Legnagyobb közös osztó

Diszkrét modellek alkalmazásai

1. Definíció és tulajdonságok

1.1. Definíció. Az $a \in \mathbb{Z}$ és $b \in \mathbb{Z}$ számok **legnagyobb közös osztója** [*greatest common divisor*] az a $c \in \mathbb{N}$ szám, amelyre $c \mid a \ \wedge \ c \mid b \ \wedge \ \forall d \in \mathbb{N} : d \mid a \ \wedge \ d \mid b \implies d \mid c$. Jelölés: $\text{lnko}(a, b)$, $\gcd(a, b)$ vagy simán (a, b).
Szavakkal megfogalmazva: a és b legnagyobb közös osztója az a c szám, amely közös osztójuk, de bármely d, amely szintén közös osztójuk, osztja c-t.

Példa. 12 és 20 legnagyobb közös osztója 4, mert közös osztó ($4 \mid 12$ és $4 \mid 20$), de bármilyen d számra, amely szintén közös osztó (ilyenek a $d = 1, 2, 4$), $d \mid 4$.

A legnagyobb közös osztónak van egy párja:

1.2. Definíció. Az $a \in \mathbb{Z}$ és $b \in \mathbb{Z}$ számok **legkisebb közös többszöröse** [*least common multiple*] az a $c \in \mathbb{N}$ szám, amelyre $a \mid c \ \wedge \ b \mid c \ \wedge \ \forall d \in \mathbb{N} : a \mid d \ \wedge \ b \mid d \implies c \mid d$. Jelölés: $\text{lkkt}(a, b)$, $\text{lcm}(a, b)$ vagy $[a, b]$.
Szavakkal megfogalmazva: a és b legkisebb közös többszöröse az a c szám, amely közös többszörösük, de bármely d, amely szintén közös többszörösük, többszöröse c-nek.

Példa. 12 és 20 legkisebb közös többszöröse 60, mert közös többszörös ($12 \mid 60$ és $20 \mid 60$), de bármilyen d számra, amely szintén közös többszörös (pl. $d = 60, 120, 180, 240, \ldots$), $60 \mid d$.

Az lnko és az lkkt bármely két egész számra létezik és egyértelmű.

(Megjegyzés: az egyértelműség azért van, mert $\mathbb{N}$-re szűkítettük le c-t, ahol az oszthatóság antiszimmetrikus. Néha a definícióban megengedik a negatív számokat is, de akkor az egyértelműség elvész, például akkor 6 és 4 legnagyobb közös osztója 2 és -2 is lehetne. Általában, az lnko és az lkkt *asszociáltság erejéig* egyértelmű.)

(Megjegyzés 2: itt hasznunkra válik, hogy az oszthatóságot úgy definiáltuk, hogy $0 \mid 0$, mert ellenkező esetben $\text{lnko}(0, 0)$ és $\text{lkkt}(0, a)$ értelmetlen lenne (bármely $a \in \mathbb{Z}$-re); így viszont ezek mind értelmesek (értékük 0), és nem kell majd kényelmetlen "nemnulla" kikötéseket tenni a számokra.)

(Megjegyzés 3: a két fogalomban szereplő "legnagyobb" és "legkisebb" jelző nem a szokásos értelemben ($\leq$, $\geq$) értendő, hanem az oszthatóság mint részbenrendezés szerint, lásd az előző jegyzetben (Oszthatóság, prímszámok) az asszociáltak utáni egyik megjegyzést. Ez pozitív számokra egybeesik a szokásos rendezéssel, de például 0-ra nem, amely itt a "legnagyobb", azaz pl. $\text{lnko}(0, 0) = 0$, hiába közös osztó minden más egész szám is.)

Sage-ben a legnagyobb közös osztót a `gcd(a,b)` függvénnyel, a legkisebb közös többszöröst pedig az `lcm(a,b)`-vel lehet kiszámítani.

1.3. Tétel (lnko és lkkt tulajdonságai)**.**

1. *azonos számokra:* $\forall a \in \mathbb{N} : \text{lnko}(a, a) = \text{lkkt}(a, a) = a$;
2. *nullával:* $\forall a \in \mathbb{N} : \text{lnko}(a, 0) = a \ \wedge \ \text{lkkt}(a, 0) = 0$;
3. *osztóval:* $\forall a, b \in \mathbb{N} : b \mid a \implies \text{lnko}(a, b) = b \ \wedge \ \text{lkkt}(a, b) = a$;
4. *negatív számokra:* $\forall a, b \in \mathbb{Z} : \text{lnko}(a, b) = \text{lnko}(|a|, |b|) \ \wedge \ \text{lkkt}(a, b) = \text{lkkt}(|a|, |b|)$;

5. *kommutatív:* $\forall a, b \in \mathbb{Z} : \text{lnko}(a, b) = \text{lnko}(b, a) \ \wedge \ \text{lkkt}(a, b) = \text{lkkt}(b, a)$*;*
6. *asszociatív:* $\forall a, b, c \in \mathbb{Z} : \text{lnko}(a, \text{lnko}(b, c)) = \text{lnko}(\text{lnko}(a, b), c) \ \wedge$
$\text{lkkt}(a, \text{lkkt}(b, c)) = \text{lkkt}(\text{lkkt}(a, b), c)$;
7. $\forall a, b \in \mathbb{Z} : \text{lnko}(a + b, b) = \text{lnko}(a, b) \ \wedge \ \text{lnko}(a - b, b) = \text{lnko}(a, b)$ *(csak* lnko*-ra!);*
8. *kiemelés:* $\forall a, b, m \in \mathbb{N} : \text{lnko}(ma, mb) = m \cdot \text{lnko}(a, b) \ \wedge \ \text{lkkt}(ma, mb) = m \cdot \text{lkkt}(a, b)$*;*
9. *korlátok:* $\forall a, b \in \mathbb{N}^+ : 1 \leq \text{lnko}(a, b) \leq \min(a, b) \ \wedge \ \max(a, b) \leq \text{lkkt}(a, b) \leq ab$*;*
10. $\forall a, b \in \mathbb{N} : \text{lnko}(a, b) \cdot \text{lkkt}(a, b) = ab$.

Az utolsó előtti tulajdonságból látszik, hogy az lnko lehetséges legkisebb értéke 1. Ha pontosan 1, annak külön neve is van:

1.4. Definíció. Két egész szám **relatív prím** [*coprime*], ha a legnagyobb közös osztójuk 1.

Példa. 5 és 8 relatív prímek, mert $\text{lnko}(5, 8) = 1$.

Relatív prímek esetén az lkkt is különleges: a fenti utolsó tulajdonságból következik, hogy akkor $\text{lkkt}(a, b) = ab$, ami az lkkt-re a létező legnagyobb érték.

Speciális esetekben könnyű eldönteni, hogy két szám relatív prím-e:

- Az 1 és a -1 minden egész számhoz relatív prím.
- Szomszédos számok relatív prímek (pl. 8 és 9).
- Egy prímszám minden számhoz relatív prím, kivéve a többszöröseihez. (Erre azt is mondhatjuk, hogy a "prím" erősebb fogalom, mint a "relatív prím".)
- A 0 csak az 1-hez és a -1-hez relatív prím.

1.1. Általánosítás több számra

A legnagyobb közös osztót és a legkisebb közös többszöröst nemcsak két számra lehet értelmezni, hanem akármennyire:

1.5. Definíció. Tetszőleges $a_1, a_2, \ldots, a_n \in \mathbb{Z}$ számokra

- $\text{lnko}(a_1, a_2, \ldots, a_n) := \text{lnko}(a_1, \text{lnko}(a_2, \text{lnko}(\ldots, a_n)\ldots))$ és
- $\text{lkkt}(a_1, a_2, \ldots, a_n) := \text{lkkt}(a_1, \text{lkkt}(a_2, \text{lkkt}(\ldots, a_n)\ldots))$.

Példa. $\text{lnko}(10, 12, 16) = \text{lnko}(10, \text{lnko}(12, 16)) = \text{lnko}(10, 4) = 2$.

Sage-ben a `gcd()` és az `lcm()` függvények több számra is használhatóak, de akkor azokat listában (vagy egyéb sorozattal) kell megadni, pl. `gcd([10,12,16])`, vagy `lcm(range(1, 1000))`.

Jobban kell vigyáznunk, ha a relatív prímek fogalmát szeretnénk általánosítani, mert erre két külön fogalom is van:

1.6. Definíció.

- $a_1, a_2, \ldots, a_n \in \mathbb{Z}$ **relatív prímek**, ha $\text{lnko}(a_1, a_2, \ldots, a_n) = 1$.
- $a_1, a_2, \ldots, a_n \in \mathbb{Z}$ **páronként relatív prímek**, ha $\text{lnko}(a_i, a_j) = 1$ bármely i-re és j-re $(i \neq j)$.

Példa. 8, 10 és 15 relatív prímek, mert $\text{lnko}(8, 10, 15) = \text{lnko}(8, 5) = 1$, de nem *páronként* relatív prímek, mert nem minden páronkénti lnko 1: $\text{lnko}(8, 10) = 2$, $\text{lnko}(8, 15) = 1$ és $\text{lnko}(10, 15) = 5$.

Példa. 5, 8 és 9 páronként (is) relatív prímek, mert $\text{lnko}(5, 8) = \text{lnko}(5, 9) = \text{lnko}(8, 9) = 1$.

Megjegyzés: a páronként relatív prímek mindig relatív prímek is, de ez fordítva nem igaz.

1.2. Kiszámítás prímfelbontásból

Ha ismerjük a két szám prímtényezős felbontását, akkor abból könnyen kiszámíthatjuk a legnagyobb közös osztójukat és a legkisebb közös többszörösüket:

1.7. Tétel.

- $\text{lnko}(a, b)$ *prímtényezős felbontásában pontosan azok a prímtényezők szerepelnek, amelyek* a*-ban és* b*-ben is szerepelnek, és a kettő közül a kisebbik kitevővel.*
- $\text{lkkt}(a, b)$ *prímtényezős felbontásában pontosan azok a prímtényezők szerepelnek, amelyek vagy* a*-ban, vagy* b*-ben szerepelnek, és a kettő közül a nagyobbik kitevővel.*

Példa. $120 = 2^3 \cdot 3 \cdot 5$ és $36 = 2^2 \cdot 3^2$, ezért $\text{lnko}(120, 36) = 2^2 \cdot 3 = 12$ és $\text{lkkt}(120, 36) = 2^3 \cdot 3^2 \cdot 5 = 360$.

Ez a számítás azonban feltételezi, hogy ismerjük a két szám prímfelbontását, amit nagy számok esetén nagyon nehéz kiszámítani. Ezért szükségünk lesz egy hatékonyabb algoritmusra.

2. Euklideszi algoritmus

Az euklideszi algoritmus egy hatékony eljárás a legnagyobb közös osztó kiszámítására.

Az alapötlet az, hogy ha az egyik számból kivonjuk a másikat, akkor a legnagyobb közös osztó nem változik: $\text{lnko}(a, b) = \text{lnko}(a - b, b)$. Ez azért hasznos, mert ha $a > b > 0$, akkor $a - b < a$, vagyis csökkentettük az egyik számot. Ezt a helyettesítést (felváltva a és b szerepével) addig csinálhatjuk, amíg mindkét szám elég kicsi nem lesz. Például:

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

Ennek megfelelően felírhatjuk a – még nem tökéletes – algoritmust:

Naiv euklideszi algoritmus(a, b)**:**

Bemenet: $a, b \in \mathbb{N}^+$
ciklus amíg $a \neq b$
  ha $a > b$
    $a := a - b$
  különben
    $b := b - a$
Kimenet: a

(Megjegyzés: ez így csak pozitív egész számokra működik. Ha valamelyik bemenő szám nulla, akkor rögtön tudjuk, hogy az lnko a másik szám lesz. Ha valamelyik szám negatív, akkor vegyük az ellentettjét, mielőtt elindítjuk az algoritmust.)

Ez azonban még nem a leghatékonyabb formája az euklideszi algoritmusnak. Ha ugyanis az egyik szám jóval nagyobb, mint a másik, akkor szokszor vonjuk le belőle a másik számot (pl. 22-ből a 6-ot: $22 \to 16 \to 10 \to 4$). Ezeket a kivonásokat összevonhatjuk egyetlen műveletté, egy maradékos osztássá, például $22 \bmod 6 = 4$. Ezzel a gyorsítással a fenti számítás kevesebb lépésben végezhető el:

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

Az algoritmus elvégzéséhez elegendő egyetlen számsorozatot előállítani: először vegyük a két bemenő számot csökkenő sorrendben, azaz legyen $r_1 := a$ és $r_2 := b$, ha $a > b$ (különben fordítva), majd számítsuk ki minden lépésben az eddigi utolsó két szám maradékát, azaz $r_k := r_{k-2} \bmod r_{k-1}$, egészen addig, amíg el nem érjük a nullát, és akkor az előző szám lesz a legnagyobb közös osztó, azaz ha $r_n = 0$, akkor $\mathrm{lnko}(a, b) = r_{n-1}$. A fenti példa esetén tehát ezt a sorozatot állítjuk elő:

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

Számítógépen elegendő mindig csak az utolsó két számot tárolni (a és b helyén), azaz ha kezdetben $a = 50$ és $b = 22$, akkor a következő lépésben $a = 22$ és $b = 6$, és így tovább. Tehát:

Euklideszi algoritmus(a, b):

- **Bemenet:** $a, b \in \mathbb{N}$
- **ciklus amíg** $b \neq 0$
  - $t := b$
  - $b := a \bmod b$
  - $a := t$
- **Kimenet:** a

Megjegyzés: a ciklusmag három utasítása helyett írhattuk volna azt is, hogy $a := a \bmod b$, majd cseréljük fel az a-t és a b-t – ezt a cserét oldottuk meg az ideiglenes t változóval. De megúszhatjuk a cserét, ha a programozási nyelv támogatja a *szimultán értékadást*, azaz egyszerre több változó értékét is meg tudja változtatni (pl. ilyen a Python és a Sage is): ekkor az egész ciklusmagot írhatjuk egyetlen utasítással úgy is, hogy $a, b := b, a \bmod b$, azaz legyen a az előző b, és egyúttal legyen b az előző értékekből számított $a \bmod b$. Például Sage-ben: `a, b = b, a % b`.

(Megjegyzés 2: ez a verzió már nullára is működik. Negatív számok esetén továbbra is abszolút értéket kell venni – akár a bemenetekét, akár a végeredményét.)

(Megjegyzés 3: bár fent kikötöttük, hogy $a > b$, de ez valójában csak egy kényelmi kikötés volt: ha $a < b$, akkor is működik az algoritmus, csak eggyel több lépésben, és az első lépés pont felcseréli a két számot, például $22 \bmod 50 = 22$.)

Bebizonyítható, hogy az euklideszi algoritmus nagy számokra nagyságrendileg $\log \min(a, b)$ lépést végez. Ez sokkal gyorsabb, mint bármilyen ismert prímfaktorizációs algoritmus.

És hogyan lehet hatékonyan kiszámítani a legkisebb közös többszöröst? Ehhez először kiszámítjuk az lnko-t az euklideszi algoritmussal, majd $\text{lkkt}(a, b) = ab \,/\, \text{lnko}(a, b)$ a fenti legutolsó tulajdonság alapján (vagy ha negatív számok is lehetnek: $\text{lkkt}(a, b) = |ab| \,/\, \text{lnko}(a, b)$).

3. Bővített euklideszi algoritmus

A legnagyobb közös osztó egy fontos tulajdonsága, hogy mindig előállítható az eredeti két szám egész együtthatós lineáris kombinációjaként:

3.1. Tétel (Bézout-lemma)**.** $\forall a, b \in \mathbb{Z} : \exists x, y \in \mathbb{Z} : ax + by = \text{lnko}(a, b)$.

Példa. $\text{lnko}(16, 10) = 2 = 2 \cdot 16 - 3 \cdot 10$, azaz az együtthatók lehetnek $x = 2$ és $y = -3$.
Ezt úgy szemléltethetjük, hogy egy számegyenesen lépkedünk, de egyszerre mindig csak 16- vagy 10-hosszú lépéseket tehetünk. Ekkor el tudunk jutni a 0-ból a 2-be úgy, hogy jobbra lépünk kétszer 16-ot, majd balra háromszor 10-et.

3.2. Definíció. A fenti tételben szereplő x-et és y-t a és b **Bézout-együtthatói**nak nevezzük.

A Bézout-együtthatók nem egyértelműek, erről részletesebben a következő szakaszban lesz szó.

Sage-ben a Bézout-együtthatókat az `xgcd(a,b)` függvénnyel számíthatjuk ki, amely visszaad egy `(g, x, y)` hármast, ahol `g` a legnagyobb közös osztó, és `x` és `y` a Bézout-együtthatók. Például: `xgcd(16,10) == (2,2,-3)`.

De hogyan működik az `xgcd()` függvény, azaz hogyan számíthatjuk ki ezeket az együtthatókat?

Erre az euklideszi algoritmus egy módosítását használjuk, a *bővített euklideszi algoritmust*. Ez amellett, hogy kiszámítja a legnagyobb közös osztót, közben további értékeket is számol, amelyek végül a kívánt Bézout-együtthatók lesznek. Ezek minden lépésben az eredeti a és b *valamilyen* x és y együtthatói, de $ax + by$ kezdetben nem $\text{lnko}(a, b)$-t állítja elő, hanem az algoritmus éppen aktuális r_i értékét. Az első két érték a két bemenő szám: $r_1 = a$ és $r_2 = b$, ezért az első két egyenlet triviális: $r_1 = a = 1 \cdot a + 0 \cdot b$ és $r_2 = b = 0 \cdot a + 1 \cdot b$. Ezután – az eredeti algoritmus szerint – $r_3 = r_1 \bmod r_2$, amit úgy is felírhatunk, hogy $r_3 = r_1 - q \cdot r_2$, ahol q a maradékos osztás hányadosa. A bővített algoritmus az utóbbi összefüggést használja, de nemcsak az r_i számokra, hanem a teljes egyenletre, azaz az $r_1 = \ldots$ fenti egyenletből kivonja az $r_2 = \ldots$ egyenlet q-szorosát.

Például 50 és 22 bővített euklideszi algoritmusa során az együtthatók így fognak változni:

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

Minden lépésben jeleztük $(-q)$-t, azaz hogy az egyenlet hányszorosát vontuk le az előző egyenletből, hogy megkapjuk a következőt. A megoldást az utolsó előtti sorból olvashatjuk le: $\text{lnko}(50, 22) = r_5 = 2$, a Bézout-együtthatók pedig $x = 4$ és $y = -9$.

Nem kell mindig kiírni a teljes egyenleteket, elég csak az együtthatópárokat és a bal oldalt:

50	1	0	
22	0	1	(-2)
6	1	-2	(-3)
4	-3	7	(-1)
2	4	-9	(-2)
0	-11	25	

Megjegyzés: vegyük észre, hogy a két bemenő számhoz tartozó együtthatópár éppen a 2×2-es egységmátrixot alkotja.

(Megjegyzés 2: ha bal oldalt kijött a 0, akkor a hozzá tartozó együtthatókat már nem szükséges kiszámítani, mert az előző sorban megkaptuk a Bézout-együtthatókat. De mit kaptunk az utolsó sorban? Vegyük észre, hogy azok – előjeltől eltekintve – éppen az eredeti két szám, leosztva a legnagyobb közös osztójukkal. Ez később még hasznos lesz.)

A bővített euklideszi algoritmus tehát így működik (szimultán értékadásokkal, lásd fent):

Bővített euklideszi algoritmus(a, b)**:**

Bemenet: $a, b \in \mathbb{N}$

$r',\ r := a,\ b$

$x',\ x := 1,\ 0$

$y',\ y := 0,\ 1$

ciklus amíg $r \neq 0$

$q := r' \operatorname{div} r$

$r',\ r := r,\ r' - q \cdot r$ (ugyanaz, mint: $r',\ r := r,\ r' \bmod r$ – de csak az r-ekre!)

$x',\ x := x,\ x' - q \cdot x$

$y',\ y := y,\ y' - q \cdot y$

Kimenet: r', x', y'

(Megjegyzés: a "vesszős" változók (r', x' stb.) az előző lépésből való értékeket jelölik. Sage-ben, ahol nincs "vessző", ezeket másképp kell jelölni, pl. `r_old`, `x_old`, vagy `rr`, `xx` stb.)

4. Lineáris diofantikus egyenletek

A *diofantikus egyenlet* olyan egész együtthatós egyenlet, amelynek megoldásait az egész számok körében keressük. Ezek legegyszerűbb fajtája a *lineáris* diofantikus egyenlet, amely legegyszerűbb formájában a következőképpen írható fel:

$$ax + by = c,$$

ahol $a, b, c \in \mathbb{Z}$ paraméterek és $x, y \in \mathbb{Z}$ az ismeretlenek, amelyekre meg szeretnénk oldani az egyenletet. Például: $16x + 10y = 6$.

Ezek kapcsán a következő kérdésekre keressük a választ:

1. Van-e megoldása az egyenletnek?
2. Ha van, hogyan tudjuk kiszámítani egy megoldását?
3. Hány megoldása van?
4. Hogyan tudjuk felírni az összes megoldását?

Az előző szakaszban már megoldottunk egy speciális esetet: ha $c = \text{lnko}(a, b)$, akkor tudjuk, hogy létezik megoldás, és a bővített euklideszi algoritmussal ki is tudunk számítani egyet.

Az is könnyen belátható, hogy ha c az lnko többszöröse, azaz $\text{lnko}(a, b) \mid c$, akkor is van megoldás, és azt megkaphatjuk az előző egyenlet ($ax + by = \text{lnko}(a, b)$) megoldásaiból egyszerű szorzással. Például ha tudjuk, hogy a $16x + 10y = 2$ egy megoldása $x = 2$ és $y = -3$, akkor a $16x + 10y = 6$ egy megoldása ennek a 3-szorosa lesz: $x = 6$ és $y = -9$.

De mi a helyzet akkor, ha $\text{lnko}(a, b) \nmid c$? Például nézzük a $16x + 10y = 3$ egyenletet ($2 \nmid 3$). Nyilvánvaló, hogy ennek nincs megoldása, hiszen a bal oldal csak páros lehet, a jobb oldal pedig páratlan. Ez általában is igaz: $ax + by$ mindenképpen osztható $\text{lnko}(a, b)$-vel (hiszen mind a, mind b osztható vele), ezért ha a jobb oldal nem, akkor nincs megoldás. Ezzel megkaptuk a következő tételt:

4.1. Tétel (megoldhatóság)**.** *Az $ax + by = c$ lineáris diofantikus egyenletnek akkor és csak akkor van megoldása, ha* $\text{lnko}(a, b) \mid c$.

A következő kérdés, hogy ha van megoldás, akkor hány van.

Ehhez menjünk vissza a bővített euklideszi algoritmushoz, és vessünk egy pillantást az utolsó (0-s) sorra. Például $a = 50$ és $b = 22$ esetén az utolsó *előtti* sor az volt, hogy $2 = 4 \cdot 50 - 9 \cdot 22$, és ez adta meg az (egyik) megoldást, de az *utolsó* sora az volt, hogy $0 = -11 \cdot 50 + 25 \cdot 22$. Miért érdekes ez? Mert ha összeadjuk a két egyenletet: $2 = -7 \cdot 50 + 16 \cdot 22$, akkor szintén egy megoldást kapunk, hiszen a bal oldal a 0 miatt nem változott. Sőt, a $0 = \ldots$ egyenlet kétszeresét, háromszorosát, vagy bármely egész számú többszörösét is hozzáadhatjuk az eredeti megoldáshoz, és így mindig újabb és újabb megoldásokat kapunk.

Ezzel beláttuk, hogy végtelen sok megoldás van. Sőt, az is bebizonyítható (ezt mellőzzük), hogy csak a fenti módon megkapható megoldások vannak. Belátható az is, hogy az utolsó, $0 = ax_s + by_s$ alakú egyenletben $|x_s| = b/\text{lnko}(a, b)$ és $|y_s| = a/\text{lnko}(a, b)$ (pl. $11 = 22/2$ és $25 = 50/2$), előjelük pedig ellentétes (ha a és b pozitívak). Ezzel készen is van a következő tétel:

4.2. Tétel (összes megoldás)**.** *Ha az $ax + by = c$ lineáris diofantikus egyenlet megoldható, és egy megoldása x_0 és y_0, akkor az összes megoldását a következőképpen írhatjuk fel:*

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

ahol k végigfut az egész számok halmazán.

(Ezt úgy kaptuk, hogy az $ax_0 + by_0 = c$ megoldáshoz hozzáadtuk a fenti $ax_s + by_s = 0$ egyenlet k-szorosát. Vagy úgy is bebizonyíthatjuk, hogy a tétel x és y megoldását visszahelyettesítjük az eredeti egyenletbe, ekkor kiesnek a k-s részek, és visszakapjuk az $ax_0 + by_0 = c$ megoldást.)

Például tekintsük a következő egyenletet: $50x + 22y = 160$. Fent már kiszámítottuk a bővített euklideszi algoritmussal, hogy $4 \cdot 50 - 9 \cdot 22 = 2$. Ha ezt megszorozzuk 80-nal, akkor megkapjuk az egyenlet egyik megoldását: $320 \cdot 50 - 720 \cdot 22 = 160$, azaz $x_0 = 320$ és $y_0 = -720$. A fenti tétel alapján pedig felírhatjuk az összes megoldását: $x = 320 + 11k$ és $y = -720 - 25k$, ahol $k \in \mathbb{Z}$. Például ha $k = -30$, akkor $x = -10$ és $y = 30$, és valóban: $-10 \cdot 50 + 30 \cdot 22 = 160$.

A megoldás menetét a következőképpen foglalhatjuk össze:

Lineáris diofantikus egyenlet megoldása:

Bemenet: $a, b, c \in \mathbb{Z}$, az egyenlet ($ax + by = c$) paraméterei
$(g, x_g, y_g) :=$ bővített euklideszi algoritmus(a, b)
ha $g \nmid c \rightarrow$ *nincs megoldás*
Egy megoldás: $x_0 := x_g \cdot c/g, \;\; y_0 := y_g \cdot c/g$
Összes megoldás: $x_k := x_0 + k \cdot b/g, \;\; y_k := y_0 - k \cdot a/g \quad (k \in \mathbb{Z})$

Lineáris diofantikus egyenleteket természetesen a Sage is meg tud oldani, például:

```
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)
```

Végezetül: mi a helyzet akkor, ha egy lineáris diofantikus egyenletet kifejezetten $\mathbb{N}$ fölött szeretnénk megoldani, azaz a negatív megoldásai nem érdekelnek? Ekkor, ha a paraméterek (a, b és c) is $\mathbb{N}$-ben vannak, akkor csak véges sok megoldás lesz, hiszen a fenti képletben x és y ellenkező irányban változik k-val, tehát mindkét irányban előbb-utóbb valamelyik negatív lesz. Az is előfordulhat, hogy egyáltalán nincs megoldás.

Az egyenlet összes $\mathbb{N}$ fölötti megoldását megkaphatjuk, ha a fenti tételből kapott – k-tól függő – megoldásokra felírjuk az $x \geq 0$ és $y \geq 0$ egyenlőtlenségeket, ezeket megoldjuk k-ra, és kiválogatjuk az egész k-kat. Például a fenti $50x + 22y = 160$ egyenlet esetén megoldjuk az $x = 320 + 11k \geq 0$ és az $y = -720 - 25k \geq 0$ egyenlőtlenségeket: $-320/11 \leq k \leq -720/25$. Mivel k egész, ezért vehetjük a k-hoz közelebbi egészrészeket: $\lceil -320/11 \rceil \leq k \leq \lfloor -720/25 \rfloor$, vagyis $-29 \leq k \leq -29$. Itt tehát egyetlen megoldás van: $k = -29$, azaz $x = 1$ és $y = 5$. Ha általánosan oldjuk meg az egyenlőtlenségeket, akkor a következőt kapjuk:

4.3. Tétel. *Ha $a, b, c \in \mathbb{N}^+$, akkor az $ax+by = c$ diofantikus egyenlet fenti tételben szereplő megoldási közül az alábbi k-kra kapunk $\mathbb{N}$-beli megoldásokat:*

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

5. Szerző

Ezt a jegyzetet Uray M. János írta, részben felhasználva az alábbi forrásokat:

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

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

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

Hibákat, észrevételeket itt lehet jelezni: uray.janos@inf.elte.hu.