

Diszkrét matematika

6. előadás

MÁRTON Gyöngyvér
mgyongyi@ms.sapientia.ro

Sapientia Egyetem,
Matematika-Informatika Tanszék
Marosvásárhely, Románia

2024, őszi félév



Miről volt szó az elmúlt előadáson?

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

Miről lesz szó?

- valós számok, komplex számok,
- valós számok lánc tört jegyei,
- polinom helyettesítési értéke, függvények ábrázolása,
- a \sin , \ln lánc tört képlete,
- a `numpy` és a `matplotlib` modulok,
- a `complex` típus,
- műveletek komplex számokkal: szorzat, hatványozás, stb
- fraktálok: Mandelbrot, Julia,
- számrendszerek
 - egész szám alapú számrendszerek,
 - a számrendszerek közötti kapcsolat,

Valós számok

- a racionális és irracionális számok halmaza,
- halmazjelölés: $\mathbb{R} = \mathbb{Q} \cup \mathbb{Q}^*$,
- egy szám egyszerre **nem lehet racionális és irracionális is**,
- a valós számokhoz hozzárendelhető, egy mindkét irányban végtelen egyenes egy-egy pontja,
- kommutatívítás, asszociatívítás, disztributívítás,
- az egész számokkal ellentétben a valós számok halmaza **nem megszámlálható**,
- a számítástechnikában nem valós mennyiségekkel dolgozunk, ezek egy közelítő értéke lesz eltárolva: **lebegőpontos ábrázolás**

Lánctörtjegyek

1. feladat

Írjunk egy Python függvényt, amely a bemenetként megadott valós szám tizedes alakja alapján meghatározza a megfelelő egyszerű lánctörtjegyeket.

```
def lanctValos(nr):  
    L = []  
    for i in range(len(str(nr))):  
        eR = int(nr)  
        L += [eR]  
        tR = nr - eR  
        if tR == 0: break  
        nr = 1 / tR  
    return L
```

```
>>> from math import pi  
>>> lanctValos(pi)  
[3,7,15,1,292,1,1,1,2,1,3,1,14,3,3,23,1]  
>>> lanctValos(1.41421356237309504880168872420969)  
[1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
```

Polinom helyettesítési értéke

2. feladat

Írjunk egy Python függvényt, amely meghatározza egy adott polinom helyettesítési értékét, egy megadott értékre.

- két algoritmust adunk meg, ahol a horner hatékonyabb lesz, mint a poly,
- mindkét függvénynek két bemeneti értéket kell megadni, az első a *megadott* érték, a második egy lista, amelyben a polinom együtthatóit kell megadni, ahol a lista első eleme a polinom legnagyobb fokszámú eleme kell legyen. Ha $A(c) = a_n \cdot c^n + \dots + a_1 \cdot c + a_0$, akkor a lista = $[a_n, a_{n-1}, \dots, a_1, a_0]$.

```
def poly(c, A):  
    res, p = 0, 1  
    for a in A[::-1]:  
        res += a * p  
        p *= c  
    return res  
  
res, p = 0, 1  
res, p = a0 * 1, c  
res, p = a1 * c + a0 * 1, c2  
res, p = a2 * c2 + a1 * c + a0 * 1, c3  
...
```

```
def horner(c, A):  
    res = 0  
    for a in A:  
        res = res * c + a  
    return res  
  
res = 0  
res = 0 * c + an = an  
res = an * c + an-1  
res = (an * c + an-1) * c + an-2  
    = an * c2 + an-1 * c + an-2  
...
```

A numpy és matplotlib csomagok

A függvények/polinomok ábrázolásához szükség lesz a **numpy** és **matplotlib** csomagokra, amelyek telepítésékor, illetve egyéb Python csomagok telepítésekor használjuk a pip telepítőt. A Python újabb verziói alapból tartalmazzák a pip-et.

- Windows alatti egyszerűbb használatához, be kell legyen állítva a PATH környezetváltozó (lásd részletesebben az 1. előadásban)
- adjuk ki a következő parancsokat:
 - `python -m pip install numpy`
 - `python -m pip install matplotlib`
 - `python -m pip install matplotlib -user`

Ha nem ismeri fel a pip-et, akkor a csomagok telepítése előtt azt is telepíteni kell:

- töltsük le egy mappába a *get-pip.py* állományt:
<https://bootstrap.pypa.io/3.2/get-pip.py>
- nyissunk meg egy Command prompt ablakot, válasszuk ki azt a mappát amibe a get-pip állományt tettük, pl: `cd C:\Downloads`
- adjuk ki a következő parancsokat:
 - `python get-pip.py`
 - `python -m install --upgrade pip`

Polinomok/függvények ábrázolása

3. feladat

Írjunk egy Python függvényt, amely grafikusán ábrázolja a megadott függvényt/polinomot.

```
import numpy as np
import matplotlib.pyplot as plt

def main():
    X = np.linspace(-200, 200, 300, endpoint=True)
    F1 = horner(X, [1, -4, 3])
    F2 = horner(X, [2, 7, 7])
    #F1 = horner(X, [4, 11, 3, 5])
    #F2 = horner(X, [10, 3, 11, 19])
    plt.plot(X, F1, label = "F1")
    plt.plot(X, F2, label = "F2")

    plt.axvline(x = 0, c = "lightgray")
    plt.axhline(y = 0, c = "lightgray")
    plt.show()

main()
```


A $\sin(z)$, $\ln(z)$ lánc törképlete

$$\sin(z) = \frac{z}{1 + \frac{z^2}{2 \cdot 3 - z^2 + \frac{2 \cdot 3 \cdot z^2}{4 \cdot 5 - z^2 + \frac{4 \cdot 5 \cdot z^2}{6 \cdot 7 - z^2 + \dots}}}}$$

$$\ln(1+z) = \frac{z}{1 + \frac{1^2 \cdot z}{2 + \frac{1^2 \cdot z}{3 + \frac{2^2 \cdot z}{4 + \frac{2^2 \cdot z}{5 + \frac{3^2 \cdot z}{6 + \dots}}}}}}$$

Polinomok/függvények ábrázolása

```
def my_sinF(z):  
    temp = 0  
    for x in range(30, 0, -2):  
        a = (x + 2) * (x + 3) - z * z  
        b = x * (x + 1) * z * z  
        temp = b / (a + temp)  
    temp = z * z / (2 * 3 - z * z + temp)  
    return z / (1 + temp)  
  
def main2():  
    X = np.arange(0, 3*np.pi, 0.1)  
    #plt.plot(X, np.sin(X))  
    plt.plot(X, my_sinF(X))  
    plt.axvline(x = 0, c = "red")  
    plt.axhline(y = 0, c = "red")  
    plt.show()  
  
main2()
```

Komplex számok

A komplex számok a valós számhalmaz egy olyan bővítése, melyben negatív számok esetén is értelmezett a gyökvonás, halmazjelölés: \mathbb{C} .

- három modell alapján is értelmezhető: **halmazelméleti, geometriai, algebrai** modell alapján
- halmazelméleti modell: $\mathbb{C} = \{(a, b) | a \in \mathbb{R}, b \in \mathbb{R}\}$, azaz a számhalmazt rendezett számpárok alkotják, ahol az elemek valós számok,
- imaginárius rész: az a komplex szám amelynek **négyzete -1**, jele az **i**
- a komplex számok $a + b \cdot i$ alakban írhatóak fel, ahol **a** valós rész, **b** imaginárius egység
- ha $b = 0$, akkor valós számot kapunk
- a valós számok körében megismert műveleti tulajdonságok megmaradnak
- additív semleges elem: $z = 0 + 0 \cdot i$
- multiplikatív semleges elem: $z = 1 + 0 \cdot i$
- additív inverz elem: $-z = -a - b \cdot i$,
- multiplikatív inverz elem: $1/z = a/(a^2 + b^2) - b/(a^2 + b^2) \cdot i, z \neq 0$

Műveletek komplex számokkal

$$a = a_1 + a_2 \cdot i$$

$$\text{abs}(a) = \sqrt{a_1^2 + a_2^2}$$

$$b = b_1 + b_2 \cdot i$$

$$a + b = (a_1 + b_1) + (a_2 + b_2) \cdot i$$

$$a - b = (a_1 - b_1) + (a_2 - b_2) \cdot i$$

$$a \cdot b = (a_1 \cdot b_1 - a_2 \cdot b_2) + (a_2 \cdot b_1 + a_1 \cdot b_2) \cdot i$$

$$\frac{1}{b} = \frac{b_1}{(b_1^2 + b_2^2)} - \frac{b_2}{(b_1^2 + b_2^2)} \cdot i$$

$$\frac{a}{b} = a \cdot \frac{1}{b}$$

```
>>> a = complex(2.5, 4.3)
>>> a.real
2.5
>>> a.imag
4.3
>>> abs(a)
4.973932046178355
>>> a ** 3.2
(-166.23299664047408-33.65354280875679j)

>>> b = complex(1.7,10.25)
>>> a + b
(4.2+14.55j)
>>> a - b
(0.8-5.95j)
>>> a * b
(-39.824999999999996+32.935j)
>>> a / b
(0.44765058706375493-0.1696579514138163j)
```

Két komplex szám szorzata

4. feladat

Írjunk egy Python függvényt, amely meghatározza az $a = a_1 + a_2i$ és $b = b_1 + b_2i$ komplex számok szorzatát, ahol alkalmazzuk az $a \cdot b = (a_1 \cdot b_1 - a_2 \cdot b_2) + (a_2 \cdot b_1 + a_1 \cdot b_2)i$ összefüggést.

```
def szorzatC(a, b):  
    a1, a2 = a  
    b1, b2 = b  
    vResz = a1 * b1 - a2 * b2  
    iResz = a2 * b1 + a1 * b2  
    return (vResz, iResz)  
  
>>> szorzatC((2.5, 5.4), (3.34, 101.5))  
(-539.75, 271.786)  
  
>>> a = complex(2.5, 5.4)  
>>> b = complex(3.34, 101.5)  
>>> a * b  
(-539.75+271.786j)
```

Fraktálok

- kiindulva egy **komplex számból**, egy iterációs folyamat eredményeként a képernyőre kirajzolt pontok fraktál alakzatokat hozhatnak létre
- minél nagyobb az iteráció szám, annál jobb a kirajzolt kép minősége
- a **Mandelbrot** fraktál:
 - a halmaz iterációs képlete:

$$\begin{aligned} z_0 &= c \\ z_{n+1} &= (z_n)^2 + c, \end{aligned}$$

ahol a c komplex szám értékét a programozó állítja be

- azokat a c komplex számokat fogja tartalmazni, amelyekre a z_n sorozat **nem tart a végtelenbe**,
- a z_n sorozat nem tart a végtelenbe, ha az i -edik iteráció után z_i **abszolútértéke kisebb vagy egyenlő lesz mint 2**.
- a **Julia fraktál** esetén halmaz iterációs képlete ugyanaz, mint a Mandelbrot halmazé, azzal a különbséggel, hogy a c értéke itt konstans.

Mandelbrot fraktál

5. feladat

Írjunk egy Python függvényt, amely meghatározza a Mandelbrot halmaz elemeit. Egy halmazbeli elem esetén írjunk #-t a képernyőre, másképp space-t.

```
def fMan():
    L1 = [a*0.07 for a in range(-15, 16)]
    L2 = [a*0.04 for a in range(-50, 26)]
    for y in L1:
        L = ""
        for x in L2:
            c = complex(x, y)
            z = c
            for i in range(40):
                z = z ** 2 + c
                if abs(z) > 2:
                    L += " "
                    break
            if abs(z) <= 2: L += "#"
    print (L)
```

Julia fraktál

6. feladat

Írjunk egy Python függvényt, amely meghatározza a Julia halmaz elemeit. Egy halmazbeli elem esetén írjunk # -t a képernyőre, másképp space-t. Legyen $c = -1 - 0.25i$

```
def fJulia():
    L1 = [a*0.07 for a in range(-15, 16)]
    L2 = [a*0.04 for a in range(-40, 36)]
    c = complex(-1, -0.25)
    for y in L1:
        L = ""
        for x in L2:
            z = complex(x, y)
            for i in range(40):
                z = z ** 2 + c
                if abs(z) > 2:
                    L += " "
                    break
            if abs(z) <= 2: L += "#"
    print (L)
```


Megjegyzések

- az L1, L2 intervallumokat módosíthatjuk, a kirajzolt alakzat nagyobb lesz:
L1 = [a*0.05 for a in range(-20, 21)]
L2 = [a*0.02 for a in range(-80, 31)]
- a **grafikus megjelenítéshez** használjuk a pygame csomagot, telepítéséhez nyissunk meg egy Command prompt ablakot, majd adjuk ki a következő parancsot:
python -m pip install pygame
- a grafikus megjelenítés *valamikori* forrása:
<http://fractalart.gallery/mandelbrot-set-in-python-pygame>
- a két fraktált grafikusán megjelenítő kódsorok a **4. labor** megoldott feladatai között találhatók

Számrendszerek

- egész számok: bármely 1-nél nagyobb számrendszerben ábrázolhatóak,
- a számítástechnikában gyakran használt számrendszerek: 10, 2, 8, 16, 256,
 - 2-es, bináris számrendszer, 2 szimbólum van: 0, 1,
 - 8-as, oktális számrendszer, 8 szimbólum van: 0, 1, 2, ..., 7,
 - 10-es, decimális számrendszer, 10 szimbólum van: 0, 1, 2, ..., 9,
 - 16-os hexadecimális számrendszer, 16 szimbólum van:
0, 1, 2, ..., 9, A, B, C, D, E, F,
 - 256-os számrendszer: 256 szimbólum van
- legyen nr az a szám, amit átszeretnénk írni b számrendszerbe, ekkor, \mathbb{Z}^* -al jelölve a nem negatív egész számok halmazát felírható:
$$nr = a_nb^n + a_{n-1}b^{n-1} + \dots + a_1b^1 + a_0b^0,$$
ahol $n \in \mathbb{Z}^*$, $a_i \in \mathbb{Z}^*$, és $a_i < b$, minden $i \in \{0, \dots, n\}$ értékre és $a_n \neq 0$,
- a nr szám, b számrendszerben felírt számjegyeit, a következő lista elemei alkotják: $[a_n, a_{n-1}, \dots, a_2, a_1]$

Számrendszerek, példák

- 2-es számrendszerben 2 szimbólum van: 0, 1,
- Pl. Mennyi $(215)_{10}$, 2-es számrendszerbeli alakja?
 - fennáll: $215 = 128 + 64 + 16 + 4 + 2 + 1 =$
 $1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0.$
 - tehát: $(215)_{10} = (1101\ 0111)_2.$
- Pl. $(1\ 0111\ 0010)_2$, melyik 10-es számrendszerbeli számnak felel meg?
 - fennáll:
 $1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 =$
 $256 + 64 + 32 + 16 + 2 = 370.$
 - tehát: $(1\ 0111\ 0010)_2 = (370)_{10}.$

Számrendszerek, példák

- 8-as számrendszerben 8 szimbólum van: 0, 1, 2, 3, 4, 5, 6, 7.
- Pl. Mi $(215)_{10}$, 8-as számrendszerbeli alakja?
 - fennáll: $215 = 3 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0$
 - tehát: $(215)_{10} = (327)_8$.
- Pl. $(6702)_8$, melyik 10-es számrendszerbeli számnak felel meg?
 - fennáll: $6702 = 6 \cdot 8^3 + 7 \cdot 8^2 + 0 \cdot 8^1 + 2 \cdot 8^0 = 3072 + 448 + 2 = 3522$.
 - tehát: $(6702)_8 = (3522)_{10}$.

Számrendszerek, példák

- 16-as számrendszerben 16 szimbólum van:
 $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$.
- Pl. Mi $(7432)_{10}$, 16-os számrendszerbeli alakja?
 - fennáll: $7432 = 1 \cdot 16^3 + 13 \cdot 16^2 + 0 \cdot 16^1 + 8 \cdot 16^0$
 - tehát: $(7432)_{10} = (1D08)_{16}$.
- Pl. $(E2D07)_{16}$, melyik 10-es számrendszerbeli számnak felel meg?
 - fennáll: $E2D07 = 14 \cdot 16^4 + 2 \cdot 16^3 + 13 \cdot 16^2 + 0 \cdot 16^1 + 7 \cdot 16^0 = 917504 + 8192 + 3328 + 7$.
 - tehát: $(E2D07)_{16} = (929031)_{10}$.

Átalakítás 10-es számrendszerből b számrendszerbe

- nr -et elosztjuk b -vel, legyen az osztási egészrész q_0 , a maradék a_0 , ekkor fennáll: $nr = b \cdot q_0 + a_0$, ahol $0 \leq a_0 < b$.
- q_0 -ot elosztjuk b -vel, legyen az osztási egészrész q_1 , a maradék a_1 , ekkor fennáll: $q_0 = b \cdot q_1 + a_1$, ahol $0 \leq a_1 < b$.
- addig folytatjuk az osztást, amíg 0 osztási egészrészt kapunk,
- a b számrendszerbeli számjegyeket: $a_k, a_{k-1}, \dots, a_1, a_0$ -t az osztási folyamat során előállt maradékok alkotják
- Pl. Alakítsuk át 7432-t 16-os számrendszerbe:

$$7432 = 464 \cdot 16 + 8$$

$$464 = 29 \cdot 16 + 0$$

$$29 = 1 \cdot 16 + 13$$

$$1 = 0 \cdot 16 + 1$$

- $7432_{10} = [1, 13, 0, 8]_{16}$

Átalakítás 10-es számrendszerből b számrendszerbe

7. feladat

Írjunk egy Python függvényt, amely átalakít egy számot 10-es számrendszerből b számrendszerbe.

A függvény kimenete egy **lista** típusú adat lesz, amely tartalmazza a b számrendszerbeli számjegyeket, ahol az előállított sorrendet *big order*-nek hívják.

```
def nrToBaseB(nr, b):  
    L = []  
    while nr >= b:  
        L = [nr % b] + L  
        nr = nr // b  
    L = [nr] + L  
    return L
```

```
>>> nrToBaseB(14, 2)  
[1, 1, 1, 0]
```

Ha az `L = [nr % b] + L` sor helyett az `L = L + [nr % b]` sort írjuk, akkor fordított sorrendet kapunk, amelyet *little order*-nek hívják.

Átalakítás b számrendszerből 10-es számrendszerbe

Példa:

- hatványösszeget számolunk a b számrendszerbeli számjegyekből
- alakítsuk át a $2D5A8_{16}$ 16-os számrendszerben megadott számot 10-es számrendszerbe:
- $2D5A8_{16} = [2, 13, 5, 10, 8]_{10}$

$$\begin{aligned}16 \cdot 0 + 2 &= 2 \\16 \cdot 2 + 13 &= 16 \cdot 2 + 13 \\16 \cdot (16 \cdot 2 + 13) + 5 &= 16^2 \cdot 2 + 16 \cdot 13 + 5 \\16 \cdot (16^2 \cdot 2 + 16 \cdot 13 + 5) + 10 &= 16^3 \cdot 2 + 16^2 \cdot 13 + 16 \cdot 5 + 10 \\16 \cdot (16^3 \cdot 2 + 16^2 \cdot 13 + 16 \cdot 5 + 10) + 8 &= 16^4 \cdot 2 + 16^3 \cdot 13 + 16^2 \cdot 5 + 16 \cdot 10 + 8\end{aligned}$$

Átalakítás b számrendszerből 10-es számrendszerbe

8. feladat

Írjunk programot, amely átalakít egy b számrendszerbeli számot 10-es számrendszerbe.

- a függvény bemenete egy lista típusú adat lesz, amely tartalmazni fogja a b számrendszerbeli számjegyeket; a függvény kimenete, pedig egy egész szám lesz
- vegyük észre, hogy az algoritmus ugyanaz mint a 4. előadáson tárgyalt *polinom helyettesítési értékét megadott értékre meghatározó horner függvény*

```
def nrFromBaseB(L, b):  
    nr = 0  
    for elem in L:  
        nr = nr * b + elem  
        #print (nr)  
    return nr
```

```
>>> nrFromBaseB([1, 1, 1, 0], 2)  
14
```

```
>>> nrFromBaseB(nrToBaseB(65, 2), 2)  
65
```

Kapcsolat a 2, 8, 16 számrendszerek között

- bináris \rightarrow oktális: **hárm**as csoportokat formálunk:

$$\text{Pl. } [1, 1, 1, 1, 0, 1, 1] \rightarrow [1, 1, 1, 1, 0, 1, 1] \rightarrow [1, 7, 3]$$

- bináris \rightarrow hexadecimális: **négy**es csoportokat formálunk

$$\text{Pl. } [1, 1, 1, 1, 0, 1, 1] \rightarrow [1, 1, 1, 1, 0, 1, 1] \rightarrow [7, B]$$

- bináris $\rightarrow 2^k$: **k-as** csoportokat formálunk

$$\text{Pl. } 2 \rightarrow 256 = 2^8, \text{ **8-as** csoportokat formálunk:}$$

$$[1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1] \rightarrow [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1] \rightarrow [57, 107]$$