Oszthatóság, prímszámok

Diszkrét modellek alkalmazásai

Számelméletben elsősorban egész számokkal ($\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$) foglalkozunk, ill. speciális esetben természetes számokkal ($\mathbb{N} = \{0, 1, 2, \ldots\}$).

1. Maradékos osztás

Az egész számok körében a négy alapművelet ($+$, $-$, $\cdot$, $/$) közül csak három működik általánosan, mert az osztás nem mindig egész számot eredményez, pl. $2/3 \notin \mathbb{Z}$.

Ezért az osztás helyett bevezetjük a **maradékos osztás** fogalmát, amely az egész számok körében is mindig elvégezhető (ha a nevező nem nulla). Ez a művelet következő tételen alapul:

1.1. Tétel (Maradékos osztás tétele)**.** *Ha $a, b \in \mathbb{Z}$ és $b \neq 0$, akkor egyértelműen létezik olyan $q, r \in \mathbb{Z}$, hogy $a = qb + r$ és $0 \leq r < |b|$.*

1.2. Definíció. A fenti tételben szereplő q az a és b **hányadosa** [*quotient*] (jelölés: a div b), r pedig a **maradéka** [*remainder*] (jelölés: a mod b). A q és r kiszámítását **maradékos osztás**nak nevezzük.

Példa. Ha $a = 36$ és $b = 10$, akkor $q = 3$ és $r = 6$ (36-ban a 10 megvan 3-szor, és maradt a 6), mert $36 = 3 \cdot 10 + 6$ és $0 \leq 6 < 10$. Azt is írhatjuk, hogy $36 \text{ div } 10 = 3$ és $36 \bmod 10 = 6$.

Példa. Ha $a = -36$ és $b = 10$, akkor $q = -4$ és $r = 4$. (Nem jó a $q = -3$ és $r = -6$, mert a tétel szerint r nem lehet negatív.)

Sage-ben a hányadost két perjellel (`//`), a maradékot pedig százalékjellel (`%`) számíthatjuk ki:

```
sage: 36 // 10
3
sage: 36 % 10
6
```

A kettőt egyszerre is kiszámíthatjuk (hatékonyabban, mint külön-külön):

```
sage: divmod(36, 10)
(3, 6)
```

Vigyázat: a sima osztást (`/`) kerüljük, ha egész számokkal számolunk, mert az eredmény nem `Integer` (egész szám) típusú lesz, hanem `Rational`, még akkor is, ha egyébként matematikailag egész szám az eredmény. Ez problémákhoz vezethet, ezért használjuk inkább mindig a maradékos osztás (`//`) műveletét, amely `Integer`-t ad. Például:

```
sage: 36 / 10
18/5                # nem egész szám
sage: 30 / 10
3                   # látszólag egész szám...
```

```
sage: type(30 / 10)
<class 'sage.rings.rational.Rational'>  # de a típusa nem Integer!
sage: type(30 // 10)
<class 'sage.rings.integer.Integer'>    # csak így lesz Integer
sage: # És a típus fontos, például a "prím" racionális számokra értelmetlen:
sage: (30 / 10).is_prime()
False
sage: (30 // 10).is_prime()
True
```

2. Osztó

2.1. Definíció. $a \in \mathbb{Z}$ **osztható** $b \in \mathbb{Z}$-vel [*divisible*], ha $\exists c \in \mathbb{Z} : a = bc$. Jelölés: $b \mid a$. "Nem osztható" jelölése: $b \nmid a$. Azt is mondjuk még, hogy b **osztója** a-nak [*divisor*], valamint a **többszöröse** b-nek [*multiple*].

Példa. 6 osztható 2-vel (2 osztója 6-nak), mert $6 = 2 \cdot 3$.

(Megjegyzés: bár "oszthatóság" a neve, mégsem osztással, hanem szorzással definiáljuk. Egyrészt azért, mert a szorzás egyszerűbb művelet, másrészt pedig így nem kell kilépnünk a $\mathbb{Z}$-ből. Ez majd lehetővé fogja tenni, hogy kiterjesszük az oszthatóságot olyan struktúrákra is, ahol egyáltalán nincs osztás.)

Vegyük észre, hogy a fenti definíció szerint 0 osztható 0-val (ez a konvenció később még hasznos lesz), tehát nem igaz, hogy "a osztható b-vel, ha a/b egész szám". Ez csak akkor igaz, ha kikötjük, hogy $b \neq 0$. Hasonló igaz a maradékos osztásra is:

2.2. Tétel (az oszthatóság kapcsolata az osztással és a maradékos osztással)**.**
Ha $a, b \in \mathbb{Z}$ *és* $\boldsymbol{b \neq 0}$*, akkor:*

1. $b \mid a \iff a/b \in \mathbb{Z}$*;*
2. $b \mid a \iff a \bmod b = 0$.

2.3. Tétel (az oszthatóság tulajdonságai)**.**

1. *reflexív:* $\forall a \in \mathbb{Z} : a \mid a$ *(minden szám osztható önmagával);*
2. $\forall a \in \mathbb{Z} : a \mid 0$ *(a* 0 *minden számmal osztható);*
3. $\forall a \in \mathbb{Z} : 1 \mid a$ *(minden szám osztható 1-gyel);*
4. $\forall a \in \mathbb{Z} : 0 \mid a \iff a = 0$ *(csak a* 0 *osztható* 0*-val);*
5. *tranzitív:* $\forall a, b, c \in \mathbb{Z} : a \mid b \wedge b \mid c \implies a \mid c$*;*
6. $\mathbb{N}$*-ben antiszimmetrikus:* $\forall a, b \in \mathbb{N} : a \mid b \wedge b \mid a \implies a = b$*;*
7. $\forall a, b, c \in \mathbb{Z} : a \mid b \wedge a \mid c \implies a \mid b + c$ *(ha minden tagnak osztója, akkor az összegnek is);*
8. $\forall a, b, c \in \mathbb{Z} : a \mid b \wedge a \mid c \implies a \mid b - c$*;*
9. $\forall a, b, c, d \in \mathbb{Z} : a \mid b \wedge c \mid d \implies ac \mid bd$*;*
10. $\forall a, b \in \mathbb{N}^+ : a \mid b \implies a \leq b$.

Kapcsolódó fogalmak:

2.4. Definíció. $b \in \mathbb{N}$ **valódi osztója** $a \in \mathbb{N}$-nek [*proper divisor*], ha osztója, és $b \neq a$.

2.5. Definíció. $a \in \mathbb{Z}$ **triviális osztói** [*trivial divisors*] az 1, -1, a és $-a$.

Példa. $\mathbb{N}$-ben 6 osztói: $1, 2, 3, 6$; valódi osztói: $1, 2, 3$; nemtriviális osztói: $2, 3$.

Sage-ben az oszthatósággal kapcsolatban a következő műveleteket használhatjuk:

```
sage: 5.divides(10)
True
sage: 5.divides(12)
False
sage: 0.divides(0)
True
sage: divisors(12)   # pozitív osztók listája
[1, 2, 3, 4, 6, 12]
sage: # Maradékos osztással (ha az osztó nem nulla):
sage: 10 % 5 == 0
True
sage: 12 % 5 == 0
False
sage: 0 % 0 == 0
[...]
ZeroDivisionError: Integer modulo by zero
```

Vezessünk be még egy kapcsolódó fogalmat:

2.6. Definíció. a és b **asszociáltak** [*associates*], ha $a \mid b$ és $b \mid a$.

$\mathbb{N}$-ben csak az egyenlő számok asszociáltak. $\mathbb{Z}$-ben viszont a és b akkor és csak akkor asszociáltak, ha $a = b$ vagy $a = -b$, tehát nemcsak az egyenlő számok, hanem az ellentettpárok is azok.

Miért érdekesek az asszociáltak? Ha két szám asszociált, akkor oszthatóság szempontjából pontosan ugyanúgy viselkednek, pl. $2 \mid 6 \implies 2 \mid -6,\ -2 \mid 6,\ -2 \mid -6$. Így tehát az asszociáltak közül legtöbbször elég csak az egyiket vizsgálni, például $\mathbb{Z}$-ben elég csak $\mathbb{N}$-nel foglalkozni.

(Megjegyzés: innen még nem látszik, hogy miért van külön neve az asszociáltaknak, és miért nem csak úgy hívjuk őket, hogy "azonos abszolút értékű számok". Később viszont majd általánosítjuk az oszthatóságot más struktúrákra, és ott az asszociáltnak már nem lesz köze az abszolút értékhez és az előjelhez.)

Az asszociáltság ekvivalenciareláció, azaz reflexív, szimmetrikus és tranzitív reláció.

Az oszthatóság fenti tulajdonságainál láthattuk, hogy $\mathbb{N}$-ben antiszimmetrikus, és ezért részbenrendezés is (= reflexív + antiszimmetrikus + tranzitív). De miért kellett kikötni, hogy $\mathbb{N}$-ben? Ha összevetjük az asszociáltság és az antiszimmetria definícióját, akkor látható, hogy az oszthatóság akkor és csak akkor lesz antiszimmetrikus (és így részbenrendezés), ha csak az egyenlő számok asszociáltak, tehát $\mathbb{N}$-ben igen, de $\mathbb{Z}$-ben nem.

(Vegyük észre, hogy $\mathbb{N}$-ben az oszthatóság mint részbenrendezés bár többnyire ugyanúgy "rendez", mint a természetes rendezés ($\leq$) – lásd az oszthatóság fenti utolsó tulajdonságát –, de nem mindig: a kivétel a 0, amely az $\mathbb{N}$ legkisebb eleme, de az oszthatóság szempontjából a "legnagyobb", hiszen minden szám osztja a 0-t.)

3. Prímszámok

3.1. Definíció. Egy $n \in \mathbb{N}$ szám **prímszám** [*prime number*], ha $n > 1$ és csak triviális osztói vannak. Másképpen megfogalmazva: n prímszám, ha pontosan két pozitív osztója van (1 és n).

Példa. Az első néhány prímszám: $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, \ldots$.

3.2. Definíció. Egy $n \in \mathbb{N}$ szám **összetett szám** [*composite number*], ha $n > 1$ és nem prímszám.

Megjegyzés: az 1 nem prímszám és nem is összetett szám. Hogy ezt miért szokás így definiálni, azt majd később (a számelmélet alaptételénél) fogunk megérteni.

A Sage néhány prímszámokkal kapcsolatos művelete:

```
sage: is_prime(7)          # vagy: 7.is_prime()
True
sage: next_prime(1000)
1009
sage: previous_prime(1000)
997
sage: nth_prime(4)         # a negyedik prímszám
7
sage: primes_first_n(4)    # az első 4 prímszám
[2, 3, 5, 7]
sage: prime_range(20)      # mint a range(), csak prímszámokra
[2, 3, 5, 7, 11, 13, 17, 19]
sage: prime_range(10, 20)
[11, 13, 17, 19]
sage: random_prime(1000)   # véletlen prímszám 2-től 1000-ig
193
sage: random_prime(1000, lbound=500)   # 500-tól 1000-ig
613
```

Figyeljük meg, hogy milyen számjegyekre végződhetnek a prímszámok. A 2 kivételével a végződés csak páratlan lehet, mert különben a prímszám osztható lenne 2-vel. Hasonlóan, az 5 kivételével nem végződhetnek 5-re, mert akkor oszthatóak lennének 5-tel. Ezért tízes számrendszerben a $p \geq 7$ prímszámok lehetséges végződései: 1, 3, 7 és 9.

3.3. Definíció. Egy p prímszám **ikerprím** [*twin prime*], ha $p-2$ vagy $p+2$ is prímszám. A két prímszámot együtt **ikerprímpár**nak nevezzük.

Példa. Az első néhány ikerprímpár: $(3,5), (5,7), (11,13), (17,19), (29,31), \ldots$.

3.1. Próbaosztás

Hogyan tudjuk eldönteni – a Sage függvényei nélkül –, hogy egy adott $n \in \mathbb{N}$ szám prímszám-e? A legegyszerűbb módszer a *próbaosztás*: megnézzük az összes lehetséges számra, hogy osztója-e.

De mik a "lehetséges számok"? Mivel 1 és n biztosan osztója, n-nél nagyobb osztója pedig nem lehet, ezért a legegyszerűbb, ha 2-től $n-1$-ig végignézzük az összes számot, hogy osztja-e n-et. Ez azonban nagy n-ekre nagyon sokáig tart, ezért célszerű javítani a módszert.

Először is, nem kell $n-1$-ig menni, elég $n/2$-ig (pontosabban $\lfloor n/2 \rfloor$-ig), hiszen annál nagyobb valódi osztója úgysem lehet az n-nek. Ezzel máris megfeleztük a futási időt.

De valójában $n/2$-ig sem kell menni, elég $\sqrt{n}$-ig, ugyanis minden k osztónak van egy n/k párja, amely szintén osztó, és ha $k > \sqrt{n}$, akkor $n/k < \sqrt{n}$, tehát ez utóbbit már meg kellett találnunk. Például $n = 100$ esetén elég 10-ig menni, mert ha osztója pl. a 20, akkor osztója a $100/20 = 5$ is, ami 10 alatt van. (És a gyökvonást megspórolhatjuk, ha a $k \leq \sqrt{n}$ feltételt négyzetre emeljük: $k^2 \leq n$.)

A próbaosztás algoritmusa tehát:

Prímszám-e(n):
- **Bemenet:** $n \in \mathbb{N}$
- **ha** $n < 2 \to$ *nem prímszám*
- **ciklus** $k := 2$-*től* **amíg** $k^2 \leq n$
  - **ha** $k \mid n$
    - $\to$ *nem prímszám*
- $\to$ *prímszám*

Ezt így implementálhatjuk Sage-ben:

```
def primszam (n):
    if n < 2: return False
    k = 2
    while k*k <= n:
        if n % k == 0:
            return False
        k += 1
    return True
```

Az algoritmust lehet még tovább javítani. Például elég kettesével lépkedni: ha ugyanis a 2-t megvizsgáltuk, és az nem osztja az n-et, akkor utána már elég csak a páratlan számokat vizsgálni. Ezt a logikát továbbfejleszthetjük, ha a 2 és a 3 ellenőrzése után hatosával ($6 = 2 \cdot 3$) lépkedünk, és csak a $6k + 1$ és a $6k + 5$ alakú számokat vizsgáljuk, mert a többi úgyis osztható 2-vel vagy 3-mal.

3.2. Prímtényezős felbontás

Miért fontosak a prímszámok? Ezt leginkább a következő tétel világítja meg.

3.4. Tétel (A számelmélet alaptétele)**.** *Minden* 1*-nél nagyobb egész szám a sorrendtől eltekintve* ***egyértelműen*** *felírható prímszámok szorzataként.*

Példa. $360 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 5$.

3.5. Definíció. A fenti tételben szereplő felírást **prímtényezős felbontás**nak vagy **prímfelbontás**nak nevezzük [*prime factorization*].

A tétel kulcsfontosságú szava az "egyértelműen", azaz olyan nem fordulhat elő, hogy különböző prímfelbontással ugyanazt a számot kapjuk, pl. $2 \cdot 7 \neq 3 \cdot 5$. Az persze lehetséges, hogy ugyanazokat a prímtényezőket különböző sorrendben írjuk fel, pl. $30 = 2 \cdot 3 \cdot 5 = 5 \cdot 2 \cdot 3$, de ez a két felírás "lényegében" ugyanaz, erre utal a "sorrendtől eltekintve egyértelműen" kitétel. A teljes egyértelműséget például úgy érhetjük el, ha kikötjük, hogy a prímszámokat növekvő sorrendben kell felírni. Ekkor rövidítésként az egymás mellé kerülő azonos prímtényezőket összevonhatjuk hatvánnyá. Az így kapott felírásnak külön neve is van:

3.6. Definíció. Egy $n > 1$ egész szám **kanonikus alakján** [*canonical representation*] az egyértelműen létező $n = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$ szorzatot értjük, ahol $p_1 < p_2 < \ldots < p_m$ prímszámok, és a k_i kitevők pozitív egész számok.

Példa. 360 kanonikus alakja: $2^3 \cdot 3^2 \cdot 5$.

Ahogy fent írtuk, a kanonikus alak már teljesen egyértelmű.

A számelmélet alaptételéből megérthetjük, hogy miért fontosak a prímszámok: ezek ugyanis a pozitív egész számok "építőkockái", azaz minden 1-nél nagyobb egész szám ilyenek szorzatára bontható fel. (Megjegyzés: nem kivételek a prímszámok sem, amelyek *egyetlen* prímszám szorzatára, azaz önmagukra bonthatók fel, pl. $13 = 13$.)

A számelmélet alaptételéből azt is megérhetjük, hogy az 1-et miért nem tekintjük prímszámnak: az ugyanis nem valódi építőkocka, mert 1-gyel szorozva nem változik a szám, így ha azt is prímszámnak tekintenénk, azzal elveszne az egyértelműség, pl. $6 = 2 \cdot 3 = 1 \cdot 2 \cdot 3 = 1 \cdot 1 \cdot 2 \cdot 3$ stb.

Sage-ben a prímfelbontást a `factor()` függvénnyel állíthatjuk elő:

```
sage: factor(360)    # vagy: 360.factor()
2^3 * 3^2 * 5
```

A `factor()` visszatérési értéke egy `Factorization` típusú objektum, amely sorozatként is tud viselkedni, azaz pl. egy `for`-ciklussal felsorolható, ahol az elemek a (p_i, k_i) párok, például:

```
sage: for (p, k) in factor(360):
....:     print(f"prím: {p}, kitevő: {k}")
prím: 2, kitevő: 3
prím: 3, kitevő: 2
prím: 5, kitevő: 1
```

Megjegyzés: a prímfaktorizáció, azaz a prímfelbontás előállítása nagy számok esetén nagyon nehéz feladat. Olyannyira, hogy egyes kriptográfiai rendszerek (pl. az RSA, mint később látni fogjuk) azon alapulnak, hogy senki nem fogja tudni megtalálni a titkos prímtényezőit egy hatalmas számnak.

3.3. Eratoszthenész szitája

Hogyan tudjuk előállítani az összes prímszámot adott n-ig?

A naiv megoldás az lenne, hogy minden számról külön-külön eldöntjük próbaosztással (lásd fent), hogy prímszám-e. Ez azonban rengeteg felesleges munkát jelentene (például ha egy számról kiderült, hogy osztható 13-mal, akkor a következő 12 számra ezt már felesleges megnézni).

Ennél jobb megoldást kínál *Eratoszthenész szitája*: írjuk fel az egész számokat 2-től n-ig, majd szisztematikusan húzzuk ki az összetett számokat a már felfedezett prímszámok segítségével. Az első prímszám a 2, ezért húzzuk ki a többi páros számot, mert azok összetett számok. A következő prímszám a 3, ezért húzzuk ki a többi 3-mal osztható számot. A 4-est már kihúztuk, tehát az nem prímszám. A következő érintetlen szám az 5-ös, tehát az prímszám, ezért húzzuk ki az összes többi 5-tel osztható számot, és így tovább. Végül pontosan a prímszámok maradnak.

Például így néz ki a szita első néhány lépése $n = 25$-ig:

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~	11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~	21	~~22~~	23	~~24~~	25
2	**3**	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~	25
2	3	~~4~~	**5**	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~	~~25~~

Néhány észrevétel, amellyel gyorsítani lehet az algoritmust:

1. Itt is elég csak $\sqrt{n}$-ig menni a prímekkel, hiszen az annál nagyobb prímszámok többszöröseit már kihúztuk valamelyik kisebb prímszámmal. (Például fent valóban elég volt 5-ig menni.

A 7 többszöröseit hiába néznénk, azzal már nem tudnánk újabb számot kihúzni: a 14-et már kihúztuk a 2-vel, a 21-et pedig a 3-mal.)

2. Egy adott p prímszámnál nem kell $2p$-től kezdeni az áthúzást, hanem elég p^2-től. Miért? Mert a p és p^2 közötti p-többszörösöknek volt kisebb prím osztója, így azokat már kihúztuk. (Például az 5-tel a 15-öt nem kellett kihúzni, mert azt a 3-mal már megtettük.)

Tehát így néz ki az algoritmus:

Eratoszthenész szitája(n):

Bemenet: $n \in \mathbb{N}^+$
Kimenet: P, egy logikai értékekből (*igaz*/*hamis*) álló lista 0-tól n-ig, ahol $P[i]$ jelzi, hogy i prímszám-e.
P-ben kezdetben minden *igaz*, kivéve $P[0] = P[1] = hamis$.
ciklus $i := 2$-*től* **amíg** $i^2 \leq n$
  ha $P[i]$
    ciklus $j := i^2$-*től* i-*esével* (i^2, $i^2 + i$, $i^2 + 2i$, ...) **amíg** $j \leq n$
      $P[j] := hamis$

(Megjegyzés: bár P-nek elég lenne 2-től indulnia, de az egyszerűség kedvéért 0-tól indul, hogy Sage-ben könnyebb legyen implementálni.)

Bebizonyítható, hogy az algoritmus futási ideje nagyságrendileg $n \log\log n$ lépés, amely nagy n-ekre sokkal gyorsabb, mint a prímek egyenkénti vizsgálata, amely $n\sqrt{n}$ lépést igényel.

4. Osztók száma és összege

Vizsgáljunk meg még néhány érdekes fogalmat az osztókkal kapcsolatban. Például:

4.1. Definíció. Jelölje $\boldsymbol{\tau(n)}$ az $n \in \mathbb{N}^+$ pozitív osztóinak számát. (τ kiejtése: "tau".)

Példa. $\tau(6) = 4$, mert $n = 6$-nak 4 pozitív osztója van: 1, 2, 3 és 6.

Írassuk ki a τ függvény első néhány értékét (Sage-ben `number_of_divisors()` a neve):

```
sage: matrix([[n, number_of_divisors(n)] for n in range(1, 26)]).transpose()
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]
[ 1  2  2  3  2  4  2  4  3  4  2  6  2  4  4  5  2  6  2  6  4  4  2  8  3]
```

Az osztószám-függvény jól elkülöníti a prímszámokat, az összetett számokat és az 1-et:

- $\tau(1) = 1$,
- $\tau(n) = 2$, ha n prímszám, és
- $\tau(n) \geq 3$, ha n összetett szám.

Hogyan tudjuk kiszámítani az osztószám-függvényt? A legegyszerűbben (és a legkevésbé hatékonyan) a definíció alapján, azaz hogy megszámoljuk az osztóit. Ha viszont ismerjük a szám prímtényezős felbontását, akkor hatékonyabb megoldást is tudunk adni:

4.2. Tétel (τ kiszámítása)**.** *Ha n kanonikus alakja $n = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$, akkor*

$$\tau(n) = (k_1 + 1)(k_2 + 1) \cdots (k_m + 1).$$

Példa. $360 = 2^3 \cdot 3^2 \cdot 5$, ezért $\tau(360) = (3 + 1)(2 + 1)(1 + 1) = 24$.

(Bizonyítás: az n bármely osztójában csak ugyanazok a prímtényezők lehetnek, mint n-ben, és legfeljebb ugyanakkora hatványon. Ezért az osztók $p_1^{l_1}p_2^{l_2}\cdots p_m^{l_m}$ alakúak, ahol a p_i-k ugyanazok, mint n-ben, és minden l_i kitevő 0 és k_i között lehet, ami (k_i+1)-féle lehetőség. Hány ilyen osztó létezik? Ahányféleképpen kiválaszthatjuk az l_i-ket, azaz $(k_1+1)(k_2+1)\cdots(k_m+1)$. De honnan tudjuk, hogy ezek tényleg mind különböző osztók lesznek, azaz nem számoltuk semelyik osztót sem többször? A számelmélet alaptételéből, amely szerint a prímtényezős felbontás egyértelmű.)

Most nézzünk egy másik nevezetes számelméleti függvényt:

4.3. Definíció. Jelölje $\boldsymbol{\sigma(n)}$ az $n \in \mathbb{N}^+$ pozitív osztóinak összegét. (σ kiejtése: "szigma".)

Példa. $\sigma(6) = 1+2+3+6 = 12$.

A σ függvény első néhány értéke:

```
sage: matrix([[n, sigma(n)] for n in range(1, 26)]).transpose()
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]
[ 1  3  4  7  6 12  8 15 13 18 12 28 14 24 24 31 18 39 20 42 32 36 24 60 31]
```

Ezt is ki lehet számítani a prímfelbontásból:

4.4. Tétel (σ kiszámítása)**.** *Ha n kanonikus alakja $n = p_1^{k_1}p_2^{k_2}\cdots p_m^{k_m}$, akkor*

$$\sigma(n) = \frac{p_1^{k_1+1}-1}{p_1-1}\cdot\frac{p_2^{k_2+1}-1}{p_2-1}\cdot\ldots\cdot\frac{p_m^{k_m+1}-1}{p_m-1}.$$

Néha használjuk a $\sigma(n)$ egy változatát, ahol az n-et magát kihagyjuk az osztók közül:

4.5. Definíció. Jelölje $\boldsymbol{s(n)}$ az $n \in \mathbb{N}^+$ valódi osztóinak összegét [*aliquot sum*].

Más szóval: $s(n) := \sigma(n) - n$.

Ezen függvény segítségével definiálhatunk néhány érdekes számtulajdonságot:

4.6. Definíció. $n \in \mathbb{N}^+$ **tökéletes szám** [*perfect number*], ha $s(n) = n$.

Példa. 6 tökéletes szám, mert $s(6) = 1+2+3 = 6$.

A nem tökéletes számokat két csoportba oszthatjuk:

4.7. Definíció. $n \in \mathbb{N}^+$ **bővelkedő szám** [*abundant number*], ha $s(n) > n$, és **hiányos szám** [*deficient number*], ha $s(n) < n$.

A tökéletes számokhoz kapcsolódik még egy fogalom:

4.8. Definíció. $a \in \mathbb{N}^+$ és $b \in \mathbb{N}^+$ **barátságos számpár** [*amicable numbers*], ha $s(a) = b$ és $s(b) = a$, de $a \neq b$.

Példa. 220 és 284 barátságos számpár, mert $s(220) = 1+2+4+5+10+11+20+22+44+55+110 = 284$ és $s(284) = 1+2+4+71+142 = 220$.

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_division

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

[4] https://en.wikipedia.org/wiki/Perfect_number

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