

SÉMINAIRE DELANGE-PISOT-POITOU. THÉORIE DES NOMBRES

MAURICE MIGNOTTE

Quelques problèmes d'informatique

Séminaire Delange-Pisot-Poitou. Théorie des nombres, tome 17, n° 2 (1975-1976),
exp. n° 30, p. 1-6

http://www.numdam.org/item?id=SDPP_1975-1976__17_2_A6_0

© Séminaire Delange-Pisot-Poitou. Théorie des nombres
(Secrétariat mathématique, Paris), 1975-1976, tous droits réservés.

L'accès aux archives de la collection « Séminaire Delange-Pisot-Poitou. Théorie des nombres » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
<http://www.numdam.org/>

QUELQUES PROBLÈMES D'INFORMATIQUE

par Maurice MIGNOTTE

Nous abordons trois questions d'algorithmique qui peuvent intéresser des mathématiciens "purs" et, en particulier, des arithméticiens: le calcul d'une puissance entière, la multiplication de deux matrices carrées, la multiplication de polynômes et d'entiers naturels.

I - Calcul de x^n

1. Méthodes.

La méthode banale nécessite $n - 1$ multiplications. Un algorithme plus efficace (pour n "grand") consiste à utiliser la formule

$$x^n = \prod_{1 \leq i \leq k, e_i \neq 0} x^{2^i}, \text{ où } n = \sum_{i=1}^k e_i 2^i, \quad e_i \in \{0, 1\},$$

est la décomposition binaire de n ; donc $[2 \log n]$ multiplications suffisent (on utilise ici la notation $\log x$ pour le logarithme binaire, $\log x = \text{Log } x(\text{Log } 2)$).

2. Applications.

(a) Calcul du n -ième terme d'une suite récurrente linéaire. - Soit une suite récurrente linéaire définie par ses h premiers termes et la condition

$$u_n = a_1 u_{n-1} + \dots + a_h u_{n-h}, \text{ pour } n \geq h.$$

Cette condition peut s'écrire, en posant

$$U_k = \begin{pmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+h-1} \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 1 & 0 & \dots & \dots & \dots \\ 0 & 0 & 1 & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & 0 & 1 \\ a_h & \dots & \dots & \dots & a_2 & a_1 \end{pmatrix}$$

$$U_n = A U_{n-1} \text{ pour } n \geq 1.$$

D'où la formule

$$U_n = A^n U_0 \text{ pour } n \geq 0.$$

L'étude précédente montre donc que le calcul de u_n nécessite au plus $O(\log n)$ opérations.

(b) Fermeture réflexive - transitive d'une relation. - Soit $R \subset E \times E$, où E est un ensemble fini de cardinal n , une relation binaire que l'on représentera par une matrice booléenne encore notée R . La fermeture réflexive de R est définie par la matrice

$$R^* = I + R + R^2 + \dots + R^m + \dots$$

(toutes les opérations sont booléennes, en particulier $X + X = X$), en fait $R^{n+k} = R^n$ pour $k \geq 0$ et donc $R^* = I + R + \dots + R^n = (I + R)^n$.

Le calcul de R^* est donc possible en effectuant au plus $[2 \log n]$ produits de matrices $n \times n$.

(c) Factorisation des polynômes sur un corps fini. - Pour simplifier, nous ne considérerons que le cas où le corps fini F est le corps premier \mathbb{F}_p . La méthode est due à BERLEKAMP (voir [1]). Soit f un polynôme à coefficients dans F , de degré d , sans facteur carré. On considère l'équation

$$(1) \quad g^p \equiv g \pmod{f};$$

c'est une équation linéaire. Si g est une des solutions, alors

$$(2) \quad f(X) = \prod_{s \in F} (f(X) - g(X) - s),$$

de plus, si g parcourt l'espace des solutions, cette méthode conduit à une factorisation complète de f . L'équation (1) se met sous la forme

$$(Q - I)g \equiv 0 \pmod{f},$$

où Q est une certaine matrice, calculable en $O(d^2 \log p)$ opérations (dans F). La partie la plus coûteuse de l'algorithme est de déterminer les valeurs de s telles que la factorisation (2) ne soit pas triviale. Ces valeurs de s sont les solutions d'un certain polynôme qui se calcule assez facilement. On aboutit au problème suivant.

PROBLÈME 1. - Soit P un polynôme de degré d sur \mathbb{F}_p , qui se factorise complètement dans \mathbb{F}_p . Combien faut-il d'opérations élémentaires pour déterminer une racine de P ? (Par opération élémentaire, il faut entendre addition, multiplication ou division dans \mathbb{F}_p .)

Pour $d = 1$, le problème est trivial : une opération suffit.

Pour $d = 2$, la formule traditionnelle exige le calcul d'une racine-carré (d'un carré). On doit résoudre l'équation, en x , $a = x^2$. Or, si 4 divise $p - 1$,

$$a^{(p+1)/4} = x^{(p+1)/2} = x^{(p-1)/2} x = \pm x,$$

donc $O(\log p)$ opérations suffisent. Le cas $p \equiv 1$ modulo 4 a été traité par LEHMER [3], qui obtient le même résultat (il utilise certaines suites récurrentes linéaires).

Pour les degrés plus grands, je ne connais pas de meilleure borne que $O(p)$ opérations dans le cas d'un polynôme générique.

3. La fonction $l(n)$.

La quantité $l(n)$ est, par définition, le nombre minimal de multiplications nécessaires pour effectuer le calcul de x^n .

PROBLÈME 2. - Etudier la fonction $l(n)$.

Cette question est traitée en détail dans l'ouvrage de KNUTH [2], il signale de nombreux problèmes ouverts concernant le comportement de la fonction $l(n)$.

II . Multiplication de matrices

1. Méthodes.

Soit à calculer $C = AB$, où A et B sont deux matrices $n \times n$. La méthode classique nécessite $O(n^3)$ opérations élémentaires (additions et multiplications). La méthode de STRASSMAN, exposée en [2], repose sur des formules permettant d'effectuer le produit de deux matrices 2×2 qui ne comportent que 7 multiplications et sont valables pour des variables non commutatives. Elle conduit à un nombre d'opérations en $O(n^{\log 7})$ (on a $\log 7 \approx 2,81$) .

2. Applications.

(a) Evaluation de n polynômes en n points. - Pour obtenir le résultat, il suffit d'effectuer le produit de deux matrices $n \times n$ convenables, donc $O(n^{\log 7})$ opérations suffisent.

(b) Produit de matrices booléennes. - Posons :

$M(n)$ = nombre minimal d'opérations (additions et multiplications) nécessaires pour effectuer le produit de deux matrices $n \times n$ quelconques,

$B(n)$ = nombre minimal d'opérations nécessaires pour calculer le produit booléen de deux matrices $n \times n$ quelconques,

$C(n)$ = nombre minimal d'opérations nécessaires pour calculer la fermeture transitive réflexive d'une matrice booléenne $n \times n$ quelconque.

Notons d'abord la relation $B(n) \leq C(3n)$ qui résulte de la formule

$$A^* = \begin{pmatrix} I & B & BC \\ 0 & I & C \\ 0 & 0 & I \end{pmatrix} \quad \text{si } A = \begin{pmatrix} 0 & B & 0 \\ 0 & 0 & C \\ 0 & 0 & 0 \end{pmatrix}.$$

En sens inverse, le paragraphe I.2(b) donne

$$C(n) \leq 2 B(n) \log n .$$

On démontre même que l'hypothèse suivante, il existe $\lambda > 2$ tel que l'on ait $B(2n) \geq \lambda B(n)$ pour tout n , implique la majoration $C(n) \leq k B(n)$, où k est une certaine constante ($k = k(\lambda)$, voir [4]).

Pour calculer le produit booléen de deux matrices booléennes, on peut procéder ainsi. Soit $\mathcal{B} = \{\emptyset, \mathbb{K}\}$ l'ensemble de Boole, et soient $\varphi : \mathcal{B} \rightarrow \mathbb{N}$ et $\psi : \mathbb{N} \rightarrow \mathcal{B}$ les fonctions définies par

$$\begin{aligned} \varphi(\emptyset) &= 0, & \varphi(\mathbb{K}) &= 1, \\ \psi(0) &= \emptyset, & \psi(x) &= \mathbb{K} \text{ si } x \geq 1. \end{aligned}$$

On calcule le produit booléen grâce à la formule

$$A \cdot B = \psi(\varphi(A) \varphi(B)).$$

D'où la majoration $B(n) \leq M(n)$.

(c) Calcul de l'inverse d'une matrice.

Les résultats sont analogues. Pour éviter tout ennui, on se limite aux matrices $n \times n$ complexes dont les coefficients engendrent un corps de degré de transcendance n^2 sur les rationnels. Soit $I(n)$ le nombre minimal d'opérations permettant de calculer l'inverse d'une matrice qui est de ce type.

On a

$$I(n) \leq M(n) \log n \quad (\text{voir [4]})$$

et

$$M(n) \leq I(3n).$$

La seconde de ces inégalités résulte de la formule

$$\begin{pmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}^{-1} = \begin{pmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{pmatrix}$$

3. Problème.

PROBLÈME 3. - Etudier $M(n)$.

On a bien sûr $M(n) \geq n^2$.

III. Multiplication de polynômes et d'entiers.

1. Multiplication de polynômes.

Le calcul du produit de deux polynômes de degré n par la méthode scolaire nécessite plus de n^2 opérations. Il existe une méthode plus efficace, la transformée de Fourier rapide (en anglais F. F. T.). Le principe en est le suivant. Si x_0, \dots, x_{2n} sont des points fixés, pour calculer $W(X) = U(X) V(X)$ on procède ainsi

1° calculer $U(x_i)$ et $V(x_i)$ pour $i = 0, \dots, 2n$;

2° évaluer $W(x_i) = U(x_i) V(x_i)$, $i = 0, \dots, 2n$;

3° interpoler le polynôme W .

La seconde étape ne nécessite que $O(n)$ opérations. Nous allons voir qu'avec un choix judicieux des x_i , les étapes 1° et 3° ne coûtent que $O(n \log n)$ opérations. On prend pour points d'interpolation les $x_i = \omega^i$, où ω est une racine primitive 2^r -ième de l'unité ($2^r > 2n$). Si $x = \omega^k$ et $y = x^2$, pour tout polynôme Q de degré $< 2^r$, on a

$$Q(x) = Q_1(y) + xQ_2(y),$$

où les polynômes Q_1 et Q_2 ont un degré plus petit que 2^{r-1} . Si $T(m)$ désigne le nombre d'opérations nécessaires pour effectuer le calcul 1° dans le cas de m points d'interpolation, on a donc la relation

$$T(2^r) = 2 T(2^{r-1}) + 3 \cdot 2^{r-1} \quad (\text{noter que } \omega^{i+2^{r-1}} = -\omega^i).$$

D'où la majoration $T(n) = O(n \log n)$.

Pour interpoler, il suffit de noter que la transformation 1° est définie par la matrice

$$\Omega = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^s \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2s} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \omega^s & \omega^{2s} & \dots & \omega^{s^2} \end{pmatrix}$$

tandis que 3° est définie par Ω^{-1} , où $\Omega^{-1} = (1/(n+1)) \bar{\Omega}$.

Le coût de la troisième étape est donc aussi de $O(n \log n)$ opérations.

2. Multiplication d'entiers.

Pour multiplier deux entiers de n chiffres, la méthode classique prend un temps en $O(n^2)$. En fait, il n'est pas difficile d'améliorer cette estimation.

Soient en effet deux nombres u et v comportant $2n$ chiffres binaires

$$u = 2^n u_1 + u_0, \quad v = 2^n v_1 + v_0.$$

La formule

$$uv = (2^{2n} + 2^n)u_1 v_1 + 2^n(u_1 - u_0)(v_0 - v_1) + (2^n + 1)u_0 v_0$$

montre que le coût $T(m)$ du produit de deux entiers binaires de m chiffres vérifie la condition

$$T(2n) \leq 3T(n) + cn;$$

ce qui implique la majoration $T(n) \leq 3cn^{\log_3}$.

Plus généralement, on peut couper les entiers à multiplier en r morceaux, ce qui conduit à un coût en $O(m^{1+(1/\log r)})$. Ainsi $T(n)$ est un $O(n^{1+\epsilon})$ pour tout ϵ positif.

Une idée plus efficace consiste à s'inspirer de la F. F. T. Si

$$u = \sum u_i 2^{Li}, \quad v = \sum v_i 2^{Li}, \quad w = uv.$$

On considère les polynômes

$$U(X) = \sum u_i X^i, \quad V(X) = \sum v_i X^i, \quad W(X) = U(X) V(X).$$

Alors $w = W(2^L)$.

L'existence possible de retenues pose un sérieux problème. Ce problème a été résolu par STRASSEN et SCHÖNHAGE. Ils obtiennent un coût en $O(n \log n \log \log n)$.

BIBLIOGRAPHIE

- [1] BERLEKAMP (E. R.). - Factoring polynomials over large finite fields, Math. of Computation, t. 24, 1970, p. 713-735.
- [2] KNUTH (D.E.). - The art of computer programming, vol. II : Semi-numerical algorithms. - Reading, Addison-Wesley publishing Company, 1969.
- [3] LEHMER (D. H.). - Computer technology applied to the theory of numbers, "Studies in number theory" ; p. 117-151, - Englewood Cliffs, Prentice Hall, 1969.
- [4] MUNRO (I.). - Problems related to matrix multiplication, "Computer complexity courant computer science symposium 7". Edited by Randal Rustin. - New-York, Algorithmics Press, 1973.

(Texte reçu le 11 octobre 1976)

Maurice MIGNOTTE
 Centre de calcul
 Université Louis Pasteur
 7 rue René Descartes
 67084 STRASBOURG CEDEX
