X-type matrix

Ren Koike, ren.ko1139@gmail.com
March 28, 2023

1 Description

The X matrix type which contains integers is a square matrix that can contain nonzero entries only in their two diagonals. It doesn't store the zero entries but the entries that can be nonzero in a sequence. Implement as methods: getting and setting the entry located at index (i, j), setting a new matrix, adding and multiplying two matrices, and printing the matrix (in a square shape). The example of X matrix (3x3):

$$X = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 3 & 0 \\ 4 & 0 & 5 \end{bmatrix} \tag{1.1}$$

2 X type matrix

2.1 Set of values

$$X(n) = \{a \in \mathbb{R}^{n \times n} | \forall i, j \in [1..n], 2 \nmid n \wedge n \geq 3 : (i \neq j \wedge i + j \neq n + 1) \Rightarrow a[i, j] = 0\} \tag{2.1}$$

2.2 Operations

1. Getting an entry
   Getting the entry of the ith column and jth row $(i, j \in [1..n]) : e := a[i, j]$.

2. Sum
   Sum of two matrices: $c := a + b$. The matrices have the same size.

3. Multiplication
   Multiplication of two matrices: $c := a * b$. The matrices have the same size.

2.3 Representation

Only the diagonals of the $n \times n$ matrix has to be stored. For example, if n = 3,

$$a = \begin{bmatrix} a_{11} & 0 & a_{13} \\ 0 & a_{22} & 0 \\ a_{31} & 0 & a_{33} \end{bmatrix} \Leftrightarrow v = < a_{11}, a_{13}, a_{22}, a_{31}, a_{33} >$$

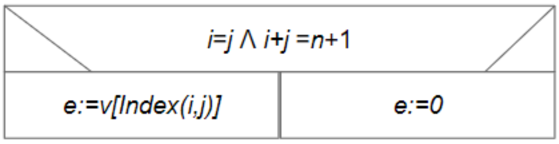
Only a one-dimension array (v) is needed, with the help of which any entry of the diagonal matrix can be get:

$$a[i,j] = \begin{cases} v[Index(i,j)] & \text{if } i = j \text{ or } i + j = n + 1 \\ 0 & \text{otherwise.} \end{cases}$$

*Index function helps to calculate the corresponding index of the vector from i, j.

2.4 Implementation

1. Getting an entry
   Getting the entry of the ith column and jth row $(i, j \in [1..n])$ $e := a[i, j]$ where the matrix is represented by $v, 1 \in n$, and n stands for the size of the matrix can be implemented as



2. Sum
   The sum of matrices a and b (represented by arrays t and u) goes to matrix c (represented by array u), where all of the arrays have to have the same size.
   $\forall i \in [0..n-1] : u[i] := v[i] + t[i]$

3. Multiplication
   The product of matrices a and b (represented by arrays t and u) goes to matrix c (represented by array u), where all of the arrays have to have the same size.
   $\forall i, j \in [0..n-1] : u[Index(i,j)] := \sum_{k=0}^{n-1} v[Index(i,k)] * t[Index(k,j)]$

3 Testing

3.1 Testing the operations(black-box)

- Creating, reading, and writing matrices of different size.
  - 0, 1, 3, 4, 5 matrix
- Getting and setting an entry

    - Getting and setting an entry in the diagonal
    - Getting and setting an entry outside the diagonal
    - Illegal index, indexing a 0-size matrix

- Copy constructor
    - Creating matrix b based on matrix a, comparing the entries of the two matrices. Then, changing one of the matrices and comparing the entries of the two matrices.

- Sum of two matrices, command $c := a + b$.
    - With matrices of different size (size of a and b differs, size of c and a differs)
    - Checking the commutativity (a + b == b + a)
    - Checking the associativity (a + b + c == (a + b) + c == a + (b + c))
    - Checking the neutral element (a + 0 == a, where 0 is the null matrix)

- Multiplication of two matrices, command $c := a * b$.
    - With matrices of different size (size of a and b differs, size of c and a differs)
    - Checking the commutativity (a * b == b * a)
    - Checking the associativity (a * b * c == (a * b) * c == a * (b * c))
    - Checking the neutral element (a * 0 == 0, where 0 is the null matrix)
    - Checking the identity element (a * 1 == a, where 1 is the identity matrix)

3.2 Testing based on the code (white-box)

1. Creating an extreme-size matrix (-1, 0, 999).

2. Generating and catching exceptions.