Greatest common divisor

Application of discrete models

1 Definitions and properties

Definition 1.1. The **greatest common divisor** of $a \in \mathbb{Z}$ and $b \in \mathbb{Z}$ is the number $c \in \mathbb{N}$ for which $c \mid a \ \wedge \ c \mid b \ \wedge \ \forall d \in \mathbb{N} : d \mid a \ \wedge \ d \mid b \implies d \mid c$. Notation: $\gcd(a,b)$, or simply (a,b).
In English, the greatest common divisor of a and b is the number c which is a common divisor, and any other common divisor (d) divides c.

Example. $\gcd(12,20) = 4$, because it is a common divisor ($4 \mid 12$ and $4 \mid 20$), and for any common divisor d (1, 2 or 4), $d \mid 4$.

The greatest common divisor has a sibling:

Definition 1.2. The **least common multiple** of $a \in \mathbb{Z}$ and $b \in \mathbb{Z}$ is the number $c \in \mathbb{N}$ for which $a \mid c \ \wedge \ b \mid c \ \wedge \ \forall d \in \mathbb{N} : a \mid d \ \wedge \ b \mid d \implies c \mid d$. Notation: $\operatorname{lcm}(a,b)$, or simply $[a,b]$.
In English, the least common multiple of a and b is the number c which is a common multiple, and any other common multiple (d) is a multiple of c.

Example. $\operatorname{lcm}(12,20) = 60$, because it is a common multiple ($12 \mid 60$ and $20 \mid 60$), and for any common multiple d (e.g. $60, 120, 180, 240, \ldots$), $60 \mid d$.

The gcd and the lcm uniquely exist for any pair of integers.

(Note: the uniqueness is true because the definition only permits $c \in \mathbb{N}$, where the divisibility is antisymmetric. Some definitions of gcd and lcm allow negative results, which is more general but sacrifices uniqueness, e.g. then 2 and -2 are both greatest common divisors of 6 and 4. In general, gcd and lcm are unique *up to associatedness*.)

(Note 2: notice how useful it is that divisibility was defined so that $0 \mid 0$, because otherwise $\gcd(0,0)$ and $\operatorname{lcm}(0,a)$ would be invalid (for any $a \in \mathbb{Z}$); but now they are valid (0), and we won't need to make inconvenient "non-zero" assumptions about numbers.)

(Note 3: "greatest" and "least" are not meant in the usual sense ($\leq$, $\geq$), but with respect to divisibility as a partial order, see the comment after the definition of associates in the previous document (Divisors, prime numbers). For positive integers, this coincides with the usual order, but not for 0, which is the "greatest" here, e.g. $\gcd(0,0) = 0$, even though all integers are common divisors.)

In Sage, the greatest common divisor can be calculated by `gcd(a,b)`, and the least common multiple by `lcm(a,b)`.

Theorem 1.3 (properties of gcd and lcm)**.**

1. *for equal numbers:* $\forall a \in \mathbb{N} : \gcd(a,a) = \operatorname{lcm}(a,a) = a$*;*
2. *with zero:* $\forall a \in \mathbb{N} : \gcd(a,0) = a \ \wedge \ \operatorname{lcm}(a,0) = 0$*;*
3. *with divisor:* $\forall a,b \in \mathbb{N} : b \mid a \implies \gcd(a,b) = b \ \wedge \ \operatorname{lcm}(a,b) = a$*;*
4. *for negative numbers:* $\forall a,b \in \mathbb{Z} : \gcd(a,b) = \gcd(|a|,|b|) \ \wedge \ \operatorname{lcm}(a,b) = \operatorname{lcm}(|a|,|b|)$*;*
5. *commutativity:* $\forall a,b \in \mathbb{Z} : \gcd(a,b) = \gcd(b,a) \ \wedge \ \operatorname{lcm}(a,b) = \operatorname{lcm}(b,a)$*;*

6. *associativity:* $\forall a, b, c \in \mathbb{Z} : \gcd(a, \gcd(b, c)) = \gcd(\gcd(a, b), c) \ \wedge$
$\operatorname{lcm}(a, \operatorname{lcm}(b, c)) = \operatorname{lcm}(\operatorname{lcm}(a, b), c)$;
7. $\forall a, b \in \mathbb{Z} : \gcd(a + b, b) = \gcd(a, b) \ \wedge \ \gcd(a - b, b) = \gcd(a, b)$ *(but only for* gcd*!);*
8. *extraction:* $\forall a, b, m \in \mathbb{N} : \gcd(ma, mb) = m \cdot \gcd(a, b) \ \wedge \ \operatorname{lcm}(ma, mb) = m \cdot \operatorname{lcm}(a, b)$*;*
9. *bounds:* $\forall a, b \in \mathbb{N}^+ : 1 \leq \gcd(a, b) \leq \min(a, b) \ \wedge \ \max(a, b) \leq \operatorname{lcm}(a, b) \leq ab$*;*
10. $\forall a, b \in \mathbb{N} : \gcd(a, b) \cdot \operatorname{lcm}(a, b) = ab$.

The next to last property shows that the smallest possible value of gcd is 1. If it is exactly 1, it has a special name:

Definition 1.4. Two integers are **coprime** if their greatest common divisor is 1. (Also known as: *relatively prime.*)

Example. 5 and 8 are coprime, because $\gcd(5, 8) = 1$.

For coprime integers, lcm is also special: it follows from the last property that $\operatorname{lcm}(a, b) = ab$, which is the greatest possible value for lcm.

There are some special cases where it is easy to decide coprimality:

- 1 and -1 are coprime to any number.
- Adjacent integers are coprime (e.g. 8 and 9).
- A prime number is coprime to any integer that is not a multiple of that prime. (Thus we can say that "prime" is a stronger concept than "coprime".)
- 0 is only coprime to 1 and -1.

1.1 Generalization for multiple arguments

The greatest common divisor and least common multiple can be defined for not just two, but any number of integers:

Definition 1.5. For any $a_1, a_2, \ldots, a_n \in \mathbb{Z}$,

- $\gcd(a_1, a_2, \ldots, a_n) := \gcd(a_1, \gcd(a_2, \gcd(\ldots, a_n) \ldots))$ and
- $\operatorname{lcm}(a_1, a_2, \ldots, a_n) := \operatorname{lcm}(a_1, \operatorname{lcm}(a_2, \operatorname{lcm}(\ldots, a_n) \ldots))$.

Example. $\gcd(10, 12, 16) = \gcd(10, \gcd(12, 16)) = \gcd(10, 4) = 2$.

In Sage, `gcd()` and `lcm()` work for multiple numbers, but then they must be given in a list (or any other sequence), e.g. `gcd([10,12,16])` or `lcm(range(1, 1000))`.

Coprimality can also be generalized for multiple integers, but it is a bit trickier, because there are two ways to do this:

Definition 1.6.

- $a_1, a_2, \ldots, a_n \in \mathbb{Z}$ are **coprime** if $\gcd(a_1, a_2, \ldots, a_n) = 1$.
- $a_1, a_2, \ldots, a_n \in \mathbb{Z}$ are **pairwise coprime** if $\gcd(a_i, a_j) = 1$ for all different i and j.

Example. 8, 10 and 15 are coprime, because $\gcd(8, 10, 15) = \gcd(8, 5) = 1$, but not *pairwise* coprime, because not all pairwise gcd's are 1: $\gcd(8, 10) = 2$, $\gcd(8, 15) = 1$ and $\gcd(10, 15) = 5$.

Example. 5, 8 and 9 are pairwise coprime (too), because $\gcd(5, 8) = \gcd(5, 9) = \gcd(8, 9) = 1$.

Note: pairwise coprimality always implies coprimality, but not vice versa.

1.2 Calculating from the factorization

If we know the prime factorization of the two numbers, then it is easy to calculate their greatest common divisor and least common multiple:

Theorem 1.7.

- *The prime factorization of* $\gcd(a,b)$ *contains exactly those prime numbers which are present in* both a *and* b*, with the smaller exponent of the two.*
- *The prime factorization of* $\operatorname{lcm}(a,b)$ *contains exactly those prime numbers which are present in* either a *or* b*, with the greater exponent of the two.*

Example. $120 = 2^3 \cdot 3 \cdot 5$, $36 = 2^2 \cdot 3^2$, so $\gcd(120,36) = 2^2 \cdot 3 = 12$ and $\operatorname{lcm}(120,36) = 2^3 \cdot 3^2 \cdot 5 = 360$.

This method requires that we know the prime factorization of the numbers, which is in general very difficult to calculate for large numbers. Therefore, we need a more efficient algorithm.

2 Euclidean algorithm

The Euclidean algorithm is an efficient method for calculating the gcd of two integers.

The basic idea is that if we subtract one number from the other, then their gcd remains the same: $\gcd(a,b) = \gcd(a-b,b)$. This is useful because if $a > b > 0$, then $a - b < a$, i.e. one of the numbers was reduced. This subtraction can be repeated (with reversed roles for a and b when necessary) until both numbers will be sufficiently small. For example:

$$\begin{aligned}
\gcd(50,22) &= \\
\gcd(28,22) &= \\
\gcd(6,22) &= \\
\gcd(6,16) &= \\
\gcd(6,10) &= \\
\gcd(6,4) &= \\
\gcd(2,4) &= \\
\gcd(2,2) &= 2.
\end{aligned}$$

Based on this, we can sketch the – not yet perfect – algorithm:

Naive Euclidean algorithm(a,b):

```
Input: a, b ∈ ℕ⁺
while a ≠ b do
    if a > b then
        a := a − b
    else
        b := b − a
Output: a
```

(Note: this works only for positive integers. But if one of the numbers is zero, then we know immediately that the gcd is the other one; and if any of them is negative, then we can take its magnitude before starting the algorithm.)

But this is not the most efficient form of the Euclidean algorithm yet, because if one number is much greater than the other, then we perform too many subtractions (e.g. for $22 > 6$: $22 \to 16 \to 10 \to 4$). But these repeated subtractions can be combined into a single division with remainder (e.g. $22 \bmod 6 = 4$). With this optimization, the calculation can be done in much fewer steps:

$$\begin{aligned}\gcd(50, 22) &= \\ \gcd(6, 22) &= \\ \gcd(6, 4) &= \\ \gcd(2, 4) &= \\ \gcd(2, 0) &= 2.\end{aligned}$$

To perform this algorithm, we only need to generate a single sequence of numbers: first take the two input numbers in decreasing order, i.e. let $r_1 := a$ and $r_2 := b$ if $a > b$ (otherwise vice versa), and in each subsequent step, calculate the remainder of the last two numbers, i.e. $r_k := r_{k-2} \bmod r_{k-1}$, until this sequence reaches zero, in which case the last nonzero number is the greatest common divisor, i.e. if $r_n = 0$, then $\gcd(a, b) = r_{n-1}$. For the above example, this means the following sequence:

$$\begin{array}{c} r_1 = 50 \\ r_2 = 22 \\ \hline r_3 = 6 \\ r_4 = 4 \\ r_5 = 2 \\ r_6 = 0 \end{array}$$

On a computer, we only need to store the last two numbers (in place of a and b), i.e. if initially $a = 50$ and $b = 22$, then in the next step, $a = 22$ and $b = 6$, and so on. So:

Euclidean algorithm(a, b)**:**

> **Input:** $a, b \in \mathbb{N}$
> **while** $b \neq 0$ **do**
>> $t := b$
>> $b := a \bmod b$
>> $a := t$
>
> **Output:** a

Note: the three statements in the loop body means '$a := a \bmod b$', and then 'swap a and b' – this swap was implemented using the temporary variable t. But we can get rid of this swap entirely if the programming language supports *parallel assignments*, i.e. it can assign multiple variables at the same time (both Python and Sage can do this): then the loop body can be expressed by the single statement $a, b := b, a \bmod b$, which means to set a to the previous value of b, and simultaneously set b to $a \bmod b$, calculated from their previous values. For example, in Sage: `a, b = b, a % b`.

(Note 2: this version also works for zero. Negative numbers still require taking their magnitude – either for the inputs, or for the output.)

(Note 3: although we required above that $a > b$, but this was really just for convenience: the algorithm also works correctly if $a < b$, but it takes one more step, and the first step just exchanges the two numbers, for example $22 \bmod 50 = 22$.)

It can be shown that the Euclidean algorithm performs approximately $\log \min(a, b)$ steps for large integers. This is much faster than any known prime factorization method.

And how can we calculate the least common multiple efficiently? By first calculating gcd using the Euclidean algorithm, and then $\operatorname{lcm}(a, b) = ab \,/ \gcd(a, b)$ from the last property in the first section (or if negative numbers are allowed too, then: $\operatorname{lcm}(a, b) = |ab| \,/ \gcd(a, b)$).

3 Extended Euclidean algorithm

An important property of the greatest common divisor is that it can always be written as an integer linear combination of the two input numbers:

Theorem 3.1 (Bézout's identity). $\forall a, b \in \mathbb{Z} : \exists x, y \in \mathbb{Z} : ax + by = \gcd(a, b)$.

Example. $\gcd(16, 10) = 2 = 2 \cdot 16 - 3 \cdot 10$, i.e. the coefficients in this case are $x = 2$ and $y = -3$. This can be illustrated by jumping on a number line, where each jump can be either exactly 16 or 10 units long, in either direction. Then, starting from 0, we can reach 2 by frist jumping 16 units twice to the right, and then 10 units three times to the left.

Definition 3.2. x and y in the above theorem are called the **Bézout coefficients** for a and b.

Note: the Bézout coefficients are not unique, but more on that in the next section.

In Sage, the Bézout coefficients can be calculated by `xgcd(a,b)`, which returns a tuple `(g, x, y)`, where `g` is the greatest common divisor, and `x` and `y` are the Bézout coefficients. For example: `xgcd(16,10) == (2,2,-3)`.

But how does `xgcd()` work, i.e. how can we calculate these coefficients without Sage functions?

This can be done by a modification of the Euclidean algorithm, the *extended Euclidean algorithm* (EEA). Besides calculating the greatest common divisor, it also maintains two other values, which will be the desired x and y coefficients at the end. In each step, they are *some* coefficients of the original a and b, but first $ax + by$ does not give $\gcd(a, b)$, instead it gives the current r_i value of the algorithm. Since the first two values are the two input numbers: $r_1 = a$ and $r_2 = b$, so the first two equations are trivial: $r_1 = a = 1 \cdot a + 0 \cdot b$ and $r_2 = b = 0 \cdot a + 1 \cdot b$. Then – according to the original algorithm – $r_3 = r_1 \bmod r_2$, which can also be written as $r_3 = r_1 - q \cdot r_2$, where q is the quotient of r_1 and r_2. The extended algorithm uses the latter calculation, but not only for the r_i values, but for the whole equations, i.e. it subtracts q times the above $r_2 = \ldots$ equation from the $r_1 = \ldots$ equation.

For example, the extended Euclidean algorithm for 50 and 22 calculates the following equations:

$$\begin{array}{rrrl}
a = r_1 = 50 = & 1{\cdot}50 & + 0{\cdot}22 & \\
b = r_2 = 22 = & 0{\cdot}50 & + 1{\cdot}22 & (-2) \\
\hline
r_3 = \ \ 6 = & 1{\cdot}50 & - 2{\cdot}22 & (-3) \\
r_4 = \ \ 4 = & -3{\cdot}50 & + 7{\cdot}22 & (-1) \\
r_5 = \ \ 2 = & 4{\cdot}50 & - 9{\cdot}22 & (-2) \\
r_6 = \ \ 0 = & -11{\cdot}50 & + 25{\cdot}22 &
\end{array}$$

The numbers in parentheses are $(-q)$, i.e. the number of times that equation was subtracted from the previous one to get the next one. The solution can be obtained from the next-to-last equation: $\gcd(50, 22) = r_5 = 2$, and the Bézout coefficents are $x = 4$ and $y = -9$.

But we don't need to write down the full equations, only the left-hand-side numbers (r_i) and the corresponding pairs of coefficients (x_i and y_i):

$$\begin{array}{r|rrr}
50 & 1 & 0 & \\
22 & 0 & 1 & (-2) \\
\hline
6 & 1 & -2 & (-3) \\
4 & -3 & 7 & (-1) \\
2 & 4 & -9 & (-2) \\
0 & -11 & 25 &
\end{array}$$

Note: the first two pairs of coefficients always give the 2×2 identity matrix.

(Note 2: if 0 appears on the left, then we don't need to calculate its coefficients anymore, because we already have all the answers in the previous row. But what are these numbers in the last row anyway? Notice that they are, up to the sign, the original two numbers divided by their greatest common divisor. This will be useful later.)

So the extended Euclidean algorithm is as follows (using parallel assignments, see above):

Extended Euclidean algorithm(a, b)**:**

> **Input:** $a, b \in \mathbb{N}$
> $r',\ r := a,\ b$
> $x',\ x := 1,\ 0$
> $y',\ y := 0,\ 1$
> **while** $r \neq 0$ **do**
> > $q := r' \operatorname{div} r$
> > $r',\ r := r,\ r' - q \cdot r$ (same as: $r',\ r := r,\ r' \bmod r$ – but only for the r's!)
> > $x',\ x := x,\ x' - q \cdot x$
> > $y',\ y := y,\ y' - q \cdot y$
>
> **Output:** r', x', y'

(Note: the "prime" variables (r', x' etc.) are the values from the previous step. In Sage, we can't use this symbol, so we need to mark them in any other way, e.g. `r_old`, `x_old`, or `rr`, `xx` etc.)

4 Linear Diophantine equations

A *Diophantine equation* is an equation with integer coefficients which is solved on the set of integers. Its simplest type is the *linear* Diophantine equation, whose simplest form can be written as follows:

$$ax + by = c,$$

where $a, b, c \in \mathbb{Z}$ are constants, and $x, y \in \mathbb{Z}$ are the unknowns for which the equation is solved. For example: $16x + 10y = 6$.

Then we seek answers to the following questions:

1. Does this equation have a solution?
2. If so, how can we calculate one?
3. How many solutions are there?
4. How can we find all of them?

We have already seen a special case in the previous section: if $c = \gcd(a,b)$, then we know that the equation has a solution, and we can calculate it using the extended Euclidean algorithm.

Also, if c is the multiple of the gcd, i.e. $\gcd(a,b) \mid c$, then the equation is still solvable, and we can get a solution from that of the previous equation ($ax + by = \gcd(a,b)$) by a simple multiplication. For example, if we know that $16x + 10y = 2$ has a solution: $x = 2$ and $y = -3$, then we can get one for $16x + 10y = 6$ by multiplying them by 3: $x = 6$ and $y = -9$.

But what if $\gcd(a,b) \nmid c$? For example, consider the equation $16x + 10y = 3$ (where $2 \nmid 3$). It cannot have a solution, because the left-hand side can only be even, but the right-hand side is odd. This is true in general: $ax + by$ is always divisible by $\gcd(a,b)$ (since both a and b are), so if the right-hand side is not, then there is no solution. So we have proved the following theorem:

Theorem 4.1 (solvability)**.** *The linear Diophantine equation $ax + by = c$ is solvable if and only if $\gcd(a,b) \mid c$.*

The next question is: if it is solvable, how many solutions does it have?

To answer this, let's go back to the extended Euclidean algorithm, and take a look at its last row (with 0). For example, for $a = 50$ and $b = 22$, the solution was in the *next-to-last* row $2 = 4 \cdot 50 - 9 \cdot 22$, but after that, the *last* row was $0 = -11 \cdot 50 + 25 \cdot 22$. Why is this interesting? Because if we add the two equations together, we get $2 = -7 \cdot 50 + 16 \cdot 22$, which is another solution, since the left-hand side wasn't changed by adding 0. Furthermore, if we add the $0 = \ldots$ equation twice, three times, or any integer number of times to the original solution, we always get new and new solutions.

Therefore, there are infinitely many solutions. It can also be proven (but omitted here) that the equation has no other solutions than what we can obtain this way. Furthermore, in the last row which had the form $0 = ax_s + by_s$, it can be proven that $|x_s| = b/\gcd(a,b)$ and $|y_s| = a/\gcd(a,b)$ (e.g. $11 = 22/2$ and $25 = 50/2$), and they have opposite signs (if a and b are positive). Thus, we have the following theorem:

Theorem 4.2 (all solutions)**.** *If the linear Diophantine equation $ax + by = c$ has a solution: x_0 and y_0, then the following formula gives all the solutions:*

$$x = x_0 + k \cdot \frac{b}{\gcd(a,b)}, \quad y = y_0 - k \cdot \frac{a}{\gcd(a,b)},$$

where k is any integer.

(We got this by adding k times the equation $ax_s + by_s = 0$ to the original $ax_0 + by_0 = c$ solution. We can also prove this by putting these x and y formulas back into the original equation, making the k-parts cancel out, and getting back the first solution: $ax_0 + by_0 = c$.)

For example, consider the equation $50x + 22y = 160$. From the extended Euclidean algorithm, we know that $4 \cdot 50 - 9 \cdot 22 = 2$. Multiplying this by 80, we get a solution for the equation: $320 \cdot 50 - 720 \cdot 22 = 160$, i.e. $x_0 = 320$ and $y_0 = -720$. From this, we can calculate all the solutions using the above theorem: $x = 320 + 11k$ and $y = -720 - 25k$, where $k \in \mathbb{Z}$. For example, if $k = -30$, then $x = -10$ and $y = 30$, and indeed: $-10 \cdot 50 + 30 \cdot 22 = 160$.

The solution process can be summarized as follows:

Solving a linear Diophantine equation:

Input: $a, b, c \in \mathbb{Z}$, the parameters of the equation $ax + by = c$
$(g, x_g, y_g) :=$ Extended Euclidean algorithm(a, b)
if $g \nmid c$ **then** $\to$ *no solution*
One solution: $x_0 := x_g \cdot c/g, \quad y_0 := y_g \cdot c/g$
All solutions: $x_k := x_0 + k \cdot b/g, \quad y_k := y_0 - k \cdot a/g \quad (k \in \mathbb{Z})$

Of course, Sage can also solve linear Diophantine equations, for example:

```
sage: var("x,y");
sage: assume(x, y, "integer")
sage: solve(50*x + 22*y == 160, x, y)
(11*t_0 + 320, -25*t_0 - 720)
```

Finally: what if we restrict the solutions to $\mathbb{N}$, i.e. to the nonnegative integers? Then, if the constants (a, b and c) are also in $\mathbb{N}$, there are only finitely many solutions, because in the above theorem, x and y grow in opposite directions with k, so in either direction, one of them will be eventually negative. It is also possible that there are no solutions in $\mathbb{N}$ at all.

The equation can be solved over $\mathbb{N}$ by turning the solutions from the above theorem into the inequalities $x \geq 0$ and $y \geq 0$, and solving them for integer k's. For example, for the above equation $50x + 22y = 160$, we solve the inequalities $x = 320 + 11k \geq 0$ and $y = -720 - 25k \geq 0$, and get $-320/11 \leq k \leq -720/25$. Since k must be an integer, we can round the bounds towards k: $\lceil -320/11 \rceil \leq k \leq \lfloor -720/25 \rfloor$, i.e. $-29 \leq k \leq -29$. So we have only one solution: $k = -29$, i.e. $x = 1$ and $y = 5$. By solving these inequalities generally, we obtain the following result:

Theorem 4.3. *If $a, b, c \in \mathbb{N}^+$, then the linear Diophantine equation $ax + by = c$ has nonnegative solutions for exactly the following values of k (as defined by the above theorem):*

$$\left\lceil -\frac{\gcd(a,b)}{b} \cdot x_0 \right\rceil \leq k \leq \left\lfloor \frac{\gcd(a,b)}{a} \cdot y_0 \right\rfloor .$$

5 Author

This document was written by Uray M. János, partly using the following sources:

[1] https://compalg.elte.gitlab-pages.hu/dimoa-web/tematika/dimoa (Hungarian)

[2] https://en.wikipedia.org/wiki/Euclidean_algorithm

[3] https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm

Errors and suggestions can be reported here: uray.janos@inf.elte.hu.