

Feladat

Valósítsa meg az egész számokat tartalmazó felsőháromszög mátrixtípust (a mátrixok a főátlójuk alatt csak nullát tartalmaznak)! Ilyenkor elegendő csak a főátló és afeletti elemeket reprezentálni egy sorozatban, amelyet egy dinamikus helyfoglalású tömbben helyezünk el. Implementálja önálló metódusként a mátrix i -edik sorának j -edik elemét visszaadó műveletet, valamint hatékony összeadás és szorzás műveleteket, továbbá a mátrix (négyzetes alakú) kiírását, és végül a másoló konstruktort és az értékadás operátort!

Felsőháromszög mátrix osztály

A feladat lényege egy felsőháromszög mátrix típusnak a megvalósítása.

Típusérték-halmaz

Olyan számokat tartalmazó $n \times n$ ($n \in \mathbb{N}$) -es négyzetes mátrixokkal akarunk dolgozni, amelyek csak a felső háromszögben tartalmazzak nullától különböző elemeket. Az $n \in \mathbb{N}$ ennek a típusnak egy paramétere, amely a típusérték-halmaz mátrixának méretét határozza meg.

Típus-műveletek

Lekérdezés

A mátrix j -edik sorának i -edik pozícióján ($i, j \in [1..n]$) álló érték kiolvasása $e := a[i, j]$

Megjegyezzük, hogy ez a művelet csak $i \geq j$ esetén igényel tényleges tevékenységet, hiszen egyébként a visszaadott elem nulla.

Felülírás

A mátrix i -edik sorának j -edik pozíciójára ($i, j \in [1..n]$) új érték beírása: $a[i, j] := e$. A felső háromszögön kívüli elemet nem szabad felülírni, azaz $i \geq j$.

Megjegyezzük, hogy ez a művelet csak $i \geq j$ esetén igényel tényleges tevékenységet; $i < j$ esetén hibás, amennyiben egy nemnulla értéket akarunk a mátrixba tenni.

Összeadás

Két mátrix összeadása: $c := a + b$. Az összeadásban szereplő mátrixos azonos méretűek.

$$\forall i, j \in [1..n]: c[i, j] := a[i, j] + b[i, j]$$

Szorzás mátrixal

Két mátrix összeszorozása: $c := a * b$. A szorzásban szereplő mátrixok azonos méretűek.

$$\forall i, j \in [1..n]: c[i, j] := \sum_{k=1}^n a[i, k] * b[k, j]$$

De mivel felső háromszög mátrixról van szó, így a szorzások száma csökkenthető azzal, hogy kihagyjuk a főátló alatti területeket:

$$\forall i, j \in [1..n]: c[i, j] := \sum_{k=M_1}^{M_2} a[i, k] * b[k, j], M_1 = \min\{i, j\}, M_2 = \max\{i, j\}$$

DOMJ1R

skorka.bence@gmail.com

Szorzás számmal

Egy mátrix szorzása egy számmal: $c := a * \lambda$ ($\lambda \in \mathbb{N}$).

$$\forall i, j \in [1..n]: c[i, j] := a[i, j] * \lambda$$

Egy $n \times n$ -es felsőháromszög mátrixban csak a felső háromszöget kell ábrázlni:

$$\begin{array}{ccc} a_{1,1} & a_{2,1} & a_{3,1} \dots a_{n,1} \\ 0 & a_{2,2} & a_{3,2} \dots a_{n,2} \\ 0 & 0 & a_{3,3} \dots a_{n,3} \end{array}$$



Ehhez egy 0-tól $\frac{n(n+1)}{2}$ -ig indexelt egydimenziós tömbre (v) van szükségünk. Ennek segítségével a felsőháromszög mátrix bármelyik elemét meghatározhatjuk az alábbi függvény alapján: (az *index* függvény az implementációban)

$$a[i,j] = \begin{cases} v[index(i,j)] & i \geq j \\ 0 & i < j \end{cases}$$

Implementáció

Segédfüggvények

$$\begin{array}{l} \text{VektorMeret}(n) \\ \text{Return} \left(\frac{n(n+1)}{2} \right) \end{array}$$

$$\begin{array}{l} \text{index}(i,j) \\ \text{Return} \left(i - j + \frac{(2n - j + 1) \times j}{2} \right) \end{array}$$

Lekérdezés

<i>Lekerdezes(i,j)</i>	
$i \geq j$	
<i>Return</i> ($v[index(i,j)]$)	<i>Return</i> (0)

Felülírás

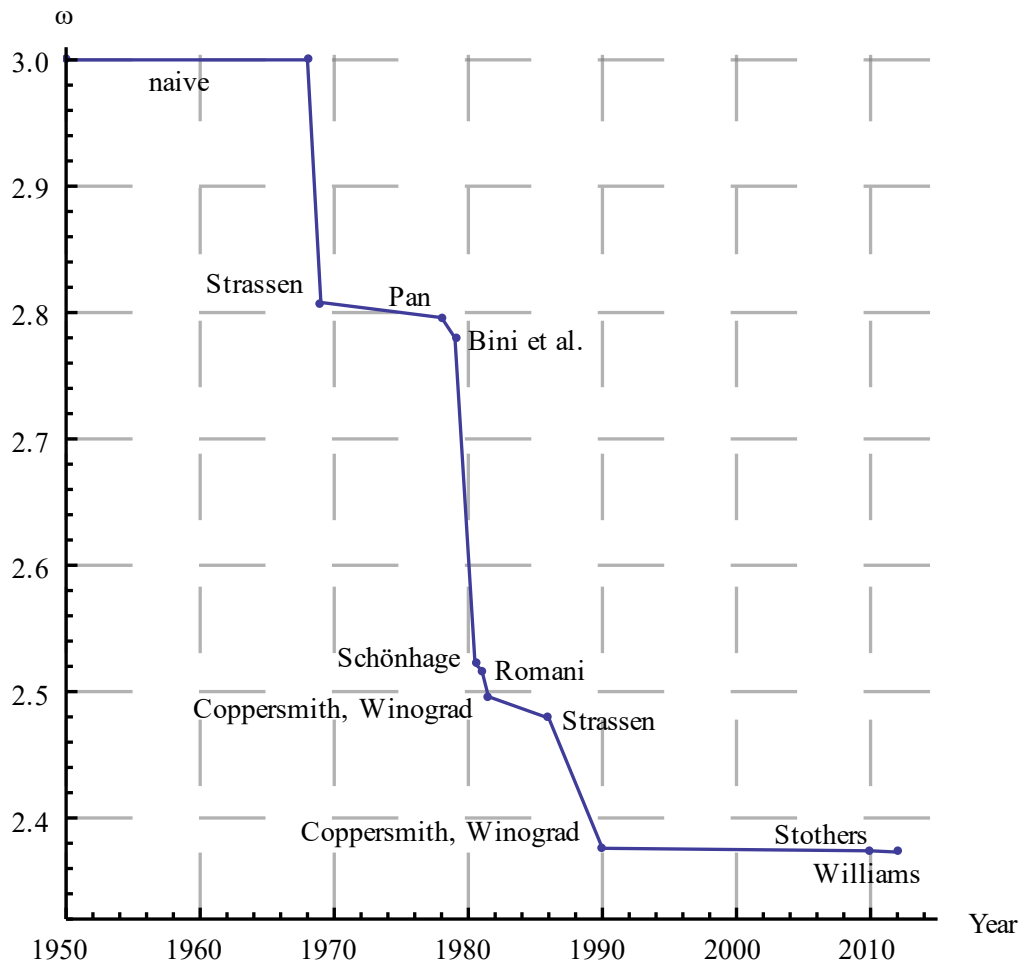
<i>Módosítás(i,j,ertek)</i>	
$i \geq j$	
$v[index(i,j)] := \text{ertek}$	<i>SKIP</i>

Összeg(m_1, m_2)	
$m_1.meret() = m_2.meret()$	
$o := \text{new Matrix}(m_1.meret())$	SKIP
$i := 1..VektorMeret(o.meret())$	
$o.v[i] := m_1.v[i] + m_2.v[i]$	
Return(o)	

Szorzás

Szorzat(m_1, m_2)	
$m_1.meret() = m_2.meret()$	
$o := \text{new Matrix}(m_1.meret())$	SKIP
$x := 1..o.meret()$	
$y := 1..x$	
$tmp := 0$	
$i := \text{Min}\{x, y\}.. \text{Max}\{x, y\}$	
$tmp := tmp + m_1.v[\text{index}(x, i)] * m_2.v[\text{index}(i, y)]$	
$o.v[\text{index}(x, y)] := tmp$	
Return(o)	

Igen, ez egy n^3 -ös algoritmus, de egyszerű implementációban nincs is sokkal jobb sajnos:



Osztály

A felsőháromszög matrixok típusát osztály segítségével úgy valósítjuk meg, hogy a matrix méretét a konstruktorban paraméterként át kell adni. Ennek hátránya, hogy minden művelet előtt ellenőrizni kell, hogy a műveletet végző mátrixok azonos méretűek e. Előnye, hogy a matrix mérete később dinamikusan változhat másolás, vagy fileből betöltés esetén.

<i>UpperTriangularMatrix</i>
<i>–size: int</i>
<i>–data: int[0..k], ahol k értékét a calculateVectorLength függvény számolja.</i>
<i>–calculateVectorLength(): int</i> <i>–calculateVectorPosition(int, int): int</i> <i>–copyFrom(UpperTriangularMatrix)</i> <i>+UpperTriangularMatrix(int)</i> <i>+UpperTriangularMatrix(UpperTriangularMatrix)</i> <i>+operator()(int, int): int</i> <i>+operator + (UpperTriangularMatrix, UpperTriangularMatrix): UpperTriangularMatrix</i> <i>+operator * (UpperTriangularMatrix, int): UpperTriangularMatrix</i> <i>+operator * (UpperTriangularMatrix, UpperTriangularMatrix): UpperTriangularMatrix</i> <i>+writeData(ostream)</i> <i>+readData(istream)</i> <i>+operator << (ostream), nem ugyanaz, mint a writeData!!</i> <i>+getSize(): int</i>

Az adatokat tároló tomb egy saját vektortípus, ami kezeli a dinamikus helyfoglalást. Így a hely lefoglalásával nem a felső háromszög matrix osztály törődik, mert nem neki a feladata.

Tesztelési terv

Vektorosztály

A vektorosztályból csak a mátrixot érintő funkciók tesztelése érdemes, így a *push_back*, *pop_back*, egyéb függvények nem.

1. Vektor méretezhetősége
 - a. Létrehozható e vektor adott mérettel
 - b. Átméretezhető e a vektor
2. Vektorelemek elérhetősége
 - a. Egy n méretű vektor $0..n - 1$ elemei közül mindegyik elérhető?
 - b. ~ módosítható?
 - c. ~ megtartja az értékét?

Mátrixosztály

1. Új n méretű üres
 - a. A mátrix létrejön?
 - b. Minden eleme 0?
2. Helyes pozíciók elérése
 - a. Elérhető?
 - b. Alapértelmezetten 0?
3. Helytelen pozíciók elérése
 - a. Hibát dob?
4. Szorzás számmal
5. Összeadás mátrixal
6. Szorzás mátrixal