Diszkrét matematika
6. előadás

MÁRTON Gyöngyvér
mgyongyi@ms.sapientia.ro

Sapientia Egyetem,
Matematika-Informatika Tanszék
Marosvásárhely, Románia

2023, őszi félév

Miről volt szó az elmúlt előadáson?

- Python: a `random`, a `numpy` és a `matplotlib` modulok, a `complex` típus,
- "híresebb" irracionális számok tízedesjegyei: π, e, ϕ,
- valós számok, komplex számok,
- algoritmusok:
  - a `sin` lánctörtképlete, radián, fok,
  - polinom helyettesítési értéke, függvények ábrázolása,
  - műveletek komplex számokkal: szorzat, hatványozás, stb
  - fraktálok: Mandelbrot, Julia,

Miről lesz szó?

- egész szám alapú számrendszerek,
- a számrendszerek közötti kapcsolat,
- vegyes alapú számrendszerek:
  - a faktoriális számrendszer,
  - a Fibonacci számrendszer,
- algoritmusok: az `n`-ik Fibonacci szám, exponenciális, lineáris, logaritmikus futásidejű algoritmusok,

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: $0, 1, 2, \ldots, 7$,
  - 10-es, decimális számrendszer, 10 szimbólum van: $0, 1, 2, \ldots, 9$,
  - 16-os hexadecimális számrendszer, 16 szimbólum van: $0, 1, 2, \ldots, 9, A, B, C, D, E, F$,
  - 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:

$$nr = a_n b^n + a_{n-1} b^{n-1} + \cdots + a_1 b^1 + a_0 b^0,$$

  ahol $n \in \mathbb{Z}^*$, $a_i \in \mathbb{Z}^*$, és $a_i < b$, minden $i \in \{0, \ldots, n\}$ értékre és $a_n \neq 0$,
- a nr szám, b számrendszerben felírt számjegyeit, a következő lista elemei alkotják: $[a_n, a_{n-1}, \ldots, a_2, a_1]$

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: $a_k, a_{k-1}, \ldots a_1, a_0$-t az osztási folyamat során előállt maradékok alkotják
- Pl. Alakítsuk át 7432-t 16-os számrendszerbe:

$$\begin{aligned} 7432 &= 464 \cdot 16 + 8 \\ 464 &= 29 \cdot 16 + 0 \\ 29 &= 1 \cdot 16 + 13 \\ 1 &= 0 \cdot 16 + 1 \end{aligned}$$

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

Átalakítás 10-es számrendszerből b számrendszerbe

1. 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 hívnak.

Átalakítás 10-es számrendszerből b számrendszerbe

Ha az eredményt `str`-ként akarom kezelni, akkor fontos, hogy a maradékok közé szóközöket, vagy más elválasztójelet tegyünk:

```
def nrToBaseBStr(nr,  b):
    L = ''
    while nr >= b:
        L = ' ' + str(nr % b) + L
        nr = nr // b
    L = str(nr) + L
    return L

>>> nrToBaseBStr(53428, 16)
    ' 13 0 11 4'

>>> nrToBaseBStr(53428, 2)
    ' 1 1 0 1 0 0 0 0 1 0 1 1 0 1 0 0'
```

Átalakítás b számrendszerből 10-es számrendszerbe

Példa:

- hatványösszeget számolunk a b számrendszerbeli számjegyekből
- alakítsuk át a $2D5A8$ 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

2. 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 $\to$ oktális: hármas csoportokat formálunk:

  Pl. $[1,1,1,1,0,1,1] \to [1,\quad 1,1,1,\quad 0,1,1] \to [1,7,3]$

- bináris $\to$ hexadecimális: négyes csoportokat formálunk

  Pl. $[1,1,1,1,0,1,1] \to [1,1,1,\quad 1,0,1,1] \to [7,B]$

- bináris $\to 2^k$: k-as csoportokat formálunk

  Pl. $2 \to 256 = 2^8$, 8-as csoportokat formálunk:
  $[1,1,1,0,0,1,0,1,1,0,1,0,1,1] \to [1,1,1,0,0,1,\quad 0,1,1,0,1,0,1,1] \to [57,107]$

Más számrendszerek

- Vegyes alapú számrendszerek
  - a számrendszer alapszáma változó
  - például, időmérés: 27. hét, 3. nap, 6 óra, 20 perc, 15 másodperc. $27_{52}3_{7}6_{24}20_{60}15_{60}$
  - az indexszám az alapszámot, a mértékegységet jelzi; a piros színű számok, a számjegyek azt jelzik, hogy az adott mértékegységből hányat használunk fel
- A faktoriális számrendszer
  - Cantor ábrázolásnak is mondják
  - alapját az képezi, hogy bármely szám egyértelműen felírható a faktoriális függvény értékeinek segítségével

$$nr = a_n \cdot n! + a_{n-1} \cdot (n-1)! + \cdots + a_2 \cdot 2! + a_1 \cdot 1!, \text{ ahol}$$
$$a_i \in [0, 1, \ldots, i], \text{ bármely } i \in [1, 2, \ldots, n]$$

  - a nr szám faktoriális számrendszerben felírt számjegyeit a következő lista elemei alkotják: $[a_n, a_{n-1}, \ldots, a_2, a_1]$

A faktoriális számrendszer

- Pl. Mennyi $(8172)_{10}$, faktoriális számrendszerbeli alakja?
  - $8172 = 1 \cdot 7! + 4 \cdot 6! + 2 \cdot 5! + 0 \cdot 4! + 2 \cdot 3! + 0 \cdot 2! + 0 \cdot 1!$
  - $8172 = 1 \cdot 5040 + 4 \cdot 720 + 2 \cdot 120 + 2 \cdot 6$
  - $8172 = [1, 4, 2, 0, 2, 0, 0]_{faktBase}$
- Pl. Mennyi $(45389)_{10}$, faktoriális számrendszerbeli alakja?
  - $45389 = 1 \cdot 8! + 1 \cdot 7! + 0 \cdot 6! + 0 \cdot 5! + 1 \cdot 4! + 0 \cdot 3! + 2 \cdot 2! + 1 \cdot 1!$
  - $45389 = 1 \cdot 40320 + 1 \cdot 5040 + 1 \cdot 24 + 2 \cdot 2 + 1 \cdot 1$
  - $45389 = [1, 1, 0, 0, 1, 0, 2, 1]_{faktBase}$
- Az algoritmus:
  - a nr szám ábrázolása esetén létezik egy n egész szám, amelyre: $n! \leq nr < (n+1)!$
  - az osztási algoritmus szerint $nr = a_n \cdot n! + r_n$, ahol $0 \leq a_n \leq n$ és $0 \leq r_n < n!$
  - hasonlóan, felírható: $r_n = a_{n-1} \cdot (n-1)! + r_{n-1}$, ahol $0 \leq a_{n-1} \leq (n-1)$ és $0 \leq r_{n-1} < (n-1)!$
  - tehát iterálva kapjuk az $a_n, a_{n-1}, \ldots a_1$ értékeket, amelyek a számrendszerbeli számjegyeket fogják jelenteni.

A faktoriális számrendszer

Hatékonyabb algoritmus a nr szám faktoriális számrendszerbeli alakjának a meghatározására:

- meghatározzuk nr 2-vel való osztási egészrészét (q_1), illetve osztási maradékát (a_1), felírható: $nr = 2 \cdot q_1 + a_1$
- meghatározzuk q_1 3-mal való osztási egészrészét, illetve osztási maradékát, felírható: $q_1 = 3 \cdot q_2 + a_2$
- folytatjuk az osztási folyamatot a 4, 5, ... értékekkel, amíg az osztási egészrész nem lesz 0
- az osztási folyamat során kiszámolt maradékok lesznek nr faktoriális számrendszerbeli számjegyei
- fennáll:
  $nr = 2 \cdot (3 \cdot (4 \cdot (5 \cdot (\ldots) + a_4) + a_3) + a_2) + a_1 =_{\ldots} \cdot 4! \cdot a_4 + 3! \cdot a_3 + 2! \cdot a_2 + a_1$
- $nr = [a_n, a_{n-1}, \ldots a_2, a_1]_{faktBase}$

A faktoriális számrendszer

Határozzuk meg az előző oldalon megadott algoritmus szerint, 8172 faktoriális számrendszerbeli számjegyeit:

nr	*q*	*n*	*a*
8172	4086	2	0
4086	1362	3	0
1362	340	4	2
340	68	5	0
68	11	6	2
11	1	7	4
1	0	8	1

```
q = nr // n
a = nr - q * n #a = nr % n
nr = q
```

$$8172 = 2 \cdot (3 \cdot (4 \cdot (5 \cdot (6 \cdot (7 \cdot (8 \cdot 0 + 1) + 4) + 2) + 0) + 2) + 0) + 0$$
$$8172 = 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 1 + 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 4 + 2 \cdot 3 \cdot 4 \cdot 5 \cdot 2 + 2 \cdot 3 \cdot 4 \cdot 0 + 2 \cdot 3 \cdot 2 + 2 \cdot 0 + 0$$
$$8172 = [1, 4, 2, 0, 2, 0, 0]_{faktBase}$$

A faktoriális számrendszer

3. feladat

Határozzuk meg egy szám faktoriális számrendszerbeli számjegyeit.

```
def digitToFaktB(nr):
    L = []
    n = 2
    while nr != 0:
        q = nr // n
        a = nr - q * n #a = nr % n
        #print (nr, q, i, a, end = "\n")
        nr = q
        L = [a] + L
        n = n + 1
    return L

>>> digitToFaktB(45389)
    [1, 1, 0, 0, 1, 0, 2, 1]
```

A Fibonacci számrendszer

- a Fibonacci számrendszert, Zeckendorf ábrázolásnak is hívjuk
- alapját az képezi, hogy bármely szám egyértelműen felírható Fibonacci számok összegeként:

$$nr = a_n \cdot F_n + a_{n-1} \cdot F_{n-1} + \cdots + a_2 \cdot F_2 + a_1 \cdot F_1, \text{ ahol}$$

$a_i \in [0, 1]$, bármely $i \in [1, 2, \ldots, n]$, és $F_n, F_{n-1}, \ldots, F_2, F_1$ a 0,1 kezdőértékeket elhagyva kapott Fibonacci számsorozat

- a nr szám Fibonacci számrendszerben felírt számjegyeit a következő lista elemei alkotják: $[a_n, a_{n-1}, \ldots, a_2, a_1]$
- legyen nr = 237, ekkor felírható:

$$237 = 233 + 3 + 1$$
$$237 = 1 \cdot 233 + 0 \cdot 144 + 0 \cdot 89 + 0 \cdot 55 + 0 \cdot 34 + 0 \cdot 21 + 0 \cdot 13 + 0 \cdot 8 + 0 \cdot 5 + 1 \cdot 3 + 0 \cdot 2 + 1 \cdot 1$$
$$237 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]_{fibBase}$$

- egy szám Fibonacci számrendszerbeli alakjában hasonlóan a kettes számrendszerhez csak két fajta szimbólum szerepel, a 0 és 1-es szimbólumok. A két alakot nem szabad összetéveszteni!
- Fibonacci számrendszerben egy szám felírásához több számjegyre (nullásra és egyesre) van szükség mint a kettes számrendszerbeli alak esetében.

A Fibonacci számrendszer

4. feladat

Határozzuk meg egy szám Fibonacci számrendszerbeli alakját. Az eredményt egy `str` *típusú változóba generáljuk ki.*

A *fibL* függvény, meghatározza, az A 1, 2-vel kezdődő Fibonacci számsorozat elemeit, amelyek nem nagyobbak mint *nr*, egy lista típusú változóba. Ezt meg kell írni!

```
def digitToFib(nr):
    fL = fibL(nr)
    L = ''
    for elem in fL[::-1]:
        if nr >= elem:
            L += '1'
            nr = nr - elem
            #print (elem, end = ' ')
        else: L += '0'
    #print('')
    return L
```

```
>>> digitToFib(100)
    1000010100

    100 = 89 + 8 + 3
```

A Fibonacci számsorozat

- A számsorozat: $0, 1, 1, 2, 3, 5, 8, 13, \ldots$,
- A rekurziós képlet: $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$,
- léteznek hatékonyabb rekurziós összefüggések,
- alkalmazásuk:
  - kombinatorika,
  - algoritmusok futási idejének elemzése,
  - minden pozitív egész szám felírható Fibonacci számok összegeként,
  - aranymetszés, zene, művészetek, természet.

A Lucas számok, ugyanaz az összefüggés a számok között, más a két kezdeti érték:

- A számsorozat: $2, 1, 3, 4, 7, 11, 18, 29, 47, \ldots$,
- Az egyik rekurziós képlet: $L_0 = 2, L_1 = 1, L_n = L_{n-1} + L_{n-2}$,

Az n-ik Fibonacci szám

5. feladat

Határozzuk meg az n-ik Fibonacci számot.

A következő algoritmus futási ideje exponenciális, az előző oldalon megadott rekurzív képletet alkalmazza:

```
def fibR1 (n):
    if n == 0: return 0
    if n == 1: return 1
    return fibR1 (n-1) + fibR1 (n-2)

>>> fibR1(10)
    55
```

Az n-ik Fibonacci szám

6. feladat

Határozzuk meg az n-ik Fibonacci számot.

A következő két algoritmus a `fibR2` és `fibR3` futási ideje lineáris:

```
def fAux(a, b, n):
    if n == 1: return a
    return fAux(b, a + b, n-1)
def fibR2(n):
    return fAux(1, 1, n)

>>> fibR2(100)
    354224848179261915075

def fibR3(n, a = 1, b = 1):
   if n == 1: return a
   return fibR3(n-1, b, a + b)

>>> fibR3(200)
    280571172992510140037611932413038677189525
```

Az n-ik Fibonacci szám

7. feladat

Határozzuk meg az n-ik Fibonacci számot:

A következő képletet alkalmazó algoritmus futási ideje logaritmikus:

$$F_n = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1}, \qquad F_5 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{4} = \begin{pmatrix} 5 & 3 \\ 3 & 2 \end{pmatrix}$$

ahol meg kell tehát határozni két 2×2 mátrix szorzatát:

$$\begin{pmatrix} a & b \\ b & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ f & g \end{pmatrix} = \begin{pmatrix} a \cdot e + b \cdot f & a \cdot f + b \cdot g \\ a \cdot f + b \cdot g & b \cdot f + d \cdot g \end{pmatrix}$$

- fennáll: $a \cdot f + b \cdot g == b \cdot e + d \cdot f$
- a hatékonyságból, a mátrixot egy számhármassal lehet ábrázolni, mert a mellékátlón az elemek megegyeznek

Az n-ik Fibonacci szám

```
def szor(t1, t2):
    a, b, d = t1
    e, f, g = t2
    return (a * e + b * f, a * f + b * g, b * f + d * g)

def fibR(n, x = (1,1,0), res = (1,0,1)):
    if n == 0: return res[1]
    if n % 2 == 1:
        res = szor(res, x)
        x = szor(x, x)
        return fibR(n // 2, x, res)
    x = szor(x, x)
    return fibR(n // 2, x, res)

>>> fibR(100)
    354224848179261915075
```

Az n-ik Fibonacci szám

A `fibI`, az előző algoritmus iteratív változata, ahol a `szor` függvény is az előző oldalon van megadva:

```
def fibI(n):
    res = (1, 0, 1)
    x = (1, 1, 0)
    n -= 1
    while n != 0:
        if n % 2 == 1: res = szor(res, x)
        n = n // 2
        x = szor(x, x)
    return res[0]
```

```
>>> fibI(100)
    354224848179261915075
```

Az n-ik Fibonacci szám

$(1,1,0)^{28}$, azaz a 29-ik Fibonacci szám meghatározása:

```
def fibI(n):
    res = (1, 0, 1)
    x = (1, 1, 0)
    n -= 1
    while n != 0:
        if n % 2 == 1: res = szor(res, x)
        n = n // 2
        x = szor(x, x)
    return res[0]
```

x	n	res
		$(1,0,1$
$(1,1,0)$	28	$(1,1,0)$
$(1,1,0)^2 = (2,1,1)$	14	$(1,1,0)$
$(1,1,0)^4 = (5,3,2)$	7	$(5,3,2)$
$(1,1,0)^8 = (34,21,13)$	3	$(233,144,89)$
$(1,1,0)^{16} = (1597,987,610)$	1	$(514229,317811,196418)$

Az eredmény:

$$F_{29} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{28} = \begin{pmatrix} 514229 & 317811 \\ 317811 & 196418 \end{pmatrix}$$