The Euclidean Algorithm

Klaus Pommerening
Fachbereich Mathematik
der Johannes-Gutenberg-Universität
Saarstraße 21
D-55099 Mainz

January 16, 2000—english version November 30, 2011—last change February 21, 2016

1 The Algorithm

Euclid's algorithm gives the greatest common divisor (gcd) of two integers,

$$\gcd(a,b) = \max\{d \in \mathbb{Z} \mid d|a, d|b\}$$

If for simplicity we define $\gcd(0,0) = 0$, we have a function

$$\gcd : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{N}$$

with the following properties:

Lemma 1 *For any $a, b, c, q \in \mathbb{Z}$ we have:*

(i) $\gcd(a,b) = \gcd(b,a)$.

(ii) $\gcd(a,-b) = \gcd(a,b)$.

(iii) $\gcd(a,0) = |a|$.

(iv) $\gcd(a-qb,b) = \gcd(a,b)$.

Proof. Trivial; for (iv) use the equivalence $d|a, b \Longleftrightarrow d|a-qb, b$. $\diamond$

One usually writes Euclid's algorithm as a sequence of divisions with remainder:

$$r_0 = |a|, r_1 = |b|, \ldots, r_{i-1} = q_i r_i + r_{i+1},$$

where q_i is the integer quotient and r_{i+1} is the unique division remainder with $0 \le r_{i+1} < r_i$. As soon as $r_n \ne 0$ and $r_{n+1} = 0$, we have $r_n = \gcd(a,b)$. For from Lemma 1 we get

$$\gcd(a,b) = \gcd(r_0,r_1) = \gcd(r_1,r_2) = \ldots = \gcd(r_n,0) = r_n.$$

Since moreover

$$r_1 > r_2 > \ldots > r_i \geq 0 \quad \text{for all } i,$$

we reach the terminating condition $r_{n+1} = 0$ after at most $n \leq |b|$ iteration steps (i. e. divisions).

A small additional consideration even gives more. Note that each r_i is an integer linear combination of the two preceeding division remainders, hence of $|a|$ and $|b|$:

$$r_{i+1} \in \mathbb{Z}r_i + \mathbb{Z}r_{i-1} \subseteq \ldots \subseteq \mathbb{Z}r_1 + \mathbb{Z}r_0 = \mathbb{Z}a + \mathbb{Z}b;$$

for r_0 and r_1 this is immediate, and in the general case it follows by induction: Let $r_j = |a|x_j + |b|y_j$ for $0 \leq j \leq i$. Then

$$\begin{aligned} r_{i+1} = r_{i-1} - q_i r_i &= |a|x_{i-1} + |b|y_{i-1} - q_i|a|x_i - q_i|b|y_i \\ &= |a|(x_{i-1} - q_i x_i) + |b|(y_{i-1} - q_i y_i). \end{aligned}$$

This consideration even gives an explicit construction for the coefficients; for they sastisfy the recursive formulas

$$x_{i+1} = x_{i-1} - q_i x_i \quad \text{with} \quad x_0 = 1,\ x_1 = 0,$$

$$y_{i+1} = y_{i-1} - q_i y_i \quad \text{with} \quad y_0 = 0,\ y_1 = 1,$$

that agree with the formula for the r_i except for the start values:

$$r_{i+1} = r_{i-1} - q_i r_i \quad \text{with} \quad r_0 = |a|,\ r_1 = |b|.$$

The **extended Euclidean algorithm** (sometimes called algorithm of LAGRANGE) is the synopsis of these three recursive formulas. In summary we have shown (if we properly adjust the signs of x_n and y_n):

Proposition 1 *The extended Euclidean algorithm gives the greatest common divisor d of two integers a and b and integer coefficients x and y with $ax + by = d$ in finitely many steps.*

Remarks

1. The least common multiple is efficiently calculated by the formula

$$\operatorname{lcm}(a, b) = \frac{ab}{\gcd(a, b)}\,.$$

2. One calculates the greatest common divisor of several inegers by the formula

$$\gcd(\ldots(\gcd(\gcd(a_1, a_2), a_3)\ldots, a_r);$$

this allows for a bit of optimisation. An analogous statement holds for the least common multiple.

2 Analysis of EUCLID's Algorithm

The algorithm of the last section has a hidden problem: Though the quotients and division remainders are safely bounded by the input parameters, the coefficients x_i and y_i are uncontrolled at first sight. How can we guarantee that we don't get an overflow, if we use the usual integer arithmetic with bounded precision? Now, the following reasoning controls the growth:

Lemma 2 *For the coefficients x_i and y_i in the extended Euclidean algorithm we have:*

(i) $x_i > 0$, *if i is even,* $x_i \leq 0$, *if i is odd, and* $|x_{i+1}| \geq |x_i|$ *for* $i = 1, \ldots, n$.

(ii) $y_i \leq 0$, *if i is even,* $y_i > 0$, *if i is odd, and* $|y_{i+1}| \geq |y_i|$ *for* $i = 2, \ldots, n$.

(iii) $x_{i+1}y_i - x_i y_{i+1} = (-1)^{i+1}$ *for* $i = 0, \ldots, n$; *in particular the x_i and y_i are always coprime for* $i = 0, \ldots, n+1$.

(iv) $|x_i| \leq |b|, |y_i| \leq |a|$ *for* $i = 0, \ldots, n+1$, *if* $b \neq 0$ *resp.* $a \neq 0$.

Proof. (Sketch.) Show (i), (ii), and (iii) by induction. From $0 = r_{n+1} = |a|x_{n+1} + |b|y_{n+1}$ then follows $x_{n+1}|b$ and $y_{n+1}|a$. $\diamond$

The Euclidean algorithm is very efficient—the number of iteration steps grows only linearly with the *number of digits* of the input parameters, the entire execution time only quadratically. In the following we perform a quite exact analysis. Without loss of generality we may assume $b \neq 0$.

Given the length n of the division chain—how large must b be? We have $r_n \geq 1, r_{n-1} \geq 2$, and $r_{i-1} \geq r_i + r_{i+1}$. The FIBONACCI numbers F_n are recursively defined by

$$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2} \quad \text{for } n \geq 2.$$

Hence by induction we get $r_i \geq F_{n+2-i}$, where the induction starts with $r_n \geq 1 = F_2$, $r_{n-1} \geq 2 = F_3$; in particular we get $|b| \geq F_{n+1}$. In other words:

Proposition 2 (BINET 1841) *For $a, b \in \mathbb{Z}$ with $0 < b < F_{n+1}$ the Eucliden algorithm finds the greatest common divisor in at most $n - 1$ iteration steps.*

Addendum. *This is true also for $b = F_{n+1}$, except if $a \equiv F_{n+2} \equiv F_n \pmod{b}$.*

This gives a quite elegant mathematical formulation, but not yet an explicit bound. However the growth of the FIBONACCI numbers is well-known. One can express it by the golden section $\varphi = \frac{1+\sqrt{5}}{2}$, that is defined by $\varphi^2 - \varphi - 1 = 0$.

Lemma 3 *For a real number $c \in \mathbb{R}$ and an index $k \in \mathbb{N}$ let $F_k > c \cdot \varphi^k$ and $F_{k+1} > c \cdot \varphi^{k+1}$. Then $F_n > c \cdot \varphi^n$ for all $n \geq k$.*

Proof. (By induction.)

$$F_n = F_{n-1} + F_{n-2} > c\varphi^{n-1} + c\varphi^{n-2} = c\varphi^{n-2}(\varphi + 1) = c\varphi^n$$

for $n \geq k + 2$. $\diamond$

Corollary 1 $F_{n+1} > 0.43769 \cdot \varphi^{n+1}$ *for* $n \geq 2$.

Proof.

$$\begin{aligned} \varphi^2 &= \varphi + 1 = \frac{3 + \sqrt{5}}{2}, \\ \varphi^3 &= \varphi^2 + \varphi = 2 + \sqrt{5}, \\ \varphi^4 &= \varphi^3 + \varphi^2 = \frac{7 + 3\sqrt{5}}{2}. \end{aligned}$$

Therefore

$$\begin{aligned} \frac{F_3}{\varphi^3} &= \frac{2}{2 + \sqrt{5}} = \frac{2(\sqrt{5} - 2)}{1} = 2\sqrt{5} - 4 > 0.47, \\ \frac{F_4}{\varphi^4} &= \frac{3 \cdot 2}{7 + 3\sqrt{5}} = \frac{6(7 - 3\sqrt{5})}{49 - 45} = \frac{21 - 9\sqrt{5}}{2} > 0.43769 \end{aligned}$$

which proves the assertion. $\diamond$

Corollary 2 *Let $a, b \in \mathbb{Z}$ with $b \geq 2$. Then the number of iteration steps in the Euclidean algorithm for* $\gcd(a, b)$ *is less then* $0.718 + 4.785 \cdot {}^{10}\log(b)$.

Proof. If the division chain has length n, then $b \geq F_{n+1}$,

$$b \geq F_{n+1} > 0.43769 \cdot \varphi^{n+1},$$

$${}^{10}\log(b) > {}^{10}\log(0.43769) + (n + 1) \cdot {}^{10}\log(\varphi) > -0.35884 + 0.20898 \cdot (n + 1),$$

hence $n < 0.718 + 4.785 \cdot {}^{10}\log(b)$. $\diamond$

Somewhat coarser, but easily to remember, is the following version:

Corollary 3 *Let $a, b \in \mathbb{Z}$ with $b \geq 2$. Then the number of iteration steps in the Euclidean algorithm for* $\gcd(a, b)$ *is less then five times the number of digits of b except for $b = 8, a \equiv 5 \pmod 8$, where 5 iteration steps are needed.*

If we additionally consider the costs for the multiplication and division of large numbers depending on their number of digits, we get a working time that grows quadratically with the number of digits as shown in the following.

If a has m digits (with respect to a base B of the integers), and b has p digits, then the expense for the first division alone is already $\leq c \cdot (m-p) \cdot p$; here c is a constant that is at most twice as large as the constant that bounds the expense for "multiplying quotient $\times$ divisor back". Considering actual computer architectures we would take $B = 2^{32}$ or 2^{64}, and count the basic operations addition, subtraction, multiplication, division with remainder, and comparision of 1-digit numbers (in base B) as primitive steps. Fortunately the involved numbers shrink in an exponential way along the Euclidean division chain. The division step

$$r_{i-1} = q_i r_i + r_{i+1}$$

yet requires $\leq c \cdot {}^B\log(q_i) {}^B\log(r_i)$ primitive operations, hence the entire division chain needs

$$\begin{aligned} A(a,b) &\leq c \cdot \sum_{i=1}^{n} {}^B\log(q_i) {}^B\log(r_i) \leq c \cdot {}^B\log|b| \cdot \sum_{i=1}^{n} {}^B\log(q_i) \\ &= c \cdot {}^B\log|b| \cdot {}^B\log(q_1 \cdots q_n). \end{aligned}$$

We further estimate the product of the q_i:

$$|a| = r_0 = q_1 r_1 + r_2 = q_1(q_2 r_2 + r_3) + r_2 = \ldots = q_1 \cdots q_n r_n + \cdots \geq q_1 \cdots q_n$$

and get the coarse bound

$$A(a,b) \leq c \cdot {}^B\log|b| \cdot {}^B\log|a|\,.$$

Proposition 3 *The number of primitive operations in the Euclidean algorithm for two integers a and b with $\leq m$ digits is $\leq c \cdot m^2$.*

Note that c is a known small constant.

So the expense for the Euclidean algorithm with input a and b is not significantly larger then the expense for multiplying a and b. We won't discuss sharper estimates or potential enhancements of this bound. But note that an algorithm by LEHMER allows replacing a great amount of divisions of large numbers in the division chain by primitive operations.

3 Congruence Division

The extended Euclidean algorithm also provides a solution of the—not entirely trivial—problem of efficient division in the ring $\mathbb{Z}/n\mathbb{Z}$ of integers mod n.

Proposition 4 *Let $n \in \mathbb{N}$, $n \geq 2$, and $a, b \in \mathbb{Z}$ with $\gcd(b, n) = d$. Then a is divisible by b in $\mathbb{Z}/n\mathbb{Z}$, if and only if $d|a$. In this case there are exactly d solutions z of $zb \equiv a \pmod{n}$ with $0 \leq z < n$, and any two of them differ by a multiple of n/d. If $d = xn + yb$ and $a = td$, then $z = yt$ is a solution.*

Proof. If b divides a, then $a \equiv bz \pmod{n}$, so $a = bz + kn$, hence $d|a$. For the converse let $a = td$. By Proposition 1 we find x, y with $nx + by = d$; hence $nxt + byt = a$ and $byt \equiv a \pmod{n}$. If also $a \equiv bw \pmod{n}$, then $b(z - w) \equiv 0 \pmod{n}$, hence $z - w$ a multiple of n/d. $\diamond$

Proposition 4 contains an explicit algorithm for the division. An important special case is $d = 1$ with a notably simple formulation:

Corollary 1 *If b and n are coprime, then each a in $\mathbb{Z}/n\mathbb{Z}$ is divisible by b in a unique way.*

Since $d = 1$ the calculation of the inverse y of b follows immediately from the formula $1 = nx + by$; for $by \equiv 1 \pmod{n}$.

Corollary 2 $(\mathbb{Z}/n\mathbb{Z})^\times = \{b \bmod n \mid \gcd(b, n) = 1\}$.

Therefore the invertible elements of the ring $\mathbb{Z}/n\mathbb{Z}$ are exactly the equivalence classes of the integers coprime with n. The most important case is: $n = p$ prime:

Corollary 3 $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$ *is a field.*

Proof. For $b \in \mathbb{F}_p, b \neq 0$ there is exactly one $c \in \mathbb{F}_p$ with $bc = 1$. $\diamond$

Corollary 4 (Fermat's Little Theorem) $a^p \equiv a \pmod{p}$ *for all $a \in \mathbb{Z}$.*

Proof. The elements $\neq 0$ of $\mathbb{F}_p$ form the multiplicative group $\mathbb{F}_p^\times$. Because the order of an element always divides the group order, we have $a^{p-1} \equiv 1 \pmod{p}$ for a coprime with p. Otherwise we have $p|a$, hence $a \equiv 0 \equiv a^p \pmod{p}$. $\diamond$

4 The Chinese Remainder Algorithm

The Chinese remainder problem asks for the solution of simultaneous congruences. The simplest case worth of mention is:

Proposition 5 (Chinese Remainder Theorem) *Let m and n coprime natural numbers ≥ 1, and a, b arbitrary integers. Then there is exactly one integer x, $0 \leq x < mn$, such that*

$$x \equiv a \pmod{m}, \ x \equiv b \pmod{n}.$$

Proof. Let us first show the uniqueness: If y is another solution, then $y = x + km = x + ln$ with integers k und l, and $km = ln$. Since m and n are coprime we conclude $n|k$, $k = cn$,

$$y = x + cmn \equiv x \pmod{mn}.$$

For the existence proof we try $x = a + tm$; then necessarily $x \equiv a \pmod{m}$ and

$$x \equiv b \pmod{n} \iff b - a \equiv x - a \equiv tm \pmod{n}.$$

Such a t exists by Proposition 4. Reduce this solution $x \bmod (mn)$. $\diamond$

The proof was constructive and easily leads to an algorithm. In the general case, for multiple congruences, the Chinese remainder problem looks like follows:

- Given q pairwise coprime integers $n_1, \ldots, n_q \geq 1$ and q integers $a_1, \ldots, a_q$,
- find an integer x such that $x \equiv a_i \pmod{n_i}$ for $i = 1, \ldots q$.

One approach is suitably adapting Proposition 5. More interesting is an abstract formulation that also comprises interpolation of polynomials; also in this more general formulation we recognise Proposition 5 together with its proof, if we bear in mind that for integers m and n with greatest common divisor d we have the equivalences:

$$m, n \text{ coprime} \iff d = 1 \iff \mathbb{Z}m + \mathbb{Z}n = \mathbb{Z}.$$

Proposition 6 (General Chinese Remainder Theorem) *Let R be a commutative ring with 1, $q \geq 1$, $\mathfrak{a}_1, \ldots, \mathfrak{a}_q \trianglelefteq R$ ideals with $\mathfrak{a}_i + \mathfrak{a}_j = R$ for $i \neq j$. Let $a_1, \ldots, a_q \in R$ be given. Then there exists an $x \in R$ with $x - a_i \in \mathfrak{a}_i$ for $i = 1, \ldots, q$, and the equivalence class $x \bmod \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_q$ is uniquely determined.*

Proof. As before the uniqueness is quite simple: If $x - a_i, y - a_i \in \mathfrak{a}_i$, then $x - y \in \mathfrak{a}_i$; if this is true for all i, then $x - y \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_q$.

We prove the existence by induction on q. In the case $q = 1$ we simply take $x = a_1$. Now let $q \geq 2$, and assume y with $y - a_i \in \mathfrak{a}_i$ for $i = 1, \ldots, q-1$ is already found. Idea: We can add to y an $s \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}$ without giving up what we already have, the solution of the first $q-1$ congruences. We need the statement: For each $r \in R$ there is an $s \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}$ with $r - s \in \mathfrak{a}_q$, or in other words,

$$(\mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}) + \mathfrak{a}_q = R.$$

To prove this intermediate assertion we choose $c_i \in \mathfrak{a}_i$ for $i = 1, \ldots, q-1$ and $b_1, \ldots, b_{q-1} \in \mathfrak{a}_q$ with $b_i + c_i = 1$. Then

$$1 = (b_1 + c_1) \cdots (b_{q-1} + c_{q-1}) = c_1 \cdots c_{q-1} + b$$

with $c_1 \cdots c_{q-1} \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}$ and $b \in \mathfrak{a}_q$.

Now for $a_q - y \in R$ choose an $s \in \mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_{q-1}$ with $a_q - y - s \in \mathfrak{a}_q$, and set $x = y + s$. Then $x \equiv y \equiv a_i \pmod{\mathfrak{a}_i}$ for $i = 1, \ldots, q-1$, and $x \equiv y + s \equiv a_q \pmod{\mathfrak{a}_q}$. $\diamond$

Remarks and examples

1. For $R = \mathbb{Z}$ or any principal ideal domain, and $\mathfrak{a}_i = Rn_i$ we have $\mathfrak{a}_1 \cap \cdots \cap \mathfrak{a}_q = R(n_1 \cdots n_q)$. From this we get the usual formulation of the Chinese Remainder Theorem.

2. If R is a principal ideal domain, then the construction of the solution proceeds as follows: If $\mathfrak{a}_i = Rn_i$, then choose s in the intermediate assertion such that $s = tn_1 \cdots n_{q-1}$ with

   $$r - tn_1 \cdots n_{q-1} \in Rn_q$$

   (congruence division mod n_q). Therefore an explicit algorithm for the Chinese remainder problem exists in R, if one exists for the congruence division, in any case for $R = \mathbb{Z}$.

3. In the case $R = \mathbb{Z}$ we iteratively calculate

   $$\begin{aligned} x_1 &= a_1 \bmod n_1, & s_1 &= n_1, \\ t_i \text{ with } 0 \le t_i &\le n_i - 1 & \text{and } a_i - x_{i-1} - t_i s_{i-1} &\in Rn_i, \\ x_i &= x_{i-1} + t_i s_{i-1}, & s_i &= s_{i-1} n_i. \end{aligned}$$

   In particular $s_k = n_1 \cdots n_k$. By induction one immediately proves $0 \le x_i \le s_i - 1$ for all i. Finally one gets the solution $x = x_q$. This consideration guarantees that none of the intermediate results causes an overflow. The expense essentially consists of $q - 1$ congruence divisions and $2 \cdot (q-1)$ ordinary integer multiplications. Therefore the total expense is of order $cq\times$ (the expense for a multiplication of long integers) with a small constant c.

4. The general look of the solution formula is

   $$x = x_1 + t_1 n_1 + \cdots + t_{q-1} n_1 \cdots n_{q-1}.$$

5. As an example we treat Sun-Tsu's problem from the 1st Century. In our notation its formulation is: Find x such that

$$x \equiv 2 \pmod 3, \quad x \equiv 3 \pmod 5, \quad x \equiv 2 \pmod 7.$$

Our algorithm gives step by step:

$$\begin{aligned} x_1 = 2, &\quad s_1 = 3, \\ 1 - 3t_2 \in 5\mathbb{Z}, &\quad t_2 = 2, \\ x_2 = 2 + 2 \cdot 3 = 8, &\quad s_2 = 15, \\ -6 - 15t_3 \in 7\mathbb{Z}, &\quad t_3 = 1, \\ x = x_3 = 8 + 1 \cdot 15 = 23. & \end{aligned}$$

6. For the polynomial ring $K[T]$ over a field K the interpolation problem is a special case of the Chinese remainder problem. Our algorithm in this case is just Newton's interpolation procedure.

5 Euler's Phi Function

An important application of the Chinese Remainder Theorem follows; we assume $n \geq 2$. The integers $\bmod n$ form the ring $\mathbb{Z}/n\mathbb{Z}$. The *multiplicative group* $\bmod n$ consists of the invertible elements of this ring, and is compactly denoted by

$$\mathbb{M}_n := (\mathbb{Z}/n\mathbb{Z})^\times.$$

Its order is given by the Euler φ function:

$$\varphi(n) = \#\mathbb{M}_n = \#\{a \in [0 \cdots n-1] \mid a \text{ coprime with } n\}.$$

Corollary 1 *For m and n coprime, $\varphi(mn) = \varphi(m)\varphi(n)$.*

Proof. The Chinese Remainder Theorem just says that the natural ring homomorphism

$$F\colon \mathbb{Z}/mn\mathbb{Z} \longrightarrow \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}, \quad x \mapsto (x \bmod m, x \bmod n),$$

is bijective, hence even a ring isomorphism. Moreover $F(\mathbb{M}_{mn}) = \mathbb{M}_m \times \mathbb{M}_n$. Therefore

$$\varphi(mn) = \#\mathbb{M}_{mn} = \#\mathbb{M}_m \cdot \#\mathbb{M}_n = \varphi(m)\varphi(n),$$

as was to be shown. $\diamond$

If p is prime, then $\varphi(p) = p - 1$. More generally $\varphi(p^e) = p^e - p^{e-1} = p^e(1 - \frac{1}{p})$, if $e \geq 1$, because p^e exactly has the divisors px with $1 \leq x \leq p^{e-1}$. From Corollary 1 we conclude:

Corollary 2 *Let $n = p_1^{e_1} \cdots p_r^{e_r}$ be the prime decomposition (all $e_i \geq 1$). Then*

$$\varphi(n) = n \cdot \prod_{i=1}^{r} (1 - \frac{1}{p_i}).$$