L'algorithme binaire récursif de calcul de pgcd.

Damien STEHLÉ

Grenoble, 01-04-2004

Travail en commun avec Paul Zimmermann

http://www.loria.fr/~stehle/

Arithmétique (naïve) des entiers

- $N \in [|0, 2^n - 1|]$ représenté en binaire, avec au plus n bits.
- Mesure de complexité : op. élémentaires sur les bits.

- Comparaison, addition, soustraction : $O(n)$.
- Multiplication : $O(n^2)$.
- Division d'un nombre de taille m par un nombre de taille $n < m$:

$$O(m \cdot (m - n)).$$

Arithmétique (rapide) des entiers

- Comparaison, addition, soustraction : c'est déjà linéaire !
- Multiplication à base de transformée de Fourier rapide : $M(n) = O(n \log n \log\log n)$, quasi-linéaire.
- Division par itération de Newton et FFT (cf. ce matin) : division $(2n, n)$ en $O(M(n))$, quasi-linéaire.

- Conclusion : arithmétique des entiers en temps quasi-linéaire.

Le pgcd, à quoi ça sert ?

– Arithmétique des nombres rationnels.
– Inversion modulaire (RSA, ...).
– Développement en fractions continues.
– Factorisation des entiers (méthode $p-1$, ECM, ...).

Petite histoire du pgcd

– ≈ -300, Euclide : algo. d'Euclide, en temps quadratique $O(n^2)$.
– 1938, Lehmer : accélération de l'algorithme d'Euclide (les bits forts suffisent pour calculer les premiers quotients).
– 1970, Schönhage et Strassen : multiplication rapide en temps

$$M(n) = O(n \log n \log \log n).$$

– 1970, Knuth : algorithme de Lehmer récursif, basé sur la multiplication rapide de Schönhage-Strassen.
Complexité quasi-linéaire $O(M(n) \log^4 n)$.
– 1971, Schönhage : analyse précise de l'algorithme de Knuth.
Complexité quasi-linéaire $O(M(n) \log n)$.

La division Euclidienne "standard"

– Étant donnés $a, b > 0$, il existe un unique couple (q, r) t.q. :

$$a = bq + r, \;\; 0 \leq r < b, \;\; q \geq 0.$$

– Exemple : $a = 157 = 10011101$, $b = 59 = 111011$.
– $(q, r) = (2, 39) = (10, 100111)$.
– Remarque : $10011 = 19$, $111 = 7$ donnent le même quotient.

L'algorithme d'Euclide "standard"

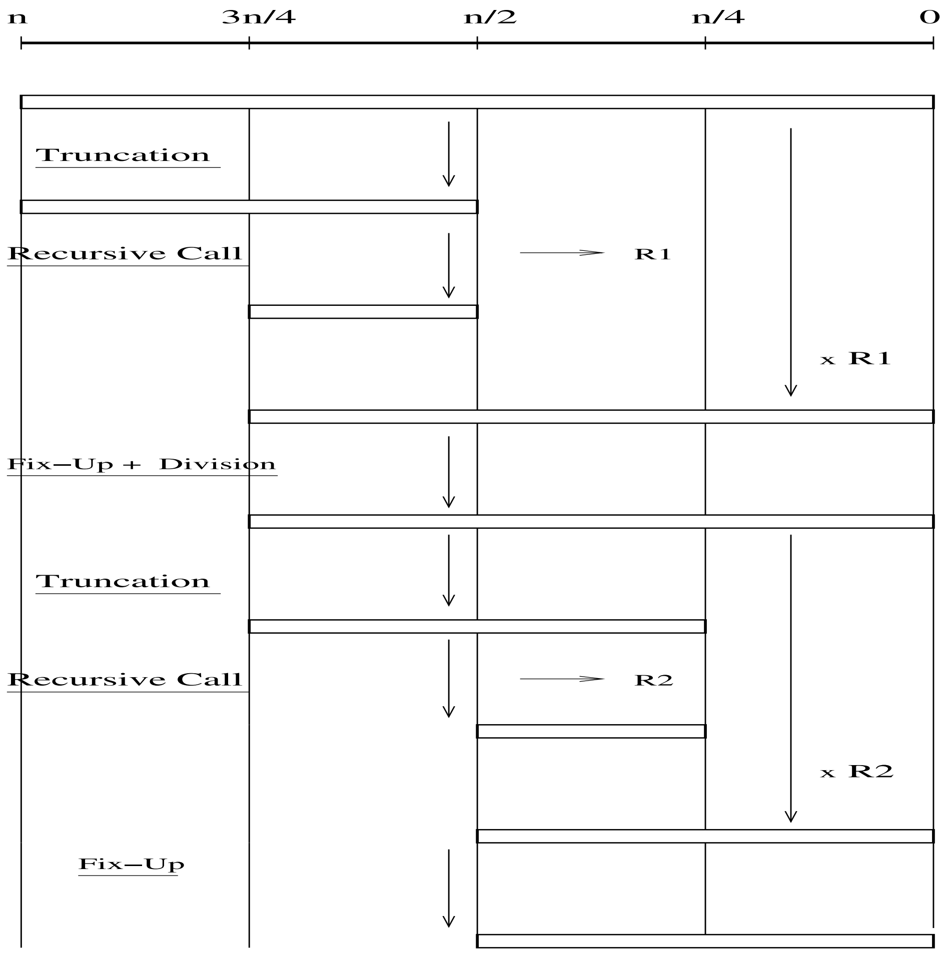
- $$\begin{pmatrix} r_0 = a \\ r_1 = b \end{pmatrix} \longrightarrow^{q_1} \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \longrightarrow^{q_2} \ldots \longrightarrow \begin{pmatrix} r_t = gcd(a,b) \\ r_{t+1} = 0 \end{pmatrix}.$$
- Complexité : "en général", $t = O(n)$, quotients de taille $O(1)$ et restes de taille $O(n)$.
  $\Longrightarrow O(n)$ multiplications $(1, n)$, donc : $O(n^2)$.
- Pour faire mieux, il faut rééquilibrer les multiplications.

L'algorithme de Knuth (1)

– Deux remarques préliminaires :
  1) Stocker les restes : $O(n^2)$, stocker les quotients : $O(n)$.
  2) Les k bits les plus significatifs permettent de calculer les $\approx k/2$ premiers bits de la suite des quotients.
– Algorithme de Knuth : utilisation récursive de 2) pour calculer tous les quotients et très peu de restes.
– PROBLÈME : 2) est faux si on enlève $\approx$.

– HalfGcd renvoie deux restes successifs de taille moitié, et la matrice de transformation. HalfGcd s'appelle récursivement.
– FastGcd utilise $O(\log n)$ fois HalfGcd pour calculer le pgcd.

L'algorithme de Knuth (2)



La division binaire

- C'est le "miroir" de la division Euclidienne.
- On ajoute à a des shiftés gche de b pour supprimer les bits forts
  $\rightarrow$ on ajoute à a des shiftés dte de b pour supprimer les bits faibles.

- Si $a \neq 0$, $\nu_2(a)$ est le plus grand k t.q. $2^k \mid a$. Et $\nu_2(0) = \infty$.
- Pour a, b avec $\nu_2(a) < \nu_2(b)$, il existe un unique couple (q, r) t.q. :

$$a = \frac{b}{2^{\nu_2(b)-\nu_2(a)}} q + r, \quad \nu_2(r) > \nu_2(b), \quad |q| < 2^{\nu_2(b)-\nu_2(a)}.$$

- En fait,

$$q = \frac{a}{2^{\nu(a)}} \left(\frac{b}{2^{\nu(b)}} \right)^{-1} \text{ cmod } 2^{\nu(b)-\nu(a)+1}.$$

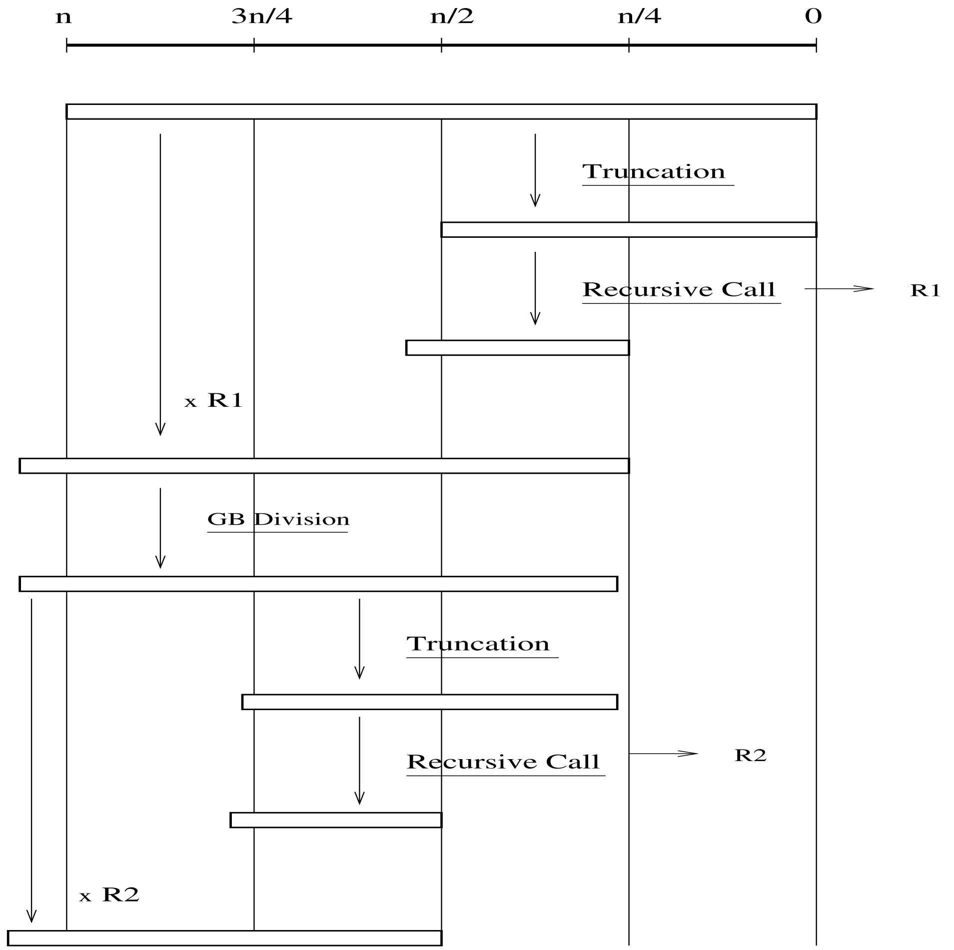
L'algorithme d'Euclide binaire

	$\xrightarrow{a+3b/4}$		$\xrightarrow{a+3b/8}$		$\xrightarrow{a+b/4}$		$\xrightarrow{a-23b/32}$	
a	$1100010010\boxed{0011}$		$1001110\boxed{1111}00$		$100111\boxed{011}00000$		$1\boxed{000101}0000000$	11000000000000
b	$10011101\boxed{111}00$		$10011\boxed{1011}00000$		$1000\boxed{101}0000000$		$\boxed{11}000000000000$	0

pgcd=3

$$\begin{bmatrix} 3.2^{12} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -23/32 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1/4 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 3/8 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 3/4 \end{bmatrix} . \begin{bmatrix} a \\ b \end{bmatrix}$$

L'algorithme récursif binaire



Analyse de l'algorithme

– Correction : la moitié faible des bits permet de simuler le premier quart de l'algorithme d'Euclide binaire.
– Complexité : $H_n = 2H_{\frac{n}{2}+1} + O(M(n))$.
$F_n = H_n + F_{\alpha n} + O(M(n))$, avec $\alpha = \frac{1}{2}(1 + \log \frac{1+\sqrt{17}}{4}) < 1$.
– D'où : $F_n, H_n = O(M(n)\log n))$.

– Meilleure estimation heuristique du $O(.)$: 17/4.
– Gain par rapport à l'algo. de Knuth : les quotients calculés récursivement sont garantis corrects, pas besoin de "réparer la séquence des quotients".

Benchmarks

"F_n" : Test avec les Fibonacci consécutifs F_{n-1} et F_n.
"G_n" : Idem pour un pire cas de la division binaire.

type, n	Magma V2.10-12	Mathematica 5.0	Fast-GB-gcd (GNU MP)
F_n, 10^6	2.89	2.11	0.70
F_n, $2 \cdot 10^6$	7.74	5.46	1.91
F_n, $5 \cdot 10^6$	23.3	17.53	6.74
F_n, 10^7	59.1	43.59	17.34
G_n, $5 \cdot 10^5$	2.78	2.06	0.71
G_n, 10^6	7.99	5.30	1.94

Conclusion

– Idée : mêmes algos avec gauche $\leftrightarrow$ droite.

– Résultats :

1. Définition d'une division par les bits faibles ayant la propriété de Lehmer (pour les bits faibles).
2. Transposition de l'algo. de Knuth pour cette division.
3. Même complexité asymptotique mais algo. bcp plus simple.