqrt{9/8}})$. On observe que ces calculs sur le corps K sont très bien adaptés aux ordinateurs.

5) On pose :

$$x = p_1^{(a_{i_1,1}+a_{i_2,1}+\ldots+a_{i_s,1})/2} \ldots p_k^{(a_{i_1,k}+\ldots+a_{i_s,k})/2} .$$

On a :

$$Q(A_{i_1})Q(A_{i_2})\ldots Q(A_{i_s}) \equiv x^2 \mod N .$$

On pose :

$$y = ([\sqrt{N}] + A_{i_1})\ldots([\sqrt{N}] + A_{i_s}) .$$

On a :

$$y^2 \equiv x^2 \mod N$$

et p.g.c.d. (y-x,N) est un facteur de N en général non trivial.

Comment trouver les A tels que les facteurs de Q(A) soient petits ? Si p divise Q(A) il divise aussi Q(A+p),Q(A+2p)... d'où la méthode de crible.

On initialise un tableau T(A) en simple précision pour $-A_{max} \leqslant A \leqslant A_{max}$, en posant :

$$T(A) = \text{Log } Q(A) .$$

Pour chacun des nombres premiers $p = p_1,\ldots,p_k$ de la base, et pour leurs puissances p^α, on soustrait $\log p$ de T(A) chaque fois que Q(A) est multiple de p^α. Les A pour lesquels Q(A) est multiple de p^α appartiennent à deux progressions arithmétiques de raison p^α qu'il est facile de déterminer.

A la fin du criblage, les A recherchés, tels que Q(A) n'ait que $p_1,\ldots,p_k$ comme facteurs premiers vérifient théoriquement $T(A) = 0$; pratiquement, à cause des erreurs d'arrondi, T(A) sera petit. Les autres A auront pour Q(A) un facteur premier $\geqslant B$, et donc $T(A) \geqslant \log B$, ce qui est un nombre relativement grand. On distinguera donc sans peine (même avec un calcul en simple précision à l'ordinateur) les A tels que Q(A) se factorise dans la base, et pour ceux-là on détermine par division successive par $p_1,p_2,\ldots,p_k$ la factorisation complète de Q(A). Pour que l'algorithme fonctionne, il faut pouvoir fabriquer suffisamment de tels A.

Comme Q(A) est voisin de $\sqrt{N}$, on peut supposer que la probabilité que Q(A) ait tous ses facteurs $\leqslant B$ est environ $\frac{1}{\sqrt{N}} \Psi(\sqrt{N},B)$. Le nombre de Q(A) qui se factorise complètement sur la base de nombres premiers serait

$$L^{\sqrt{9/8}} \frac{1}{\sqrt{N}} \quad (\sqrt{N},L^{1/\sqrt{8}}) = L^{1/\sqrt{8}+O(1)}$$

c'est-à-dire voisin de k. On peut donc prévoir un algorithme en $L^{\sqrt{9/8}+O(1)}$. En fait un récent algorithme, dû à D. Wiedemann, permet d'inverser des matrices carrées d'ordre n à coefficients dans un corps fini en $O(n^{2+\varepsilon})$. En utilisant cet algorithme, au lieu de la méthode de Gauss, on obtient une estimation en $L^{1+O(1)}$, comme pour les deux précédentes méthodes de factorisation.

Cet algorithme nécessite une grande mémoire pour stocker les $a_{i,j}$. Mais c'est actuellement le plus rapide : il détient le "record" pour avoir cassé $(10^{71}-1)/9$.

Références

[Adl] L.M. ADLEMAN, C. POMERANCE, R.S. RUMELY : On distinguishing prime numbers from composite numbers. Ann. Math. 117 (1983), p. 173-206.

[Can] E.R. CANFIELD, P. ERDÖS and C. POMERANCE : On a problem of Oppenheim concerning "Factorisatio Numerorum", J. Number Theory 17, 1983, p. 1-28.

[Coh 1] H. COHEN : Méthodes de factorisation et nombre de classes des corps quadratiques, Séminaire D.P.P., Paris, 14^{e} année 1972-73, G7.

[Coh 2] H. COHEN and H.W. LENSTRA, Jr : Primality testing and Jacobi Sums, Math. of Comp. 42, 1984, p. 297-330.

[Coh 3] H. COHEN and A.K. LENSTRA : Implementation of a new primality test, preprint, Center for mathematics and Computer Science, Amsterdam.

[De B] N.G. DE BRUIJN : On the number of positive integers $\leqslant x$ and free of prime factors $> y$, II, Nederl. Akad. Wetensch. Proc. Ser. A, 69, 1966, p. 239-247 = Indag. Math., 28, 1966, p. 239-247.

[Dix] J.D. DIXON : Factorization and primality tests, Amer. Math. Monthly 91, 1984, p.333-352.

[Guy] R.K. GUY : How to factor a number, Congressus Numerantium XVI, Proc. of the fifth Manitoba Conference on numerical mathematics, 1975, p. 49-89.

[Hil] A. HILDEBRAND and G. TENENBAUM : On integers free of large prime factors, to appear.

[Kob] N. KOBLITZ : Introduction to elliptic curves and modular forms, Springer Verlag, 1984, Graduate texts in mathematics n° 97.

[Len] H.W. LENSTRA Jr : Prime factorization using elliptic curves, to appear.

[Mai] H. MAIER : On integers free of large prime divisors, Preprint.

[Mon] L. MONIER : Algorithmes de factorisation d'entiers, Thèse de 3^{e} cycle, Orsay, 1980.

[Nic] J.-L. NICOLAS : Tests de primalité, Expositiones Mathematicae 2, 1984, p. 223-234.

[Pol] J.-M. POLLARD : Theorems on factorization and primality testing, Proc. Cambridge Philos. Soc. 76, 1974, p. 521-528.

[Pom 1] C. POMERANCE : Analysis and comparison of some integer factoring algorithms, Computational methods in number theory, R. Tijdeman, H. Lenstra (Eds), Mathematisch Centrum, Amsterdam, Tract 154, 1982, p. 89-139

[Pom 2] C. POMERANCE : The quadratic sieve factoring algorithm, à paraitre aux compte-rendus d'EUROCRYPT 84.

[Rie] H.I. RIESEL : Prime numbers and computer methods for factorization, Birkhäuser, 1985, progress in Math. Vol. 57.

[Sch] C.P. SCHNORR and H.W. LENSTRA Jr : A Monte-Carlo Factoring Algorithm with linear storage, Math. of Comp. 43, 1984, p. 289-311.

[Sha] D. SHANKS : Class number, a theory of factorization and genera, Proc. Symp. Pure Math. 20, A.M.S., 1971, p. 415-440.

[Sil] J. SILVERMAN : Arithmetic of elliptic curves, à paraitre, Springer Verlag, Graduate texts in mathematics.

[Wie] D. WIEDEMANN : Solving sparse linear equations over finite fields, preprint.

Jean-Louis NICOLAS
U.E.R. des Sciences de Limoges
Dpt de Mathématiques-Informatique
123, avenue Albert-Thomas
87060 LIMOGES CEDEX