Diszkrét matematika
5. előadás

MÁRTON Gyöngyvér
mgyongyi@ms.sapientia.ro

Sapientia Egyetem,
Matematika-Informatika Tanszék
Marosvásárhely, Románia

2023, őszi félév

Miről volt szó az elmúlt előadáson?

- legnagyobb közös osztó,
- a kiterjesztett euklideszi algoritmus,
- diofantoszi egyenletek,
- a `strip`, `split`, `zip`, `enumerate` függvények,

Miről lesz szó?

- a `zip`, az `enumerate` függvények, a `*` operátor,
- szövegállományok,
- racionális számok: sorba rendezés (két módszer),
- lánctörtek,
- "híresebb" irracionális számok tízedesjegyei: $\sqrt{n}, \pi, e, \phi$,

A zip és a enumerate függvények

```
>>> hL =  ['januar','februar','marcius','aprilis']
>>> homL = [-3, -1, 3, 5]
>>> for h1, h2 in zip(hL, homL): # osszecipzarozzuk a ket listat
        print(h1, h2)

    januar -3
    februar -1
    marcius 3
    aprilis 5

>>> for i, h1 in enumerate(hL):
        print(i, h1)

    0 januar
    ...

>>> for i, (h1, h2) in enumerate(zip(hL, homL)):
        print(i, h1, h2)

    0 januar -3
    ...
```

A * operátor

Ha vissza akarjuk állítani a cipzározás előtti állapotot, akkor használjuk a * operátort:

```
>>> L = [('januar', -3), ('februar', -1), ('marcius', 3), ('aprilis', 5)]

>>> h1, h2 = zip(*L)

>>> h1
    ('januar', 'februar', 'marcius', 'aprilis')
>>> h2
    (-3, -1, 3, 5)
```

Pythonban a * operátor használata sokrétű, bővebben itt lehet utána olvasni: *link*

Szövegállományok, az adatok beolvasása

1. feladat

A `homerseklet.txt` a hónapok neveit, illetve a hónapokhoz tartózó hőmérsékleti értéket tárolja. Írjunk egy Python függvényt, amely beolvassa az adatokat az állományból, és meghatározza, hogy mely hónapokban volt negatív a hőmérséklet.

A `homerseklet.txt` tartalma legyen a következő, ahol minden hónapnév külön sorban van megadva, úgy hogy a hónapok, és a hónaphoz tartozó hőmérsékleti értékek között egy *szóköz* van:

```
januar -7
februar -5
marcius -1
aprilis 4
majus 8
junius 10
julius 12
augusztus 12
szeptember 9
oktober 4
november -1
december -5
```

Szövegállományok, az adatok beolvasása

- az állományban levő adatok feldolgozásához szükséges az állományt először megnyitni: `open`, majd a végén bezárni: `close`
- a `beolvas` során egy, tuple elemtípusú listát hozunk létre, ahol a tuple első eleme a hónapnevet, második eleme pedig a hőmérsékleti értéket jelöli
- az adatok beolvasása soronként történik: `readline`, ahol a beolvasott érték típusa mindig `str`, szükség esetén ezt át kell alakítani `Int`, `Float` stb. típusúvá

```
def beolvas():
    inf = open('homerseklet.txt', 'rt')
    L = []
    while True:
        temp = inf.readline()
        if not temp: break
        temp = temp.strip('\n')
        honap, homerseklet = temp.split(' ')
        L += [(honap, int(homerseklet))]
    inf.close()
    return L
```

Szövegállományok, az adatok beolvasása

```
def beolvas_():
    with open('homerseklet.txt', 'r') as inf:
        L = []
        while True:
            temp = inf.readline()
            if not temp: break
            temp = temp.strip('\n')
            honap, homerseklet = temp.split(' ')
            L += [(honap, int(homerseklet))]
    return L
```

Szövegállományok, az adatok beolvasása

- az adatok feldolgozását végezhetjük a `homerseklet1` függvénnyel:

```
def homerseklet1(honapL, homersekletL):
    resL = []
    for i in range(12):
        if homersekletL[i] < 0:
            resL += [honapL[i]]
    return resL

if __name__ == "__main__":
    L = beolvas_()
    hL, homL = zip(*L)
    resL = homerseklet1(hL, homL)
    print(resL)
```

Racionális számok

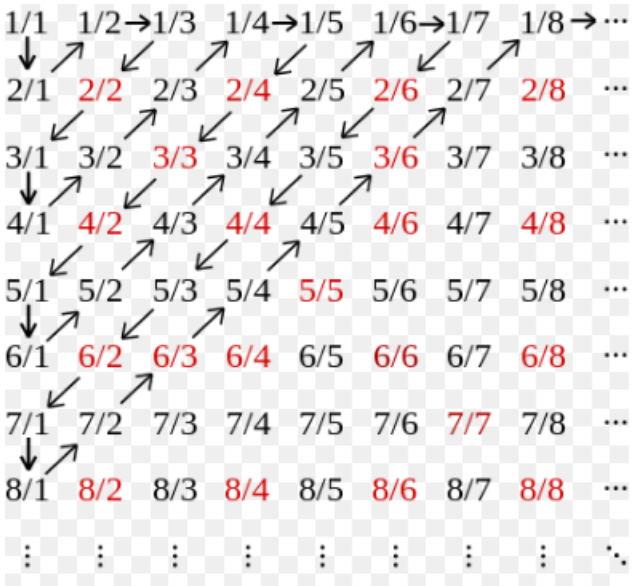
A racionális számok halmazát: $\mathbb{Q}$-val jelöljük. Bármelyik eleme felírható: $\frac{a}{b}$ alakba, ahol $a, b \in \mathbb{Z}, b \neq 0$. Tekinthetünk úgy is rájuk, mint egész számok rendezett párjaira. Tulajdonságok:

- az összeadás, szorzás kommutatív, asszociatív műveletek,
- az összeadás a szorzásra nézve disztributív művelet,
- a racionális számok halmaza zárt az összeadásra, kivonásra, szorzásra, osztásra nézve,
- az összeadásra, szorzásra nézve minden elemnek lesz inverz eleme,
- a racionális számok halmaza megszámlálható, azaz létezik egy számsorozat amely a racionális számokból áll,
- a racionális számok sűrűn rendezett halmazt alkotnak: bármely két racionális szám között van egy harmadik,
- minden racionális szám felírható tizedes alakba, azaz véges, vagy végtelen szakaszos tizedes jegyek segítségével.

Racionális számok sorba rendezése

A racionális számok halmaza megszámlálható, sorba rendezhető:

- létezik egy számsorozat, amelyet a racionális számok alkotnak : $r_1, r_2, \ldots, r_n, \ldots,$
- az első módszer szerint elindulunk a következő mátrix bal-felső sarkában található elemtől, majd a nyilakat követjük,
- pirossal azokat a számok jelennek meg, amelyek nem irreducibilis alakban vannak, korábban pedig már ki lettek generálva.



Racionális számok sorba rendezése

A második módszer szerint a következőképpen járunk el:

- az első racionális szám $\frac{1}{1}$,
- az $\frac{x}{y}$ után következő racionális számot, ahol $\lfloor\ \rfloor$ alsó egészrészt jelent a következő racionális szám reciproka adja:

$$2 \cdot \left\lfloor \frac{x}{y} \right\rfloor + 1 - \frac{x}{y} \tag{1}$$

- az $\left\lfloor \frac{x}{y} \right\rfloor$ értéket // művelettel kell meghatározni, azaz osztási egészrészt kell számolni,
- a különbség valódi törtek, azaz *tuple elemek* közötti különbséget jelent.

pl: $\frac{5}{2} \to 2 \cdot \left\lfloor \frac{5}{2} \right\rfloor + 1 - \frac{5}{2} = 2 \cdot 2 + 1 - \frac{5}{2} = 5 - \frac{5}{2} = \frac{10-5}{2} = \frac{5}{2} \to \frac{2}{5}$

pl: $\frac{5}{3} \to 2 \cdot \left\lfloor \frac{5}{3} \right\rfloor + 1 - \frac{5}{3} = 2 \cdot 1 + 1 - \frac{5}{3} = \frac{9-5}{3} = \frac{4}{3} \to \frac{3}{4}$

Ezzel a módszerrel a következő törteket kapjuk:
$\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{3}{2}, \frac{2}{3}, \frac{3}{1}, \frac{1}{4}, \frac{4}{3}, \frac{3}{5}, \frac{5}{2}, \frac{2}{5}, \frac{5}{3}, \frac{3}{4}, \frac{4}{1}$, *stb.*

A fractions modul

A Python `Fraction` típusa: használatához szükséges importálni a `fractions` modult:

```
from fractions import Fraction
>>> rac1 = Fraction(2,3)
>>> rac2 = Fraction(4,5)

>>> rac1 + rac2
    Fraction(22, 15)

>>> rac1 * rac2
    Fraction(8, 15)

>>> print('szamlalo: ', rac1.numerator)
    szamlalo:  2
>>> print('nevezo: ', rac1.denominator)
    nevezo:  3
```

A fractions modul

2. feladat

Írjunk egy Python függvényt, amely meghatározza az $\frac{x}{y}$ utáni racionális számot, alkalmazva a (1) képletet.

```
from fractions import Fraction
def nextRacFrac(r):
    x, y = r.numerator, r.denominator
    temp = 2 * (x // y) + 1
    res = Fraction(temp, 1) - Fraction(x, y)
    return Fraction(res.denominator,  res.numerator)

>>> racNr = Fraction(4,3)
>>> nextRacFrac(racNr)
    Fraction(3, 5)
```

Lánctörtek (Continued fraction)

A lánctört egy *emeletes* tört, amely kétféle alakban is megadható, ahol a két alak átalakítható egymásba:

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \cfrac{b_4}{\ddots}}}} \qquad d_0 + \cfrac{1}{d_1 + \cfrac{1}{d_2 + \cfrac{1}{d_3 + \cfrac{1}{\ddots}}}}$$

A második alakot egyszerű lánctörtnek, a $[d_0, d_1, d_2, d_3, \ldots]$ számsorozatot, pedig a lánctört jegyeinek hívjuk.

Ha a $[d_0, d_1, d_2, d_3, \ldots]$ számsorozat véges számú elemet tartalmaz, akkor véges lánctörtről beszélünk.

A racionális számok mindegyike felírható egyszerű, véges lánctört alakba.

Lánctörtek, példa

Az $\frac{x}{y}$ racionális szám lánctört jegyeinek a meghatározásához a következőképpen járunk el:

- meghatározzuk az `x` és `y` osztási egészrészét (//), illetve osztási maradékát (%),
- felülírjuk az `x` értékét `y`-al és az `y` értékét az osztási maradékkal,
- addig ismételjük a műveletsort, amíg az `x` és `y` osztási maradéka nem lesz 0,
- a legnagyobb közös osztó algoritmusának gondolatmenetét követjük,
- eltároljuk az osztási egészrészeket, ezek fogják képezni a lánctörtjegyeket.

Példa, $\frac{61}{47}$ lánctört jegyeinek a meghatározása:

x	y	$x//y$	$x\%y$
61	47	1	14
47	14	3	5
14	5	2	4
5	4	1	1
4	1	4	0

Lánctörtek, példa

$$\frac{61}{47} = 1 + \cfrac{1}{3 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{4}}}}$$

$$\frac{61}{47} = 1 + \cfrac{1}{3 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{3 + \cfrac{1}{1}}}}}$$

$\frac{61}{47}$ lánctört jegyei: $[1, 3, 2, 1, 4]$, vagy

$\frac{61}{47}$ lánctört jegyei: $[1, 3, 2, 1, 3, 1]$

$\frac{61}{47}$ tizedes alakja: $1.(29787234042553191489361702127659574468085106 38)$

Ha a lánctört utolsó jegye nem 1, akkor ez az x érték helyettesíthető két további értékkel: $x - 1, 1$. A két alak ekvivalens.

Lánctörtek, példa

Alakítsuk át $\frac{41}{11}$-t lánctörtté, hat. meg a lánctört jegyeket, és a tizedes alakot:

$$\frac{41}{11} = 3 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{2}}}}$$

$$\frac{41}{11} = 3 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1}}}}$$

$\frac{41}{11}$ lánctört jegyei: $[3, 1, 2, 1, 2]$, vagy

$\frac{41}{11}$ lánctört jegyei: $[3, 1, 2, 1, 1, 1]$

$\frac{41}{11}$ tizedes alakja: $3.(72)$

Irracionális számok

- halmazjelölés: $\mathbb{Q}^*$,
- azok a számok, melyek nem írhatóak fel két egész szám hányadosaként, azaz a végtelen, nem szakaszos tizedes törtek,
- "híresebb" irracionális számok:

$$\begin{aligned} \sqrt{2} &= 1.4142\ldots, \\ \pi &= 3.1415\ldots, \\ e &= 2.7182\ldots, \\ \phi &= \frac{1+\sqrt{5}}{2} = 1.6118\ldots, \text{az aranyarány.} \end{aligned}$$

- a számítástechnika az irracionális számok közelített értékét tudja kezelni
- az irracionális számok végtelen lánctörtek
- lánctörtek segítségével könnyedén meglehet határozni a közelített érték tízedesjegyeit

$\sqrt{n}$ értéke

3. feladat

Határozzuk meg $\sqrt{n}$ értékét lánctörtek segítségével.

A kiinduló képlet a következő, ahol a értéke egy akármilyen szám lehet:

$$\sqrt{n} = a + \frac{n - a^2}{a + \sqrt{n}}$$

ha $a = 1$, akkor: $\sqrt{n} = 1 + \cfrac{n-1}{2 + \cfrac{n-1}{2 + \cfrac{n-1}{2 + \cfrac{n-1}{2 + \dots}}}}$

$\sqrt{2}$ értéke

4. feladat

Határozzuk meg $\sqrt{2}$ értékét lánctörtek segítségével.

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ldots}}}$$

```
def my_sqrt2_():
    temp = 0
    for x in range(0, 15):
        temp = 1 / (2 + temp)
    return 1 + temp

>>> my_sqrt2_()
    1.4142135623730951
>>> 2 ** 0.5
    1.4142135623730951
```

A decimal modul

- a Python bevezeti a `Decimal` típust, amely pontosabb számolást tesz lehetővé, a nem egész számok körében,
- A `Decimal` típus a `decimal` modulban van definiálva, és a felhasználó által óhajtott pontossággal ábrázolja a lebegőpontos (valós) számokat

```
>>> from decimal import Decimal
>>> (Decimal(10)**6).log10()
    Decimal('6')

>>> Decimal(1)/3
    Decimal('0.33333333333333333333333333333')
```

A `getcontext.prec()` segítségével a tizedes jegyek számát adhatjuk meg.

```
>>> from decimal import getcontext
>>> getcontext().prec = 50
>>> from math import ceil
>>> temp = 10**(1/Decimal('3'))
>>> temp
>>> ceil(temp *temp * temp)
```

$\sqrt{2}$ értéke

Ha több tizedes jegyet szeretnénk, akkor a `Decimal` típussal és a `getcontext` metódussal kell dolgozzunk.

```
from decimal import Decimal, getcontext
def my_sqrt2(p):
    getcontext().prec = p
    temp = Decimal(0)
    for x in range(0, 500):
        temp = 1 / (2 + temp)
    return 1 + temp

>>> my_sqrt2(30)
    Decimal('1.41421356237309504880168872421')

>>> my_sqrt2(50)
    Decimal('1.41421356237309504880168872420969807856967187537694')
```

A π szám

- a kör kerületének és átmérőjének hányadosa az eukleidészi geometriában:

$$\pi = \frac{K}{2 \cdot R},$$

  ahol R a kör sugara, K a kör kerülete
- más definíciók is léteznek, melyek kihagyják a kört
- irracionális és transzcendens szám (nincs olyan egész együtthatós polinom amelynek gyöke lenne)

A π értékének meghatározására több lánctört-képlet is ismert:

$$\pi = \cfrac{4}{1 + \cfrac{1^2}{3 + \cfrac{2^2}{5 + \cfrac{3^2}{7 + \ldots}}}}$$

$$\pi = 3 + \cfrac{1}{6 + \cfrac{3^2}{6 + \cfrac{5^2}{6 + \cfrac{7^2}{6 + \ldots}}}}$$

A π szám

5. feladat

Határozzuk meg π értékét lánctörtek segítségével.

```
def my_pi(p):
    getcontext().prec = p
    temp = Decimal(0)
    for x in range(1000, 0, -1):
        a = x * x
        b = 2 * x + 1
        temp = a / (b + temp)
    return 4 / (1 + temp)

>>> my_pi(100)
    Decimal('3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 7')
```

Az e szám

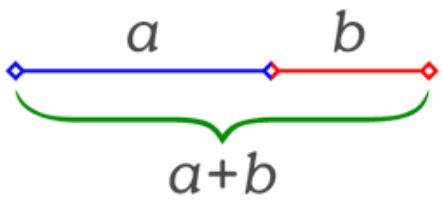
- irracionális és transzcendens szám
- többféleképpen lehet értelmezni:

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n.$$

$$e = 2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{2}{3 + \cfrac{3}{4 + \ldots}}}}$$

A φ szám, aranymetszés, aranyarány (Golden Ratio)

- két mennyiség, $a, b, a > b$ az aranymetszés szerint aránylik egymáshoz, ha fennáll:



$$\varphi \stackrel{def}{=} \frac{a}{b} = \frac{a+b}{a}$$

- a φ meghatározása érdekében felírhatjuk: $\frac{a+b}{a} = 1 + \frac{1}{\frac{a}{b}}$, azaz fennáll:

$$\varphi = 1 + \frac{1}{\varphi} \Leftrightarrow \varphi^2 = \varphi + 1$$

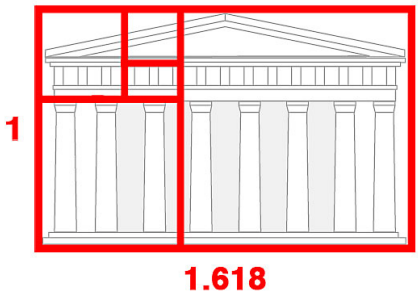
- megoldva a fenti egyenletet kapjuk, hogy

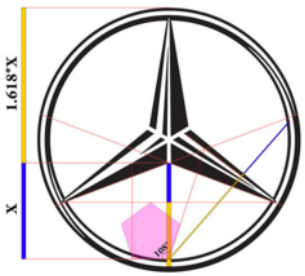
$$\varphi = \frac{1+\sqrt{5}}{2} = 1.61803\ldots \text{ és } \hat{\varphi} = \frac{1-\sqrt{5}}{2} = -0.61803\ldots$$

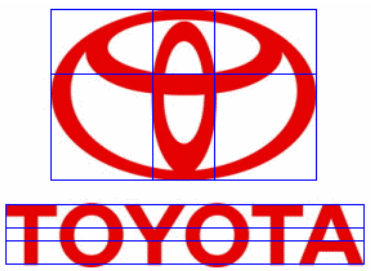
- a φ irracionális szám

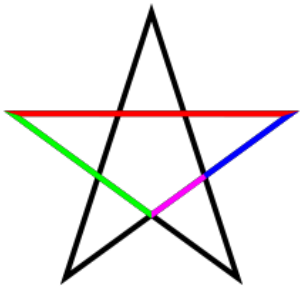
A φ szám, aranymetszés, aranyarány (Golden Ratio)

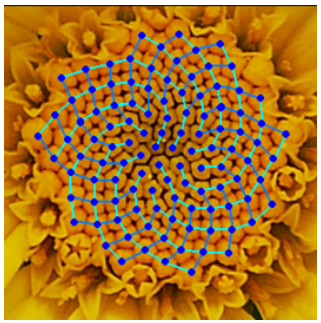
- építészet: Parthenon homlokzatának arányértékei:
- logok: Toyota, Mercedesz, stb.
- Pentagramma (szabályos ötszög): $\frac{piros}{zold} = \frac{zold}{kek} = \frac{kek}{lila} = \varphi$
- természet: napraforgó spiráljai











A φ szám, aranymetszés, aranyarány (Golden Ratio)

Kiindulva a $\varphi = 1 + \frac{1}{\varphi}$ összefüggésből, a φ értékét felírhatjuk a következőképpen:

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\ddots}}}}$$

A φ szám, aranymetszés, aranyarány (Golden Ratio)

- két egymás utáni Fibonacci szám arányaként is felírhatjuk a φ értékét
- egy Fibonacci szám a Fibonacci számsorozat egy eleme
- a Fibonacci számsorozat: $0, 1, 1, 2, 3, 5, 8, 13...$, kiindulva a 0, 1 kezdőértékekből, a következő elemet az előző két elem összegéből kapjuk

Az `n` és `m` közötti, két egymásutáni Fibonacci számból képzett arányok meghatározása:

```
def fib_phi(n, m):
    f1 = 0
    f2 = 1
    for i in range(2, m):
        f = f1 + f2
        if i >= n:
            print("%4d%5d%5d%10.6f" % (i, f, f2, f/f2))
        f1 = f2
        f2 = f

>>> fib_phi(10, 14)
   10    55    34   1.617647
   11    89    55   1.618182
   12   144    89   1.617978
   13   233   144   1.618056
```