

Komputeralgebra rendszerek

XVI. Összetett adatszerkezetek a Sage-ben

Czirbusz Sándor
czirbusz@gmail.com

Komputeralgebra Tanszék
ELTE Informatika Kar

2010-2011 őszi

Index I

1 Tuplek, listák, sorozatok

- Tuplek
- Sorozatok
- Listák
 - A lista definiálása
 - Egyszerű listakezelés
 - Listainformációk
- Listaértelmezés
- Sorkezelés

2 Halmazok

- A Pythonos halmazok
- A Sage halmazok

3 Közös tulajdonságok

4 Szótárak

Index II

- A szótár fogalma
- A szótár használata

5 Iterátorok

6 Tömbök, vektorok

- Vektorok
- Mátrixok
 - Mátrix-konstrukciók
 - Mátrixműveletek
 - Lineáris algebra

Tuplek

Tuple

A **tuple** egy nem módosítható rendezett sorozat, az elemek különféle típusúak is lehetnek.

- Létrehozása : $v = \text{ert1}, \text{ert2}, \dots$ (tuple packaging).
Zárójelezve jelenik meg
- Hivatkozás elemre: $v[i]$
- $\text{type}(v) : \langle \text{type 'tuple'} \rangle$
- Alkalmazások : koordináták, rekordok
- Tuple szétszedés : $x, y, z = v$

Sorozatok

Sorozatok

A **sorozatok** azonos típusú elemek listája , mely nem módosítható is lehet.

- Létrehozása : **Sequence([lista])** vagy **seq([lista])**, **opt**
- Az opciók :
 - megadhatunk egy típus–kényszerítést (coercing)
 - nem módosítható : `immutable = true`
- Elemek típusa : **.universe()**
- Módosítható–e : **.is_immutable()**
- Példák létrehozás : `Sequence([1,2,3], immutable=True)`

Listák

Listák

- A lista alapvető adatszerkezet a Sage-ban, elemei bármik lehetnek
- Létrehozása : szögletes zárójelekkel, vagy a **list** –kulcsszóval
- Üres lista : []
- Az indexelés 0-tól indul, ellentétben a Maple–vel.
- Indexelés : $L[m:n]$ alakban (list slicing) , a negatív index visszafelé működik

Listaműveletek

A műveletek helyben történnek

Listaműveletek

- Hozzáadás : **.append(value)**
- Érték első előfordulásának törlése : **.remove(value)**
- Beszúrás adott index elé : **.insert(index, value)**
- Megfordítás : **.reverse()**
- Rendezés : **.sort()**
- Bővítés : **.extend(iterable)**
- Elem törlése és visszaadása : **.pop([index])**

Listainformációk, veremkezelés

Listainformációk

- A lista hossza : **len(lista)** nem alkalmazható objektum–tulajdonságként
- Egy elem előfordulásainak száma : **.count(value)**

A **.pop()** és az **.append()** segítségével megvalósítható a stack–szerkezet

Listaértelmezés

...avagy list comprehension.

Listakezelés felsőfokon

- Szintaxis : `[kif(x) for x in iteralhato [if felt(x)]]`
- Példa :

`vec1 = [2, 4, 6]`

`vec2 = [4, 3, -9]`

`[x + y for x in vec1 for y in vec2]`

Kimenete: `[6, 5, -7, 8, 7, -5, 10, 9, -3]`

Sorkezelés

Sorkezelés

A **from collections import deque** hatására egy sorkezelő áll rendelkezésünkre. A listaműveletek mellet a lista „baloldalát” is könnyű lesz kezelnünk.

- Sor létrehozása : **q = deque([lista])**
- Nyilvánvaló függvények : **.popleft()**, **.appendleft()**, **.extendleft()**
- Az összes elem törlése : **.clear()**

A Python halmazai

A halmaz módosítható, rendezetlen kollekció, elemei különfélék lehetnek.

A Python halmazai

- Létrehozása : **set(tuple vagy lista)**
- Üres halmaz : **set()**
- A redundáns elemek automatikusan törlődnek.
- A sorrend „borul”
- Halmazműveletek : **.union(), .intersection(), .difference(), .symmetric_difference(), .difference()**
- Információk : **.isdisjoint(), .issubset(), issuperset()**
- A műveleteknek van **_update** toldalékú párja, ami az obejktumot átírja

A Sage hamazai

A Sage hamazai

- Létrehozása : **Set(tuple vagy lista)**
- Üres halmaz : üres tuple-ra vagy üres listára alkalmazzuk a függvényt
- Az összes halmazelméleti művelet neve egyezik az előzővel, de mindig az első operandust módosítják
- Alkalmazása : **EZ A** matematika halmaz jobb megközelítése; struktúrákat is tekinthetünk halmaznak
- Végtelen halmazokat is létre tudunk hozni: **Set(Primes())**.
„Set of all prime numbers: 2, 3, 5, 7, ...”

Közös tulajdonságok

A halmazok, listák, tuplék rendelkeznek azonos objektum–tulajdonságokkal

Halmazok, tuplék és listák

- Elemszám : **len(L)**
- Bennfoglaltsás : **$x \in L$**

Listák és tuplék

- Egy elem előfordulásai : **L.count(obj)**
- Egy elem első előfordulása : **L.index(obj)**
- Egyesítés : **$L_1 + L_2$**

A szótár struktúra

A Python-os terminológia : dictionary; más nyelvekben asszociatív tömb, vagy hash

dictionary

- Kulcs–érték párok rendezetlen kolleckciója
- Konstrukció : $d = \{k1:e1, k2:e2, \dots\}$
- Konverálás : **dict([párok listája])**
- Egy elem lekérdezése : $d[kn]$
- Egyszerű infók : **.keys()**, **.values()**

A szótár használata

A dictionary tulajdonságai

- Az összes elem törlése : **.clear()**
- Másolat készítése **.copy()**
- Elem lekérdezése másképp : **.get(k[, default])**
- Iterátorok : **.iteritems()**, **.iterkeys()**, **.itervalues()** Példa :
[a*b for a, b in d.iteritems()] összeszorozza a kulcs–érték párokat
- Érték vagy teljes elem törlése : **.pop(k)**, **.popitem(k)**
- Frissítés : **.update(d2)** (Az azonos kulcsút felülírja, az újat beszúrja)

Iterátorok

Az iterátor memóriatakarékosság miatt menet közben állítja elő az adatot (generátor)

Az iterátorok

- Konstrukció : `w = (kif(n) for n in iterálható if felt(n))`
- Konverzió : `iter(list|tuple[, strázsa])`
- A Sage megkülönbözteti az iterátorokat az alaptípus szerint
- Amit minden iterátor tud `.next()`
- Példa :

```
w = (4 * p + 1 for p in Primes() if is_prime(4 * p + 1))
w.next()
```

- A Sage objektumok jelentős rész rendelkezik saját iterátorral.

Konstrukció és műveletek

vector

- Konstruktor : **vector([R,] lista|szótár)**
- Az első paraméter opcionális, az alapgyűrű. Default : \mathbb{Z}
- Ha szótárral adjuk meg, akkor a kulcs–érték párokkal a vektor adott pozícióját lévő értéket adjuk meg. (A többi 0)
- Műveletek : lineáris kombináció, **.dot_product(v)** (avagy **.inner_product()**), **.cross_product(v)**
- Norma : **.norm([p])**; ha nem adjuk meg, akkor $p=2$, az Euklideszi norma. Lehet végtelen is.

Mátrix–konstrukciók

Az első paraméter itt is az alapgyűrű, most nem tüntetjük föl (Default: \mathbb{Z})

Konstrukció

- Listák listája : **matrix([sor1,sor2,...])** ; a sorok azonos hosszúságú listák
- Megadva a sorok számát : **matrix(r,lista)**; „r” sorra bontja a listát. Hiba, ha nincs elég elem
- Mindkét koordinátát megadva : **matrix(r,c,[par1,par2,...])**; pontosan $r * c$ elem kell felsorolni
- Zérus–mátrix : **matrix(r,c,0)**
- Egységmátrix : **identity_matrix(c)**
- A pontos méret feltüntetése estén az elemek megadhatók szótárral is.

Mátrixműveletek I

Alapműveletek

- Összeadás , szorzás, hatványozás : mint linalgban, nincs speciális műveleti jel
- Ha „ v ” vektor, „ A ” mátrix, összeszorozhatók, akkor a **`B.iterate(v)`** olyan mátrixot generál, mely sorai : $v * B^k$
- Transzponálás : **`.transpose()`**
- Exponenciális függvény : **`.exp()`**
- Mátrixpolinom : **`.act_on_polynomial(p)`**, ha „ p ” ugyanazon gyűrű fölött értelmezett.

Mátrixműveletek II

Manipulációk

- Sor/oszlop szorzása skalárral : **`.rescale_row|col(i, s[, start=0])`**
- Egyik sor/oszlop többszörösének hozzáadása a másikhoz: **`.add_multiple_of_row|column(i, j, s[, start=0])`**
- Sor/oszlop csere : **`.swap_rows|columns(r1, r2)`**
- Háromszögalakra hozás : **`.echelon_form()`**, **`.echelonize()`**
- Részmátrix : lehetőség van sorok, oszlopok, blokkok külön kezelésére (megannyi függvény)
- Mátrixok összerakás: **`.augment()`** és **`.stack()`**

Lineáris algebra I

Elemi linalg

- Trace, determináns, norma : szokásos kulcsszavak
- Karakterisztikus polinom :
.characteristic_polynomial(var). vagy röviden
.charpoly(var)
- Sajátértékek : **.eigenvalues()** (Lehetőség van a bal é jobboldali kezelésére)
- Jordan forma : **.jordan_form()**
- LU-dekompozíció : **.LU()**

Lineáris algebra II

Vektor- és mátrixterek

A Sage-ben létrehozhatunk adott gyűrű, test fölött vektorteret **VectorSpace**, illetve **MatrixSpace** kulcsszavakkal és minden gyűrűvel kapcsolatos tevékenységet el tudunk végezni

Példák

Lásd a munkalapokon