Proof: By Theorem 2.3, there are $x_0, y_0, x_1, y_1 \in \mathbb{Z}$ such that $1 = ax_0 + my_0 = bx_1 + my_1$. Hence, $ax_0bx_1 = (1 - my_0)(1 - my_1) = 1 - my_2$, where $y_2 = y_0 + y_1 - my_0y_1$. Now, from $abx_0x_1 + my_2 = 1$, we conclude that $\gcd(ab, m) = 1$. □

Proposition 2.5. *If $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$.*

Proof: From $\gcd(a, b) = 1$, it follows that there are integers x and y such that $ax + by = 1$. Multiplying this equation by c, we obtain $acx + bcy = c$. Now, from $a \mid bc$, we conclude that $a \mid c$. □

2.2 Euclid's algorithm

We will now focus on the issue of efficient computing of the greatest common divisor of two integers b and c. A related issue is the problem of finding integers x and y such that $bx + cy = \gcd(b, c)$. The existence of these numbers has been proved in Theorem 2.3, but we have not yet provided any information on efficient ways of finding them.

Proposition 2.6. *Let a, b, x be integers. Then*

$$\gcd(a, b) = \gcd(a, b + ax).$$

Proof: Let $\gcd(a, b) = d$, $\gcd(a, b + ax) = g$. By Theorem 2.3, there are $x_0, y_0 \in \mathbb{Z}$ such that $d = ax_0 + by_0$, which can be written as

$$d = a(x_0 - xy_0) + (b + ax)y_0.$$

From this, it follows that $g \mid d$. Let us now show that $d \mid g$. Since $d \mid a$ and $d \mid b$, we have $d \mid (b + ax)$. Hence, d is a common divisor of a and $b + ax$, and by Theorem 2.3 we conclude that $d \mid g$. Since the numbers d and g are positive by definition, from $d \mid g$ and $g \mid d$, it follows that $d = g$. □

The previous proposition can be understood as one step in Euclid's algorithm. Since $\gcd(b, c) = \gcd(|b|, |c|)$, we can assume that b and c are positive integers.

Theorem 2.7 (Euclid's algorithm)**.** *Let b and c be positive integers and let $b > c$. Suppose that by successive application of Theorem 2.2 we obtain the*

sequence of equalities

$$\begin{aligned} b &= cq_1 + r_1, \quad 0 < r_1 < c, \\ c &= r_1q_2 + r_2, \quad 0 < r_2 < r_1, \\ r_1 &= r_2q_3 + r_3, \quad 0 < r_3 < r_2, \\ &\dots \\ r_{j-2} &= r_{j-1}q_j + r_j, \quad 0 < r_j < r_{j-1}, \\ r_{j-1} &= r_jq_{j+1}. \end{aligned}$$

Then $\gcd(b, c)$ *is equal to* r_j*, the last remainder different from zero. Values of* x *and* y *in the expression* $\gcd(b, c) = bx + cy$ *can be obtained by expressing each remainder* r_i *as a linear combination of* b *and* c*.*

Proof: By Proposition 2.6, we have

$$\begin{aligned} \gcd(b, c) &= \gcd(b - cq_1, c) = \gcd(r_1, c) = \gcd(r_1, c - r_1q_2) = \gcd(r_1, r_2) \\ &= \gcd(r_1 - r_2q_3, r_2) = \gcd(r_3, r_2). \end{aligned}$$

By continuing this procedure, we obtain

$$\gcd(b, c) = \gcd(r_{j-1}, r_j) = \gcd(r_j, 0) = r_j.$$

We will prove by induction that every r_i is a linear combination of b and c. This is true for r_1 and r_2, so let us assume that it holds for r_{i-1} and r_{i-2}. Since r_i is a linear combination of r_{i-1} and r_{i-2}, by the inductive assumption, we conclude that it is also a linear combination of b and c. □

Euclid's algorithm is one of the oldest but also one of the most important algorithms in number theory. It was named after the famous Greek mathematician Euclid (approximately 330 BC – 275 BC), who described it in his *Elements*. With the assumption that $b > c \geq 0$ (which is no loss of generality) and with an agreement that "$b \bmod c$" denotes the remainder in the division of b by c, we can summarize it as follows:

Euclid's algorithm:

while $(c > 0)$
    $(b, c) = (c, b \bmod c)$
return b

Example 2.1. *Determine* $d = \gcd(252, 198)$ *and write* d *as a linear combination of* 252 *and* 198*.*

Solution:

$$\begin{aligned} 252 &= 198 \cdot 1 + 54 \\ 198 &= 54 \cdot 3 + 36 \\ 54 &= 36 \cdot 1 + 18 \\ 36 &= 18 \cdot 2 \end{aligned}$$

Therefore, $\gcd(252, 198) = 18$. Furthermore, we have:

$$\begin{aligned} 18 &= 54 - 36 \cdot 1 = 54 - (198 - 54 \cdot 3) \cdot 1 = 4 \cdot 54 - 1 \cdot 198 \\ &= 4 \cdot (252 - 198 \cdot 1) - 1 \cdot 198 = 4 \cdot 252 - 5 \cdot 198. \end{aligned}$$ ♢

Solutions of the equation $bx + cy = \gcd(b, c)$ can be efficiently obtained in the following way: if

$$\begin{aligned} r_{-1} &= b, \quad r_0 = c; \quad r_i = r_{i-2} - q_i r_{i-1}; \\ x_{-1} &= 1, \quad x_0 = 0; \quad x_i = x_{i-2} - q_i x_{i-1}; \\ y_{-1} &= 0, \quad y_0 = 1; \quad y_i = y_{i-2} - q_i y_{i-1}, \end{aligned}$$

then

$$bx_i + cy_i = r_i, \quad \text{for } i = -1, 0, 1, \ldots, j+1.$$

This formula is clearly valid for $i = -1$ and $i = 0$, so the statement follows by induction because both sides of the formula satisfy the same recurrence relation. In particular, we have

$$bx_j + cy_j = \gcd(b, c).$$

Example 2.2. *Determine $g = \gcd(3587, 1819)$ and find integers x, y such that $3587x + 1819y = g$.*

Solution:

$$\begin{aligned} 3587 &= 1819 \cdot 1 + 1768 \\ 1819 &= 1768 \cdot 1 + 51 \\ 1768 &= 51 \cdot 34 + 34 \\ 51 &= 34 \cdot 1 + 17 \\ 34 &= 17 \cdot 2 \end{aligned}$$

i	-1	0	1	2	3	4
q_i			1	1	34	1
x_i	1	0	1	-1	35	-36
y_i	0	1	-1	2	-69	71

Hence, $g = 17$ and $3587 \cdot (-36) + 1819 \cdot 71 = 17$. $\diamondsuit$

The version of Euclid's algorithm which does not only calculate $\gcd(b, c)$ but also integers x and y such that $bx + cy = \gcd(b, c)$ is called the extended Euclid's algorithm. For a real number x, we will denote by $\lfloor x \rfloor$ the largest integer which is $\leq x$, and by $\lceil x \rceil$ the smallest integer which is $\geq x$. Then $\lfloor b/c \rfloor$ is the quotient in the division of b by c.

Extended Euclid's algorithm:

$(x, y, g, u, v, w) = (1, 0, b, 0, 1, c)$
while $(w > 0)$
    $q = \lfloor g/w \rfloor$
    $(x, y, g, u, v, w) = (u, v, w, x - qu, y - qv, g - qw)$
return (x, y, g)

Let us now discuss the efficiency of Euclid's algorithm. First of all, we are interested in the number of steps (divisions) in the algorithm. We will show that the number of steps is proportional to the number of digits of c. Such algorithms, in which the number of operations is proportional to some power of the number of digits of input data are called polynomial time algorithms.

Proposition 2.8. *The number of steps j in Euclid's algorithm satisfies $j < 2 \log_2 c$.*

Proof: Let us consider the i-th step. We have $r_i \leq \frac{r_{i-1}}{2}$ or $\frac{r_{i-1}}{2} < r_i < r_{i-1}$. In the latter case, we have $q_{i+1} = 1$ and $r_{i+1} = r_{i-1} - r_i < \frac{r_{i-1}}{2}$. Thus, in any case, $r_{i+1} < \frac{r_{i-1}}{2}$. It follows that

$$1 \leq r_j < \frac{r_{j-2}}{2} < \frac{r_{j-4}}{4} < \cdots < \frac{r_0}{2^{j/2}}$$

if j is even, and

$$2 \leq r_{j-1} < \frac{r_{j-3}}{2} < \cdots < \frac{r_0}{2^{(j-1)/2}}$$

if j is odd.

Therefore, in any case $c = r_0 > 2^{j/2}$, so $j < 2 \log_2 c$. $\square$

Note that $2\log_2 c \approx 2.885\ \ln c$. By a more precise analysis, which includes Fibonacci numbers, it can be proved that for integers $1 \leq b, c \leq N$, the number of steps in Euclid's algorithm required for calculating $\gcd(b, c)$ is less than or equal to

$$\left\lceil \frac{\ln(\sqrt{5}N)}{\ln((1+\sqrt{5})/2)} \right\rceil - 2 \approx 2.078\ \ln N\,. \tag{2.1}$$

Indeed, let $b > c$ be positive integers such that Euclid's algorithm for calculating $\gcd(b, c)$ requires n steps. Then $b \geq F_{n+2}$, $c \geq F_{n+1}$. Let us prove that using the mathematical induction over n. For $n = 1$ we have $c \geq 1 = F_2$, $b \geq 2 = F_3$. Suppose that the statement holds for $n - 1$ steps. For the numbers c and r_1, Euclid's algorithm needs $n - 1$ steps. Therefore, by the inductive assumption, we have $c \geq F_{n+1}$, $r_1 \geq F_n$. However, then $b = q_1 c + r_1 \geq c + r_1 \geq F_{n+2}$. Since $\lceil F_{k+1}/F_k \rceil = 1$ and $F_{k+1} \bmod F_k = F_{k-1}$, all quotients in Euclid's algorithm for F_{n+2} and F_{n+1} are equal to 1, and the algorithm needs exactly n steps. From that and Binet's formula for Fibonacci numbers, the estimate (2.1) follows.

It is known that the average number of steps in Euclid's algorithm for numbers b and c from the set $\{1, \ldots, N\}$ is approximately equal to

$$\frac{12\ln 2}{\pi^2}\ln N \approx 0.843\ \ln N$$

(a proof can be found in [249, Chapter 4.5.3]).

Simply put, by ignoring the constant of proportionality, the number of steps is $O(\ln N)$ (generally, the notation $f(N) = O(g(N))$ means that $|f(N)| \leq Cg(N)$ for a constant $C > 0$). Since each step of Euclid's algorithm requires one division of numbers $\leq N$, for which we need $O(\ln^2 N)$ bit operations (see [147, Chapter 5.1.2] and [249, Chapter 4.3.1]), we find that the complexity of Euclid's algorithm is $O(\ln^3 N)$. We can improve this estimate if we notice that in each step of Euclid's algorithm, we are working with smaller numbers. Thus, the number of operations is

$$\begin{aligned} O(\ln b \cdot \ln q_1 \,+\, & \ln c \cdot \ln q_2 + \ln r_1 \cdot \ln q_3 + \cdots + \ln r_{n-2} \cdot \ln q_n) \\ &= O(\ln N \cdot (\ln q_1 + \ln q_2 + \cdots + \ln q_n)) \\ &= O(\ln N \cdot \ln(q_1 q_2 \cdot \ldots \cdot q_n)) = O(\ln^2 N) \end{aligned}$$

(the last equality is obtained by multiplying all left and all right-hand sides in the equalities of Euclid's algorithm).

Proposition 2.9. *Let b and c be positive integers. Then $|x_j| \leq \frac{c}{2g}$, $|y_j| \leq \frac{b}{2g}$, where $g = \gcd(b, c)$.*

Proof: Let us prove by induction that $(-1)^i x_i \leq 0$, $(-1)^i y_i \geq 0$ for $i = -1, 0, 1, \ldots, j+1$. For $i = -1, 0$, the statement holds by definition. If we assume that it holds for $i-2$, $i-1$, then from $x_i = x_{i-2} - q_i x_{i-1}$ we get $(-1)^i x_i = (-1)^{i-2} x_{i-2} + (-1)^{i-1} q_i x_{i-1} \leq 0$. For y_i the proof is completely analogous.

Hence, $|x_i| = |x_{i-2}| + q_i |x_{i-1}|$, $|y_i| = |y_{i-2}| + q_i |y_{i-1}|$. Furthermore, since $r_{j+1} = 0$, we have $\frac{b}{g} x_{j+1} = -\frac{c}{g} y_{j+1}$, and from $\gcd(\frac{b}{g}, \frac{c}{g}) = 1$ and $\gcd(x_{j+1}, y_{j+1}) = 1$, by Proposition 2.5, it follows that $|x_{j+1}| = \frac{c}{g}$, $|y_{j+1}| = \frac{b}{g}$. If we insert this in $|x_{j+1}| = |x_{j-1}| + q_{j+1}|x_j|$, $|y_{j+1}| = |y_{j-1}| + q_{j+1}|y_j|$ and take into account that $q_{j+1} \geq 2$ (due to $r_j < r_{j-1}$), we obtain the result. □

We will describe another algorithm for calculating the greatest common divisor, the so-called binary gcd-algorithm. In this algorithm, instead of division, only the operations of subtraction and binary shift (division by 2) are used. The result is an algorithm which has a larger number of steps, but those steps are simpler. In the algorithm, we will come across two ideas. The first one is that even though factorization of positive integers is generally a difficult problem, extracting powers of 2 is simple. The second idea is replacing division by subtraction, and it is connected to the fact that very often in the original Euclid's algorithm there are instances of subtraction instead of division since the quotient is equal to 1. It can be shown that the probability that Euclid's quotient is equal to q is

$$P(q) = \log_2 \left(1 + \frac{1}{(q+1)^2 - 1}\right)$$

(see [245, Chapter 15], [249, Chapter 4.5.3], [353, Chapter 5]). Consequently, $P(1) \approx 0.415$, $P(2) \approx 0.170$, $P(3) \approx 0.093, \ldots$ Hence, in 41.5 % of cases, the quotient is equal to 1. These results are closely connected to the mentioned average number of steps in Euclid's algorithm.

Let $2^{\nu_2(k)}$ denote the largest power of 2 dividing k.

Binary gcd-algorithm:

$\beta = \min(\nu_2(a), \nu_2(b))$
$a = a/2^{\nu_2(a)}$; $b = b/2^{\nu_2(b)}$
while $(a \neq b)$,
    $(a, b) = (\min(a, b), |b - a|/2^{\nu_2(b-a)})$
return $2^\beta a$