$10^{2098959}$

C'est à quelque chose près la période du programme :

$r := 6972593$, $s := 3037958$
tant que vrai **faire**
  $t \leftarrow a_{r-1}$
  $a_i \leftarrow a_{i-1}$, $1 \leq i < r$
  $a_0 \leftarrow t$
  $a_s \leftarrow a_s \oplus t$
fin tant que.

où les a_i, $0 \leq i < r$, sont des bits initialisés de façon quelconque, et $\oplus$ est le « ou exclusif » ($0 \oplus 1 = 1 \oplus 0 = 1, 0 \oplus 0 = 1 \oplus 1 = 0$).

Ce programme est un *générateur pseudo-aléatoire* utilisant le *trinôme primitif*

$$f = x^{6972593} + x^{3037958} + 1$$

sur GF(2), le corps fini à deux éléments 0 et 1, vérifiant $0 + x = x + 0 = x$, $1+1 = 0$, $0 \times x = x \times 0 = 0$, $1 \times 1 = 1$. En effet, on peut récrire ce générateur de la façon suivante :

$p = a_0 + a_1 x + \ldots + a_{r-1} x^{r-1}$
tant que vrai **faire**
  $p \leftarrow xp \bmod f$
fin tant que.

La bogue. Avec Richard Brent (Université d'Oxford) et Samuli Larvala (Université d'Helsinki), nous avons débuté en 2001 la recherche de trinômes primitifs de grand degré sur GF(2), motivés par un article paru dans la revue *Mathematics of Computation*. Les auteurs de celui-ci, après une recherche complète pour $r = 859433$, avaient trouvé un seul trinôme primitif, à savoir $x^r + x^s + 1$ pour $s = 288477$ (avec son symétrique $x^r + x^{r-s} + 1$). En vérifiant leurs calculs, nous avons trouvé un second trinôme pour $s = 170340$! Ils ont admis plus tard qu'une « bogue » s'est glissée dans leur programme. Depuis, nous avons traité le degré 756839 (non traité jusque là), pour lequel 3 trinômes primitifs ont été trouvés ($s = 215747$, 267428 et 279695), puis le degré 3021377, pour lequel 2 trinômes primitifs ont été trouvés ($s = 361604$ et 1010202).

Le degré 6972593. Après avoir fini la recherche pour 3021377, il était tentant de s'attaquer à l'exposant de Mersenne suivant : 6972593. La recherche pour 3021377 ayant pris 8 mois, et le coût de la recherche étant proportionnel à r^3, nous avons estimé à 8 ans le temps de recherche pour $r = 6972593$ sur les ordinateurs de nos laboratoires. Pour mener à bien cette recherche en un temps raisonnable, il nous fallait donc une puissance de calcul supérieure d'un ordre de grandeur. C'est la raison pour laquelle nous avons fait appel au CINES en France et à l'APAC en Australie, pays d'où est originaire Richard Brent.

Principe de la recherche. Pour chaque valeur de s, il suffit de calculer $h_s := x^{2^r} \bmod f$ où $f = x^r + x^s + 1$: f est irréductible (donc primitif) si et seulement si $h_s = x$. h_s se calcule par exponentiation binaire :

$p \leftarrow x$
pour i **de** 1 **à** r **faire**
  $q \leftarrow p^2$
  $p \leftarrow q \bmod (x^r+x^s+1)$
si $p = x$ renvoyer *primitif*

Le calcul $q \leftarrow p^2$ est trivial sur GF(2) car le carré du polynôme $a_0 + a_1 x + \ldots + a_{r-1}x^{r-1}$ est simplement $a_0 + a_1 x^2 + \ldots + a_{r-1}x^{2r-2}$; en effet, les termes croisés $2a_i a_j x^{i+j}$ sont nuls car $2 = 0$ sur GF(2). La seule opération coûteuse est donc la réduction de q modulo $x^r + x^s + 1$; nous avons pour cela mis au point un nouvel algorithme — déjà utilisé pour $r = 3021377$ — qui permet d'effectuer cette réduction avec moins d'opérations et d'espace mémoire. La gestion de la mémoire est en effet cruciale pour de tels degrés : pour $r = 6972593$, chaque polynôme utilise près de 900Ko de mémoire. La performance de notre programme de recherche se dégrade considérablement lorsque cette taille dépasse celle du cache secondaire. C'est le cas de la plupart des stations de travail actuelles, qui ont au mieux 512Ko de cache secondaire. Sur les (anciens) processeurs R12000 du CINES, qui disposent de 8Mo de cache secondaire, notre programme vérifie une valeur de s en 15 heures et demie environ.

En pratique, nous ne calculons pas $x^{2^r} \bmod (x^r+x^s+1)$ pour les quelques $r/2$ valeurs possibles de s. En effet, une méthode de crible — qui consiste à détecter les polynômes ayant un facteur de degré $\leq k$ pour k bien choisi, ici $k \approx 26$ — permet d'éliminer rapidement environ 93% des trinômes. La vérification des quelques 230.000 trinômes restants ne nécessite donc que 3.5 millions d'heures de R12000 !

Résultats obtenus. La recherche pour $r = 6972593$ a commencé à l'été 2001. Fin octobre 2002, après environ un an et demi de calculs, nous avons vérifié 61% des trinômes possibles, dont environ 24% grâce aux moyens du CINES. Ces derniers calculs ont été effectués essentiellement sur l'Origin 2000 en 2001, et sur l'IBM SP3 en 2002, avec un programme multi-série écrit en C utilisant la bibliothèque MPI. Fin août 2002, nous avons enfin trouvé un premier trinôme irréductible, pour $s = 3037958$.

Comment vérifier ? Une des difficultés de ce type de calcul est la vérification. Il n'est pas possible de fournir une preuve de nos calculs qui soit vérifiable à la main ; l'utilisation de l'ordinateur est nécessaire. Comme on a pu le constater avec l'article ayant motivé nos recherches, personne n'est à l'abri d'une erreur de programma-

tion, voire d'une erreur du compilateur ou du système. Pour les résultats positifs, il est relativement facile de vérifier avec un programme indépendant. C'est ce que nous avons fait pour le trinôme trouvé : nous avons calculé $x^{2^r} \bmod (x^r + x^s + 1)$ avec la bibliothèque NTL développée par Victor Shoup (en trois fois le temps de notre programme).

Quant aux résultats négatifs, comment convaincre que $x^r + x^s + 1$ n'est pas primitif ? Pour les trinômes ayant un petit facteur, nous conservons le degré de ce facteur. Cependant, ces trinômes sont les plus faciles à tester, donc le gain global n'est pas significatif. Pour les trinômes nécessitant le coûteux calcul de $x^{2^r} \bmod f$, nous conservons les coefficients de poids faible de ce polynôme, qui ne contiennent certes qu'une partie de l'information totale, mais devraient permettre de détecter la plupart des erreurs. Cependant, la vérification dans ce cas est aussi coûteuse que le calcul initial.

Conclusion. Hormis l'aspect « défi scientifique » et le nouveau record établi, nous avons beaucoup appris lors de ce projet. Le premier constat est que la puissance des ordinateurs actuels permet aujourd'hui d'effectuer des calculs impossibles — voire inimaginables — il y a quelques années seulement, grâce à la fameuse « loi de Moore » (voir ci-dessous).

La seconde leçon est que les centres de ressources comme le CINES permettent de gagner un ordre de grandeur en puissance de calcul ; selon la complexité du problème considéré, cubique comme dans notre cas, quadratique ou linéaire, on peut ainsi atteindre une taille 2 fois, 3 fois, voire 10 fois plus grande respectivement.

Le troisième constat est que, plus que la fréquence d'horloge, la capacité mémoire des ordinateurs risque de devenir le principal facteur limitant, que ce soit la mémoire cache de niveau un, deux, voire trois, la mémoire vive, ou la mémoire disque auxiliaire. Concevoir des algorithmes optimisant l'utilisation de la mémoire, aussi bien que des procédés et implantations efficaces de « ramasse-miettes » devrait devenir un enjeu majeur des années à venir.

Paul Zimmermann
INRIA Lorraine/LORIA
zimmerma@loria.fr

Générateurs pseudo-aléatoires et registres à décalage. Un générateur *aléatoire* produit un vrai aléa, totalement imprévisible — comme une suite de lancers de dé — alors qu'un générateur *pseudo-aléatoire* comme celui ci-dessus produit une suite de valeurs apparemment aléatoires, mais en fait produites par une formule. Les plus courants et les plus utilisés sont les *registres à décalage*, où un registre

de k valeurs — $k = 5$ dans l'exemple ci-dessous — est décalé d'un cran à chaque étape :

étape k	a_0	a_1	a_2	a_3	a_4
étape $k+1$	a'_0	a_0	a_1	a_2	a_3

et l'une des valeurs se calcule à partir des autres : $a'_0 = g(a_0, a_1, a_2, a_3, a_4)$. On utilise habituellement une des valeurs du registre, par exemple a_0, comme générateur pseudo-aléatoire de bits.

Trinômes primitifs. Un *trinôme* est un polynôme composé de trois monômes. Enfin, un polynôme f de degré r sur GF(2) est dit *primitif* s'il est irréductible — *i.e.* ne se décompose pas en produit de polynômes de degrés inférieurs, comme les nombres premiers pour les entiers — et si de plus les polynômes $x^i \bmod f$ pour $0 \leq i < 2^r - 1$ correspondent à tous les polynômes non nuls de degré inférieur à r sur GF(2). Un polynôme irréductible n'est pas forcément primitif, par exemple $f = x^6 + x^3 + 1$ est irréductible mais non primitif. On peut montrer que dans ce cas, il existe un diviseur non trivial d de $2^r - 1$ tel que $x^d \bmod f = 1$. Dans l'exemple ci-dessus, $d = 9$ divise en effet $2^6 - 1 = 63$.

Si l'on choisit un degré r tel que $2^r - 1$ est premier, ce qui correspond aux *nombres de Mersenne*, alors les seuls diviseurs de $2^r - 1$ sont 1 et $2^r - 1$, et tous les polynômes irréductibles de degré r sont aussi primitifs. Un tel polynôme primitif induit un générateur pseudo-aléatoire de période $2^r - 1$. Si l'on considère un générateur à décalage utilisant un registre de r bits, et une fonction g des ces r bits pour calculer le bit suivant, la période maximale est de 2^r. Les polynômes primitifs donnent donc des registres à décalage de période quasi-maximale à taille de registre fixée.

Nombres de Mersenne. Un nombre de Mersenne est un entier *premier* de la forme $2^p - 1$, c'est-à-dire divisible seulement par 1 et lui-même. Par exemple $2^2 - 1 = 3$, $2^3 - 1 - 7$, $2^5 - 1 = 31$, $2^7 - 1 = 127$ sont des nombres de Mersenne. On montre facilement que l'exposant p d'un nombre de Mersenne est forcément premier. En effet, si $p = qr$, alors $2^p - 1$ s'écrit $x^q - y^q$ avec $x = 2^r$, $y = 1$, et comme $x^q - y^q$ est divisible par $x - y$, $2^p - 1$ est divisible par $2^r - 1$. Il y a actuellement 39 nombres de Mersenne connus, dont les exposants sont 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243, 110503, 132049, 216091, 756839, 859433, 1257787, 1398269, 2976221, 3021377, 6972593, 13466917. Les nombres de la forme $n = 2^p - 1$ sont célèbres car ce sont les préférés des chasseurs au record du plus grand nombre premier connu. En effet la vérification de primalité d'un tel nombre

est facile. George Woltman a écrit un programme très efficace pour déterminer si un exposant donné p correspond à un nombre de Mersenne. Ce programme est utilisé depuis plusieurs années par des milliers d'internautes. Cette gigantesque « grappe de calcul » s'appelle GIMPS (*Great Internet Mersenne Prime Search*, `www.mersenne.org`). Quelques internautes ont ainsi eu la chance de trouver un nombre premier record de la forme $2^p - 1$, et fait la une du New York Times.

Après le début de notre recherche pour 6972593, un exposant de Mersenne encore plus grand a été trouvé par GIMPS. Cependant, un théorème dû à Swan montre que si r modulo 8 vaut 3 ou 5, ce qui est le cas pour $r = 13466917$, il n'existe pas de trinôme primitif de la forme $x^r + x^s + 1$, sauf pour $s = 2$ ou $r - 2$, ce qui est facile à vérifier.

La loi de Moore. Selon la « loi de Moore », la puissance des ordinateurs augmente d'un facteur 2 tous les 18 mois. Ainsi pour les très longs calculs, il faut en fait considérer qu'on dispose d'un ordinateur dont la fréquence augmente de façon continue suivant la loi de Moore. Voici deux conséquences étonnantes : (i) Combien de temps faut-il sur un tel ordinateur pour reproduire un calcul commencé depuis la nuit des temps ? La réponse est 1.5 an seulement ! (ii) Supposons qu'on dispose de 10^{10} ordinateurs (de puissance actuelle) qui effectuent un calcul en parallèle durant un an ; combien de temps faudra-t-il sur un seul ordinateur suivant la loi de Moore pour effectuer le même calcul ? La réponse, encore plus surprenante, est d'un peu plus de 48 ans seulement ! Le tableau ci-dessous indique pour n années sur une machine actuelle (col. centrale) le temps équivalent sur une machine de puissance suivant la loi de Moore, pour un calcul se finissant (col. de gauche) ou commençant (col. de droite) aujourd'hui.

$-t \to 0$	n	$0 \to t$
∞	2.16	1.5
impossible	10	3.7
impossible	10^2	8.3
impossible	10^5	23.2
impossible	10^{10}	48.2

La colonne centrale représente aussi le nombre de processeurs nécessaires pour effectuer le calcul correspondant en une année ; la comparaison avec celle de droite peut susciter des questions quant à l'utilité du parallélisme...