Travaux Dirigés : Complexité et algorithmes arithmétiques usuels

1 Algorithme de multiplication de Karatsuba

Soient a et b deux entiers, dont les écritures en base N sont

$$a = \sum_{i=0}^{m} a_i \, N^i \quad \text{et} \quad b = \sum_{i=0}^{n} b_i \, N^i$$

On suppose que l'addition et la multiplication de deux entiers dans $[\![0, N-1]\!]$ se font en temps $O(1)$.

On suppose $n \geq m$. Soit $d = \lceil (n+1)/2 \rceil$.

1. Montrer qu'il existe des entiers $A_0, A_1, B_0, B_1 \in [\![0, N^d - 1]\!]$ tels que $a = A_0 + A_1 \, N^d$ et $b = B_0 + B_1 \, N^d$. Expliquer pourquoi leur calcul ne demande pas d'opération arithmétique.
2. On pose $C = (A_0 + A_1)(B_0 + B_1)$. Justifiez que

$$ab = A_0 B_0 + (C - A_0 B_0 - A_1 B_1) \; N^d + A_1 B_1 \, N^{2d}$$

   En déduire que le calcul de ab peut se faire avec trois produits d'entiers de taille au plus d ainsi qu'un certain nombre d'additions/soustractions, de coût en $O(d)$.

En utilisant la même méthode pour chacune des multiplications intermédiaires, on obtient l'algorithme de multiplication de Karatsuba, algorithme récursif de type "diviser pour régner". On note $K(d)$ la complexité (dans le pire des cas) de la multiplication de deux entiers de taille au plus d.

3. Justifier la relation de récurrence :

$$K(d) = 3K\left(\lceil d/2 \rceil\right) + O(d)$$

   Dans la suite, on note $c > 0$ une constante telle que pour tout $d \in \mathbb{N}^*$ on ait la majoration $K(d) \leq 3K\left(\lceil d/2 \rceil\right) + c\,d$. On pose aussi $K(1) = 1$.
4. Déterminer une expression du terme général de la suite (u_k) définie par :

$$\begin{cases} u_0 = 1 \\ \forall k \in \mathbb{N}^*, \; u_k = 3u_{k-1} + c\,2^k \end{cases}$$

   Que peut-on en déduire sur $K(2^k)$?
5. En admettant que la fonction K soit croissante, montrer que $K(d) \leq (2c+1)3^{\lceil \log_2(d) \rceil} - c\,2^{\lceil \log_2(d) \rceil + 1}$ pour tout $d \in \mathbb{N}^*$.
6. Montrer finalement que $K(d) = O(d^{\log_2(3)}) = O(d^{1,58\ldots})$.

2 Autour d'Euclide étendu

I. Dans la version de l'algorithme d'Euclide étendu vue en cours, en partant des deux entiers a et b, on calcule les suites (r_i), (u_i) et (v_i) vérifiant :

- $r_0 = a,\ u_0 = 1,\ v_0 = 0$
- $r_1 = b,\ u_1 = 0,\ v_1 = 1$
- si $r_i \neq 0$, $\begin{cases} r_{i+1} = r_{i-1} - q_i r_i \\ u_{i+1} = u_{i-1} - q_i u_i \\ v_{i+1} = v_{i-1} - q_i v_i \end{cases}$ où q_i est le quotient de la division de r_{i-1} par r_i.

1. On suppose $a > b > 0$. Montrer que la suite $((-1)^{i+1} v_i)$ est croissante, de même que la suite $((-1)^i u_i)$ à partir du rang 1.
2. Justifier que $au_i + bv_i = r_i$ pour tout indice i où ces éléments sont définis.
3. Démontrer de même la relation $\begin{vmatrix} u_i & v_i \\ u_{i+1} & v_{i+1} \end{vmatrix} = (-1)^i$.

   Que peut-on en déduire sur les coefficients de Bézout calculés par l'algorithme ?

On note n le dernier indice pour lequel $r_n \neq 0$. On a donc $r_n = a \wedge b$, et $r_{n+1} = 0$.

4. Que vaut $au_{n+1} + bv_{n+1}$?
5. En calculant $\begin{vmatrix} u_n & v_n \\ bu_{n+1} & bv_{n+1} \end{vmatrix}$ de deux manières, montrer que $u_{n+1} = \pm \dfrac{b}{a \wedge b}$ et donner la valeur de $|v_{n+1}|$.
6. En déduire une majoration des termes des suites (u_i) et (v_i) calculées par l'algorithme.

II. Soient a, b deux entiers (avec $b \neq 0$) ; on note q et r le quotient et le reste de la division de a par b. Soit $d = a \wedge b$.

1. On suppose connus des coefficients de Bézout s, t pour le couple (b, r), c'est-à-dire que $bs + rt = d$. Déterminer une relation de Bézout pour le couple (a, b).
2. En déduire un algorithme *récursif* (Euclide étendu "de bas en haut") calculant le pgcd et des coefficients de Bézout de deux entiers.
3. On suppose encore a, b strictement positifs. Démontrer par récurrence que les coefficients de Bézout (s, t) renvoyés par l'algorithme vérifient $|s| \leq b/d$ et $|t| \leq a/d$.

3 D'autres itérations de Newton : calcul de racines carrées modulaires

Le but de cet exercice est de résoudre rapidement l'équation

$$(C)\ :\ x^2 = a\ [p^n]$$

où p est un nombre premier impair, $n \in \mathbb{N}^*$, et a un entier premier à p.

Si cette équation possède des solutions, on dit que a est un *carré* ou un *résidu quadratique* (inversible) modulo p^n. Une solution de l'équation (C) est naturellement appelée *racine carrée* de a modulo p^n.

1. En considérant l'application $x \mapsto x^2$ de $(\mathbb{Z}/p^n\mathbb{Z})^\times$ dans lui-même, déterminer le nombre de carrés dans $(\mathbb{Z}/p^n\mathbb{Z})^\times$.
2. Cas $n = 1$.
   (a) On suppose $p = 3$ [4]. Montrer que si a est un carré inversible modulo p, alors $a^{(p+1)/4}$ est une racine carrée de a.
   (b) Écrire un programme qui calcule une racine carrée de a modulo p, en utilisant la formule ci-dessus ou une recherche exhaustive suivant les cas.
   Quelle est sa complexité ?
3. Démontrer l'implication : a est un carré modulo $p^n \implies a$ est un carré modulo p.

L'implication réciproque est vraie, et la démonstration consiste en des constructions effectives par itération de Newton.

Premier algorithme

Soit a un résidu quadratique inversible modulo p, et x_0 une racine carrée de a modulo p. On définit la suite (x_k) par la relation suivante :

$$\forall k \in \mathbb{N},\ x_{k+1} = \frac{x_k^2 + a}{2x_k}\ [p^{2^{k+1}}]$$

où la division est en fait la multiplication par l'inverse modulaire de $2x_k$.

4. Démontrer par récurrence que pour tout $k \in \mathbb{N}$, l'inverse modulaire de $2x_k$ est bien défini et que $x_k^2 = a\ [p^{2^k}]$.
5. Écrire un programme qui résout l'équation $(C) : x^2 = a\ [p^n]$. Quelle est sa complexité ?
   Le tester avec les valeurs qui suivent :
   • $a = 2021$ et $p^n = 5^5$ • $a = 10^{81} + 2$ et $p^n = 11^{80}$ • $a = 7$ et $p^n = 3^{1024}$
6. Expliquer pourquoi on qualifie cet algorithme d'itération de Newton.

Deuxième algorithme : racine carrée inverse

Soit a un résidu quadratique inversible modulo p, et u_0 une racine carrée de l'**inverse** a^{-1} de a modulo p. On définit la suite (u_k) par la relation suivante :

$$\forall k \in \mathbb{N},\ u_{k+1} = \frac{u_k(3 - au_k^2)}{2}\ [p^{2^{k+1}}].$$

7. Démontrer par récurrence que pour tout k on a l'égalité $au_k^2 = 1\ [p^{2^k}]$.
8. Comment peut-on calculer une division par 2 modulo $p^{2^{k+1}}$ sans passer par une inversion modulaire ? (Indication : parmi x et $x + p^{2^{k+1}}$, un des deux est pair...)
9. En utilisant la relation $\sqrt{a} = a.\sqrt{a^{-1}}$, écrire un deuxième programme qui résout l'équation $(C) : x^2 = a\ [p^n]$. Quelle est sa complexité ?
   Le tester avec les mêmes valeurs que ci-dessus.
10. Expliquer pourquoi cet algorithme correspond à une itération de Newton pour la fonction $f : x \mapsto x^{-2} - a$.
11. Comparer les performances des deux programmes. Quel est l'intérêt de la deuxième méthode ?

Pour aller plus loin

12. Par un argument de dénombrement, prouver directement l'implication :
$$a \text{ est un carré inversible modulo } p \implies a \text{ est un carré inversible modulo } p^n.$$
13. Est-ce que p est un carré modulo p^2 ?
Plus généralement, si $a = p^l b$ avec $l \in \mathbb{N}$ et $p \nmid b$, donner une condition nécessaire et suffisante pour que a soit un carré modulo p^n.
14. Cas $p = 2$
    (a) Déterminer les carrés de $(\mathbb{Z}/2\mathbb{Z})^\times$, de $(\mathbb{Z}/4\mathbb{Z})^\times$, et de $(\mathbb{Z}/8\mathbb{Z})^\times$.
    (b) Soit a un entier congru à 1 modulo 8. Pour $u_0 \in (\mathbb{Z}/8\mathbb{Z})^\times$ quelconque, on définit la suite (u_k) par une relation similaire à celle d'avant :
    $$\forall k \in \mathbb{N},\ u_{k+1} = \frac{u_k(3 - au_k^2)}{2} \quad [2^{2^{k+1}+2}].$$
    Démontrer que $au_k^2 = 1\ [2^{2^k+2}]$ pour tout $k \in \mathbb{N}$, et en particulier que $3 - au_k^2$ est un entier pair (ce qui justifie la division par 2).
    (c) Démontrer l'équivalence : pour tout $n \geq 3$ et tout a impair,
    $$a \text{ est un carré modulo } 8 \iff a \text{ est un carré modulo } 2^n.$$
    (d) Écrire un programme qui calcule des racines carrées modulo 2^n et estimer sa complexité. Quelles sont les racines carrées de 17 modulo 1024 ? Modulo 2^{2050} ?