Algorithmes pour l'arithmétique II
Cours 7

Julien Lavauzelle

Université Paris 8

Master 2 ACC et CSSD – Algorithmes pour l'arithmétique

19/01/2020

La semaine dernière

Questions ?

Motivation

La semaine dernière on a vu des méthodes **spéciales** pour factoriser des entiers N :

1. divisions successives, algorithme ρ de Pollard pour trouver de petits facteurs de N
2. algorithme de Fermat pour trouver des facteurs très proches de $\sqrt{N}$
3. méthode $p-1$ de Pollard et $p+1$ de Williams pour trouver des facteurs p tels que $p-1$ ou $p+1$ est superfriable

On a rapidement présenté la méthode ECM de Lenstra, qui utilise le groupe des points de courbes elliptiques aléatoires pour factoriser en **temps sous-exponentiel**

$$O\Big(\exp(\sqrt{2\log p \log\log p})\Big)$$

où p est le plus petit facteur de N.

Dans cette séance, on va voir l'algorithme de **crible quadratique** qui permet de factoriser n'importe quel entier N en temps

$$O\Big(\exp(\sqrt{\log N \log\log N})\Big)$$

Son extension, l'algorithme de **crible algébrique** (ou crible par corps de nombres généralisé) (*general number field sieve*, NFS) atteint une complexité encore meilleure :

$$O\Big(\exp((\tfrac{64}{9}\log N)^{1/3}(\log\log N)^{2/3})\Big)\,.$$

Complexité

Objectif : représenter des grandeurs sous-exponentielles $n^\alpha \ll f(n) \ll 2^{\beta n}$.

Définition. Notation L :

$$L_n[a,b] := \exp\left(b\, n^a\, (\log n)^{1-a}\right).$$

On prendra souvent exp et log en base 2.

Exemples :

- Les fonctions exponentielles $2^{\beta n}$ sont des $L_n[1,\beta]$.
- Les fonctions polynomiales n^α sont des $L_n[0,\alpha]$.

On va utiliser cette notation pour les algorithmes de factorisation. On s'intéresse donc à leur complexité en fonction de $\log_2 N$ ou de $\log_2 p$. Ainsi :

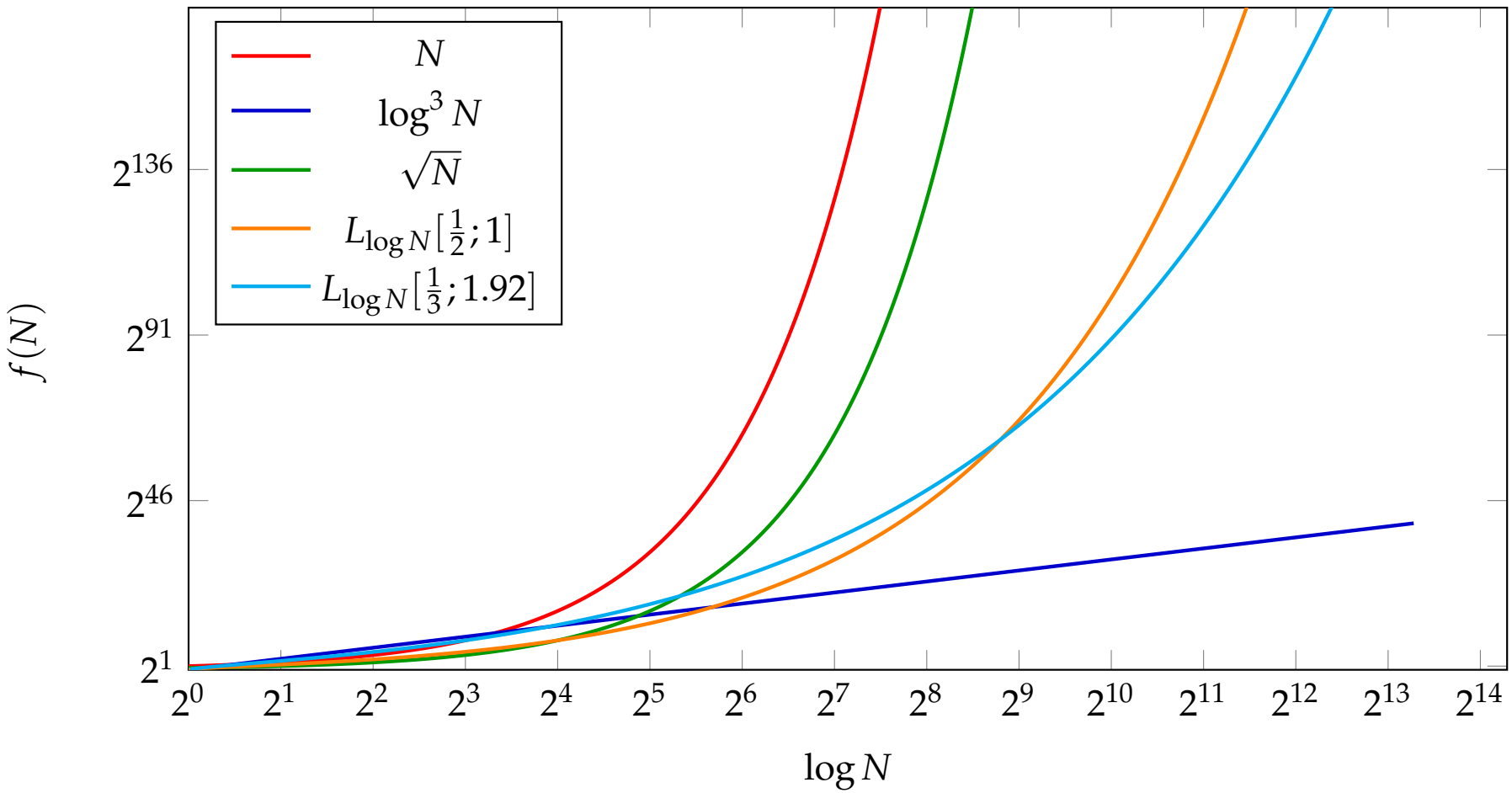
- La complexité de la méthode ρ est en $O(\sqrt{p}) = O(L_{\log p}[1, \frac{1}{2}])$.
- La complexité de ECM est en $O(\exp(\sqrt{2 \log p \log\log p})) = O(L_{\log p}[\frac{1}{2}, \sqrt{2}])$.

Complexité

Quelques **exemples** de comportement de fonctions (échelle « log log »).

Rappel :

- 2^{80} : très, très difficile à calculer $\simeq 2^{30}$ op./s sur un processeur
- 2^{128} : supposé inatteignable



Crible quadratique : origines

Méthode de Fermat : si on arrive à écrire $N = a^2 - b^2$, alors $N = (a-b)(a+b)$ donne une factorisation de N.

Raffinement de l'idée (Kraitchik, puis Dixon) :

> **Idée :** Si on obtient « seulement » $N \mid a^2 - b^2 = (a-b)(a+b)$, alors on peut espérer que $\mathrm{pgcd}(a-b, N)$ ou $\mathrm{pgcd}(a+b, N)$ donne un facteur propre de N.

Exemple. $N = 91$. On a $4^2 = 16$ et $17^2 = 289 \equiv 16 \bmod 91$. Puis, $\mathrm{pgcd}(4+17, N) = 7$. Ici, on a également $\mathrm{pgcd}(17-4, N) = 13$.

Remarque. On obtient un facteur propre lorsque $a \not\equiv \pm b \bmod N$.

Lemme. Si N admet t diviseurs premiers distincts, alors l'équation $x^2 = 1$ admet 2^t solutions modulo N.

Conséquence. Si N est composé ($t \geq 2$) et si $a^2 \equiv b^2 \bmod N$ ont été trouvés « aléatoirement », alors il y a plus d'une chance sur deux pour que $a \not\equiv \pm b \bmod N$.

Question. Comment trouver a, b tels que $a^2 \equiv b^2 \bmod N$?

Comment trouver $a^2 \equiv b^2 \mod N$?

Idée. Étant donnée une borne de lissité $B \geq 2$:

1. on collecte une quantité importante d'éléments B-friables de la forme
$$Q(x) := (x + \lceil \sqrt{N} \rceil)^2 - N$$
2. on essaie de combiner certains $Q(x_i)$ pour que
$$Q(x_1) \cdots Q(x_k) \equiv b^2 \mod N$$

Alors, on aura obtenu $a^2 \equiv b^2 \mod N$, où $a := (x_1 + \lceil \sqrt{N} \rceil) \cdots (x_k + \lceil \sqrt{N} \rceil)$.

Exemple : $N = 1649$ donne $\lceil \sqrt{N} \rceil = 41$. Puis, modulo N, on obtient

$$\begin{cases} (x=0) & 41^2 = 1681 & \equiv 32 & \equiv 2^5 & \mod N \\ (x=1) & 42^2 = 1764 & \equiv 115 & \equiv 5 \times 23 & \mod N \\ (x=2) & 43^2 = 1849 & \equiv 200 & \equiv 2^3 \times 5^2 & \mod N \end{cases}$$

Ainsi, on note que
$$41^2 \times 43^2 \equiv 2^8 \times 5^2 \equiv (2^4 \times 5)^2 \mod N\,.$$

Il résulte que $1763^2 \equiv 80^2 \mod N$, on a bien $114 \equiv 1763 \not\equiv \pm 80 \mod N$. Puis, on obtient $\mathrm{pgcd}(114 - 80, 1649) = 17$ un facteur propre de $N = 1469$.

Remarque. Avec l'algorithme de Fermat, on aurait dû aller jusqu'à 57^2 pour factoriser.

Comment trouver $a^2 \equiv b^2 \mod N$?

Idée. Étant donnée une borne de lissité $B \geq 2$:

1. on crée une base de facteurs premiers $\mathcal{P} = \{p_1, \ldots, p_s\}$, tous inférieurs à B
2. on collecte une quantité importante d'éléments qui se décomposent sur $\mathcal{P}$, et de la forme
$$Q(x) := (x + \lceil \sqrt{N} \rceil)^2 - N$$
3. on essaie de combiner certains $Q(x_i)$ pour que
$$Q(x_1) \cdots Q(x_k) \equiv b^2 \mod N$$

Alors, on aura obtenu $a^2 \equiv b^2 \mod N$ où $a := (x_1 + \lceil \sqrt{N} \rceil) \cdots (x_k + \lceil \sqrt{N} \rceil)$.

Trois questions :

1. quelle base de facteurs premiers $\mathcal{P}$ choisir ?
2. comment obtient-on ces éléments B-friables de la forme $Q(x)$?
   $\rightarrow$ technique de crible (ici, quadratique)
3. comment savoir quels $Q(x_i)$ multiplier pour obtenir un carré modulo N ?
   $\rightarrow$ algèbre linéaire dans $\mathbb{F}_2$

Étape I : base de facteurs

Première étape : construction de la **base de facteurs**.

Si $x < \sqrt{N}/3$, alors on a

$$Q(x) = (x + \lceil \sqrt{N} \rceil)^2 - N \simeq x^2 + 2\sqrt{N}x < N.$$

Donc $(Q(x) \bmod N)$ vaut $Q(x)$, et pour tout premier $p \geq 2$, on a

$$p \mid Q(x) \implies N \text{ est un carré modulo } p$$

Pour constituer la base de facteurs, on peut donc considérer uniquement les premiers p tel que $\left(\frac{N}{p}\right) = 1$ et $p \leq B$.

Exemple : $N = 369713 = 457 \times 809$. On choisit ici $B = 21$. On a alors

p	2	3	5	7	11	13	17	19
$\left(\frac{N}{p}\right)$	1	-1	-1	1	1	-1	-1	1

La base de facteurs est donc $\{2, 7, 11, 19\}$.

Étape III : algèbre linéaire

Troisième étape : l'**algèbre linéaire**.

Soit $\mathcal{P} = \{p_1, \ldots, p_s\}$ la base de facteurs construite précédemment.

Supposons que l'on ait collecté t éléments $Q(x_1), \ldots, Q(x_t)$ tels que les $Q(x_i) \mod N$ se décomposent dans $\mathcal{P}$ (étape II). On peut alors écrire $Q(x_i) \mod N$ sous la forme

$$p_1^{e_1^{(i)}} \times p_2^{e_2^{(i)}} \times \cdots \times p_s^{e_s^{(i)}}$$

et on note $\boldsymbol{e}(x_i) = (e_1^{(i)}, \ldots, e_s^{(i)})$.

Alors, on a $Q(x_{i_1})Q(x_{i_2}) \ldots Q(x_{i_k}) \equiv p_1^{e_1} \cdots p_s^{e_s} \mod N$ où

$$(e_1, \ldots, e_s) \;=\; \boldsymbol{e}(x_{i_1}) + \boldsymbol{e}(x_{i_2}) + \cdots + \boldsymbol{e}(x_{i_k})\,.$$

Lemme. L'élément $Q(x_1) \ldots Q(x_k)$ est un carré modulo N si et seulement si

$$\sum_{i=1}^{k} \boldsymbol{e}(x_i) = \mathbf{0} \mod 2\,.$$

Étape III : algèbre linéaire

Lemme. L'élément $Q(x_1)\dots Q(x_k)$ est un carré modulo N si et seulement si

$$\sum_{i=1}^{k} \boldsymbol{e}(x_i) = \mathbf{0} \mod 2\,.$$

On va alors construire une matrice $\boldsymbol{M}$ entière de taille $(t \times s)$ telle la i-ème ligne de $\boldsymbol{M}$ est le vecteur ligne $\boldsymbol{e}(x_i)$:

$$\boldsymbol{M} = \begin{bmatrix} e_1^{(1)} & e_2^{(1)} & \dots & e_s^{(1)} \\ e_1^{(2)} & \dots & \dots & e_s^{(2)} \\ \vdots & & & \\ & & & \\ e_1^{(t)} & \dots & \dots & e_s^{(t)} \end{bmatrix} \in \mathbb{N}^{t\times s}$$

Pour obtenir un carré modulo N de la forme $Q(x_1)\dots Q(x_k)$, il suffit donc de chercher un élément du noyau à gauche de la matrice $(\boldsymbol{M} \bmod 2)$, car

$$\boldsymbol{u} = (u_1, \dots, u_t) \in \mathbb{F}_2^t \text{ vérifie } \boldsymbol{uM} = \mathbf{0} \quad \Longrightarrow \quad \sum u_i \boldsymbol{e}(x_i) = \mathbf{0}\,.$$

Remarque. Pour que la matrice $(\boldsymbol{M} \bmod 2)$ ait un noyau (à gauche) non-nul, il est suffisant que le nombre de lignes non-nulles de $(\boldsymbol{M} \bmod 2)$ soit plus grand que le nombre de ses colonnes. En pratique, on souhaite donc que t soit sensiblement plus grand que s.

Étape III : exemple

Pour $N = 369713$ et $\mathcal{P} = \{2, 7, 11, 19\}$, supposons que l'on ait obtenu les éléments suivants :

x_i	6	8	24	106	120
$Q(x_i)$	8512	10976	30976	141512	161728

La matrice $\boldsymbol{M}$ est alors :

$$\boldsymbol{M} = \begin{bmatrix} 6 & 1 & 0 & 1 \\ 5 & 3 & 0 & 0 \\ 8 & 0 & 2 & 0 \\ 3 & 2 & 0 & 2 \\ 6 & 1 & 0 & 2 \end{bmatrix}$$

Modulo 2, on obtient :

$$(\boldsymbol{M} \mod 2) = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Puis, on trouve un élément $\boldsymbol{u}$ tel que $\boldsymbol{uM} = \boldsymbol{0}$, par exemple

$$\boldsymbol{u} = (0, 1, 1, 1, 1) \text{ ou ecnore } \boldsymbol{u} = (0, 0, 1, 0, 0)$$

Étape III : exemple

x_i	6	8	24	106	120
$Q(x_i)$	8512	10976	30976	141512	161728

$$\boldsymbol{M} = \begin{bmatrix} 6 & 1 & 0 & 1 \\ 5 & 3 & 0 & 0 \\ 8 & 0 & 2 & 0 \\ 3 & 2 & 0 & 2 \\ 6 & 1 & 0 & 2 \end{bmatrix} \qquad (\boldsymbol{M} \mod 2) = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{u} = (0,1,1,1,1)$$

On calcule $\boldsymbol{u} \cdot \boldsymbol{M} = (22,6,2,4)$, donc on va construire

1. b une racine carrée de $Q(x_2)Q(x_3)Q(x_4)Q(x_5) = p_1^{22}\, p_2^6\, p_3^2\, p_4^4$, c'est-à-dire

$$b = p_1^{11}\, p_2^3\, p_3\, p_4^2 \equiv 369672 \mod N$$

2. a le produit des $x_i + \lceil \sqrt{N} \rceil$ correspondant, donc

$$a = (x_2 + \lceil \sqrt{N} \rceil)(x_3 + \lceil \sqrt{N} \rceil)(x_4 + \lceil \sqrt{N} \rceil)(x_5 + \lceil \sqrt{N} \rceil) \equiv 102784 \mod N$$

Puis on a (finalement) :

$$\mathrm{pgcd}(a-b, N) = 457 \quad \text{et} \quad \mathrm{pgcd}(a+b, N) = 809\,.$$

Étape II : effritement (méthode naïve)

Deuxième étape : **effritement**, ou phase de collection

But : pour $t \geq s$, trouver t éléments de la forme $Q(x)$ qui se décomposent dans la base de facteurs $\mathcal{P} = \{p_1, \ldots, p_s\}$

Une première idée est de chercher ces éléments de manière itérative (x croissant de 1 en 1).

COLLECTION D'ÉLEMENTS FRIABLES (MÉTHODE NAÏVE)

1. $i \leftarrow 0, \quad x = 0, \quad r = \lceil \sqrt{N} \rceil, \quad q = r^2 - N$
2. `liste_elements` $= []$
3. $\boldsymbol{M} = []$
4. **Tant que** $i < t$:
   - 4.1 **Si** q se décompose sur $\mathcal{P}$ comme $q = p_1^{e_1} \ldots p_s^{e_s}$
     - Ajouter le vecteur $(e_1, \ldots, e_s)$ comme dernière ligne de $\boldsymbol{M}$
     - Ajouter (x, q) à `liste_elements`
     - $i \leftarrow i + 1$
   - 4.2 $q \leftarrow q + 2(x + r) + 1$
   - 4.3 $x \leftarrow x + 1$
5. **Retourner** $\boldsymbol{M}$ et `liste_elements`

Exemple

$N = 369713$ et $\mathcal{P} = \{2, 7, 11, 19\}$.

x	$Q(x)$	$(e_1, \ldots, e_4)$	reste
0	1168	$[4,0,0,0]$	73
1	2387	$[0,1,1,0]$	31
2	3608	$[3,0,1,0]$	41
3	4831	$[0,0,0,0]$	4831
4	6056	$[3,0,0,0]$	757
5	7283	$[0,0,0,0]$	7283
6	**8512**	$[6,1,0,1]$	**1**
7	9743	$[0,0,0,0]$	9743
8	**10976**	$[5,3,0,0]$	**1**
9	12211	$[0,0,0,0]$	12211
⋮	⋮	⋮	⋮
23	29711	$[0,0,1,0]$	2701
24	**30976**	$[8,0,2,0]$	**1**
25	32243	$[0,0,0,1]$	1697
⋮	⋮	⋮	⋮
106	**141512**	$[3,2,0,2]$	**1**
⋮	⋮	⋮	⋮
120	**161728**	$[6,1,0,2]$	**1**

On obtient la liste

$$\{(6, 8512), (8, 10976), (24, 30976), (106, 141512), (120, 161728)\}$$

Cela donne la matrice

$$\boldsymbol{M} = \begin{bmatrix} 6 & 1 & 0 & 1 \\ 5 & 3 & 0 & 0 \\ 8 & 0 & 2 & 0 \\ 3 & 2 & 0 & 2 \\ 6 & 1 & 0 & 2 \end{bmatrix}$$

Problème. Pour chaque ligne calculée, on fait beaucoup de tests de divisibilité qui échouent. Autrement dit, il y a **beaucoup** de zéros dans les $(e_1, \ldots, e_s)$ calculés.

Étape II : effritement (méthode de criblage)

Idée : pour être plus efficace, on va collecter ces éléments par une méthode de **crible**.

Remarque. La notion de crible a déjà été vue pour établir une liste de nombres premiers : c'est le **crible d'Eratostène**.

Pour $A \geq s$ un entier dépendant de B qu'on déterminera plus tard :

MÉTHODE DE CRIBLE QUADRATIQUE (VERSION PÉDAGOGIQUE)

1. On initialise un tableau $T = [Q(0), Q(1), \ldots, Q(A)]$
2. Pour chaque premier $p \in \mathcal{P}$:
   - 2.1 $e = 1$
   - 2.2 Tant que $p^e \leq A$:
     - On cherche les solutions $0 \leq y < p^e$ de $Q(y) \equiv N \mod p^e$
     - On divise par p tous les $T[x]$ où x est de la forme $y + kp^e$ (criblage)
     - On incrémente $e \leftarrow e + 1$
3. **Retourner** tous les $(x, Q(x))$ tels que $T[x] = 1$.

Remarques :

- la recherche de solution de $Q(y) \equiv N \mod p$ est essentiellement une recherche de racine carrée ($\rightarrow$ Tonelli-Shanks, ou Cipolla)
- on peut déduire très efficacement les solutions modulo p^e des solutions modulo p^{e-1} (relèvement de Hensel).

Exemple de crible

Pour $N = 73217 = 211 \times 347$ (nouveau N) avec la base de facteurs $\mathcal{P} = \{2, 7, 11, 13\}$.

Les étapes successives de l'algorithme sont :

p	e	solutions	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
			224	767	1312	1859	2408	2959	3512	4067	4624	5183	5744	6307	6872	7439	8008	8579
2	1	[0]	112	767	656	1859	1204	2959	1756	4067	2312	5183	2872	6307	3436	7439	4004	8579
	2	[2, 0]	56	767	328	1859	602	2959	878	4067	1156	5183	1436	6307	1718	7439	2002	8579
	3	[2, 4, 6, 0]	28	767	164	1859	301	2959	439	4067	578	5183	718	6307	859	7439	1001	8579
	4	[2, 8, 10, 0]	14	767	82	1859	301	2959	439	4067	289	5183	359	6307	859	7439	1001	8579
	5	[0, 2]	7	767	41	1859	301	2959	439	4067	289	5183	359	6307	859	7439	1001	8579
7	1	[4, 0]	1	767	41	1859	43	2959	439	581	289	5183	359	901	859	7439	143	8579
	2	[7]	1	767	41	1859	43	2959	439	83	289	5183	359	901	859	7439	143	8579
11	1	[5, 3]	1	767	41	169	43	269	439	83	289	5183	359	901	859	7439	13	8579
13	1	[3, 1]	1	59	41	13	43	269	439	83	289	5183	359	901	859	7439	1	8579
	2	[3]	1	59	41	1	43	269	439	83	289	5183	359	901	859	7439	1	8579

Et on obtient la liste $\{(0, 224), (3, 1859), (14, 8008)\}$

La matrice associée est

$$\boldsymbol{M} = \begin{bmatrix} 5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 3 & 1 & 1 & 1 \end{bmatrix}$$

Améliorations

Observation 1. Les termes du tableau peuvent être initialement grands, et leur division par des premiers est légèrement coûteuse en pratique.Pour éviter cela, on peut

- remplacer les valeurs exactes $Q(x)$ par $\lfloor \log Q(x) \rfloor$
- soustraire $\lfloor \log p \rfloor$ (précalculé) $T[x]$ au lieu de diviser $T[x]$ par p
- en fin d'algorithme, plutôt que de tester si $T[x] = 1$, on vérifie si $|T[x]| \leq \log B^2$

Observation 2. Si A est grand, il devient peu probable que $Q(x)$ soit B-friable lorsque x s'approche de A.

Idée : on remplace $Q(x)$ par des $Q_{u,v}(x) = Q(ux + v)$ où u et v sont choisis de telle sorte que $Q_{u,v}(x)$ donne des nombres « petits » modulo N, lorsque $x \in [0, A]$.

C'est la variante à **polynômes multiples** ;

- cela permet de réduire sensiblement la taille de A,
- on peut choisir avantageusement u et v pour produire beaucoup de relations.

Résumé de l'algorithme

FACTORISATION PAR CRIBLE QUADRATIQUE (PÉDAGOGIQUE)

Entrée : N un entier à factoriser
Sortie : un facteur propre de N

1. **Initialisation** : calculer $B \simeq 2^{0.5\sqrt{\log N \log\log N}}$ (on verra pourquoi)
2. Calculer la **base de facteurs** $\mathcal{P} = \{p_1, \ldots, p_s\}$ tels que $p_j \leq B$ et $\left(\frac{N}{p_j}\right) = 1$.
3. **Effritement** :
   - 3.1 Calculer la matrice $\boldsymbol{M}$ et les éléments $\{(x, Q(x))\}$ associés par criblage.
   - 3.2 Si $\boldsymbol{M}$ a un noyau à gauche nul, revenir à 2. avec $B \leftarrow 2B$.
4. **Algèbre linéaire** :
   - 4.1 Calculer une solution aléatoire $\boldsymbol{u}$ de $\boldsymbol{u} \cdot (\boldsymbol{M} \bmod 2) = \boldsymbol{0}$
   - 4.2 Calculer $a = \prod_{j \in J}(x_j + \lceil\sqrt{N}\rceil)$ et $b = \prod_{j \in J} Q(x_j)$ où $J = \{j, u_j \neq 0\}$.
5. **Si** $\{\mathrm{pgcd}(a-b, N), \mathrm{pgcd}(a+b, N)\}$ ne contient pas de facteur propre de N, revenir à l'étape 4.1.
6. **Sinon,** retourner les facteurs propres obtenus.

Qu'en est-il de la **complexité** de l'algorithme ?

Complexité du crible quadratique

D'abord, **quelques résultats de théorie des nombres.**

Soit M un entier ≥ 2.

Théorème de Tchebychev. Le nombre $\pi(M)$ de nombres premiers compris entre 2 et M vérifie

$$\alpha \frac{M}{\log M} \leq \pi(M) \leq \beta \frac{M}{\log M}\,.$$

Remarque. Les théorèmes de Hadamard et de de la Vallée-Poussin assurent $\pi(M) \sim \frac{M}{\ln M}$.

On définit la **fonction de de Bruijn** $\psi(M,B)$ comme le nombre d'entiers $\leq M$ qui sont B-friables.

Proposition. Si $\log M \ll B \ll M$, alors $\psi(M,B)$ vérifie

$$\frac{\psi(M,B)}{M} \sim \left(\frac{\log M}{\log B}\right)^{-(\log M)/(\log B)}.$$

Complexité du crible quadratique

Étape 1 : base de facteurs

- Il y a $\pi(B)$ premiers pour lequels on doit tester la résiduosité quadratique de N
  $\rightarrow$ complexité en $O(B \log B \log N)$
  $\rightarrow$ on a donc $s \simeq \pi(B)/2$ premiers dans $\mathcal{P}$

Étape 2 : effritement par crible. Soit M la taille maximale d'un entier $Q(x)$ à traiter.

- Alors, il faudra traiter $s \cdot \frac{M}{\psi(M,B)}$ entiers en moyenne
- On peut montrer que pour chaque entier du tableau, on fait $O(\log\log B)$ opérations
- La complexité est donc $C_2 = O(s \log\log B \cdot \frac{M}{\psi(M,B)}) = O(B^{\frac{\log\log B}{\log B}} u^u)$ où $u = \frac{\log M}{\log B}$
- Si $M \simeq \sqrt{N}$ et $\log B \geq \sqrt{\log N}$, alors on obtient :

$$\log C_2 \simeq \log B + \frac{\log N \log\log N}{4 \log B}$$

Cette dernière quantité se maximise pour $\log B \simeq \frac{1}{2}\sqrt{\log N \log\log N} = L_{\log N}[\frac{1}{2}, \frac{1}{2}]$, et donne

$$C_2 = \exp(\sqrt{\log N \log\log N}) = L_{\log N}[\tfrac{1}{2}, 1]\,.$$

Complexité du crible quadratique

Étape 3 : algèbre linéaire.

- On résout un système linéaire **creux** de taille $t \times s$ sur $\mathbb{F}_2$ où $t \simeq s \simeq \frac{B}{2 \log B}$
  $\rightarrow$ Par l'algorithme de Wiedemann (par exemple), on a également une complexité en

$$O(ts) = O(B^2 / \log^2 B) = O(\exp(\sqrt{\log N \log\log N})) = O(L_{\log N}[\tfrac{1}{2}, 1])$$

- Calculs terminaux (produits, pgcd) en $O(B \log^2 N)$

Conclusion. L'algorithme de crible quadratique permet de factoriser un entier N en temps

$$O(\exp(\sqrt{\log N \log\log N}))\,.$$

Long exemple final

démo avec `sage`

Crible algébrique

Le **crible algébrique** généralise l'idée du crible quadratique. L'idée est de chercher des $a^2 \equiv b^2 \mod N$ en cherchant des carrés dans des **anneaux d'entiers**.

Idée. Soit $f(X) \in \mathbb{Z}[X]$ unitaire et irréductible, et $m \in \mathbb{Z}$ qui satisfait $f(m) \equiv 0 \mod N$. En posant $\alpha = \overline{X} \in \mathbb{Z}[X]/(f)$, on a $\mathbb{Z}[X]/(f) = \mathbb{Z}[\alpha]$.

On considère ensuite le morphisme d'anneaux

$$\begin{array}{rccc} \phi & : \mathbb{Z}[\alpha] & \to & \mathbb{Z}/N\mathbb{Z} \\ & P(\alpha) & \mapsto & \overline{P}(m) \mod N \end{array}$$

Alors on cherche $P \in \mathbb{Z}[X]$ tel que $P(\alpha)$ est un carré $z^2 \in \mathbb{Z}[\alpha]$ et $\overline{P}(m)$ est un carré $b^2 \in \mathbb{Z}/N\mathbb{Z}$. Si $a = \phi(z)$, alors $a^2 \equiv b^2 \mod N$.

Ensuite, pour trouver les éléments P, on va cribler les **normes** de valeurs de la forme $u + v\alpha \in \mathbb{Z}[\alpha]$ (ce sont les petits p_i pour le crible quadratique). [...]

Au final, on obtient une **complexité** du crible algébrique en

$$O\Big(\exp((\tfrac{64}{9} \log N)^{1/3} (\log\log N)^{2/3}) \Big).$$

Plus de détails dans :

A Tale of Two Sieves. C. Pomerance. Notices of the AMS. **1996**. . Plutôt accessible `[lien]`.

Prime Numbers, a Computational Perspective. R. Crandall, C. Pomerance. Springer. **2001**.

En pratique

Pour extraire des facteurs de taille $\leq$ 60-70 chiffres (« moyens »), on utilise la **méthode ECM**. Le record de factorisation d'ECM a produit un facteur de 83 chiffres.

Pour des facteurs de taille plus importante, on utilise le **crible algébrique**.

Le **record** de factorisation de modules RSA est RSA-250, effectuée en février 2020 :

$$
\begin{aligned}
&21403246502407449612644230728393335630086147151447550177977549208814180234471401366433455190958046796109928518724709145876873962619215573630474547705208051190564931066876915900197594056934574522305893259766974716817380693648946998715784949759374979 37\\
&= 64135289477071580278790190170577389084825014742943447208116859632024532344630238623598752668347708737661925585694639798853367\\
&\times 33372027594978156556226010605355114227940760344767554666784520987023841729210037080257448673296881877565718986258036932062711
\end{aligned}
$$

Temps de factorisation : équivalent 2700 années (oui !) de calcul sur 1 coeur.

Source :

`https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2020-February/001166.html`

En pratique

Une **implémentation** proche de l'état de l'art de la recherche : CADO-NFS

- majoritairement codé en `C`, `C++`, plus de l'assembleur pour accélérer certains calculs
- licence LGPL (libre), développé principalement en France (notamment une équipe Inria à Nancy)
- permet de factoriser sur un processeur standard (Intel(R) Xeon(R) CPU E5-2650 @2.00GHz) : (source : site web cado-nfs)

RSA-120	RSA-130	RSA-140	RSA-155
1,9 heure	7,5 heures	23 heures	5,3 jours

- utilisé pour la factorisation record

CADO-NFS, An Implementation of the Number Field Sieve Algorithm. The CADO-NFS Development Team. . **2017**. `http://cado-nfs.gforge.inria.fr`

Fin du cours

Questions ?