|  |  |  |
| --- | --- | --- |
| Amina Arooj  Uxm59h  uxm59h@inf.elte.hu Group 3 | **1. assignment/2. task** | 22 March 2023 |

# Task

*Implement the diagonal matrix type which contains integers. These are square matrices that can contain nonzero entries only in their two diagonals. Don't store the zero entries. Store only the entries that can be nonzero in a sequence. Implement as methods: getting and setting the entry located at index (i, j), adding and multiplying two matrices, reading and printing the matrix (in a square shape).*

# Diagonal matrix type

***Set of values***

*Diag*(*n*) = { *a*∈ℤn×n ⎪∀*i,j*∈[*1*..*n*]: *i*≠*j* ∧ *(i+j*≠*n+1)* → *a*[*i,j*]=*0* }

## Operations

1. *Getting an entry*

Getting the entry of the *i*th column and *j*th row (*i,j*∈[*1*..*n*]): *e*:=*a*[*i,j*] if( i = j) , f:=*a*[*i,j*]if ( i +j=n+1)

Formally: *A* : *Diag*(*n*) ℤ × ℤ × ℤ × ℤ

*i j e f*

*Pre* = ( *a*=*a’* ∧ *i*=*i’* ∧ *j*=*j’*∧  *i,j*∈[*1*..*n*] )

*Post =* ( *Pre* ∧ *e* ∧ *f=a*[*i,j*] )

This operation needs any action only if *i*=*j* or *(i + j = n + 1)* otherwise the output is zero.

1. *Setting an entry*

Setting the entry of the *i*th column and *jth* row (*i,j*∈[*1*..*n*]): *a*[*i,j*]:=*e if (i=j) and a*[*i,j*]:=*f if (i+j=n+1)*.. Entries outside the diagonals cannot be modified (*i*=*j*) ∨ *(i+j=n)* .

Formally: *A* = *Diag*(*n*) ℤ × ℤ × ℤ× ℤ

* 1. *i j e f*

*Pre* = ( *e*=*e’* ∧ *f*=*f’* ∧ *a*=*a’* ∧ *i*=*i’* ∧ *j*=*j’* ∧ *i,j*∈[*1*..*n*] ∧ *i*=*j* ∨ *(i+j=n+1* )

*Post* = (*e*=*e’*∧ *f*=*f’* ∧ *i*=*i’* ∧ *j*=*j’* ∧ *a*[*i,j*]=*e* ∧ *a*[*i,j*]=*f* ∀*k,l*∈[*1*..*n*]: (*k*≠*i* ∧ *l*≠*j*)→ *a*[*k,l*]=*a’*[*k,l*] )

This operation needs any action only if *i*=*j or (i+j=n+1*)*,* otherwise it gives an error if we want to modify a zero entry.

1. *Sum*

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

Formally: *A* = *Diag*(*n*) × *Diag*(*n*) × *Diag*(*n*)

* 1. *b c*

*Pre* = ( *a*=*a’* ∧ *b*=*b’*)

*Post* = ( *Pre* ∧ ∀*i,j*∈[*1*..*n*]: *c*[*i*,*j*]= *a*[*i*,*j*] + *b*[*i*,*j*] )

1. *Multiplication*

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

Formally: *A* = *Diag*(*n*) × *Diag*(*n*) × *Diag*(*n*)

* 1. *b c*

*Pre =* ( *a*=*a’* ∧ *b*=*b’*)

*Post* = ( *Pre* ∧ ∀*i,j*∈[*1*..*n*]: *c*[*i*,*j*]=∑*k=1..n**a*[*i*,*k*] \* *b*[*k*,*j*])

## Representation

Only the diagonal of the *n×n* matrix has to be strored.

*a11 0 0 … c11*

*0 a22  c22 … 0*

*a = 0 c33 a33  … 0*  ↔

*cnn 0 0 … ann*

*v1* = < *a11 a22  a33 ann >*

*v2* = < *c11 c22  c33 cnn >*

Two one-dimension arrays (*v1 and v2*) are needed, with the help of which any entry of the matrix can be get:

𝑎[𝑖, 𝑗] =

## Implementation

1. *Getting an entry*

Getting the entry of the *i*th column and *jth* row (*i,j*∈[*1*..*n*]) *e*:=*a*[*i,j*] where the matrix is represented by 1≤*i*≤*n*, and *n* stands for the size of the matrix can be implemented as

A diagram of a mathematical equation

Description automatically generated

1. *Setting an entry*

Setting the entry of the *i*th column and *jth* row (*i,j*[*1*..*n*]) *a*[*i*,*j*]:=*e* where the matrix is represented by ,1≤*i*≤*n*, and *n* stands for the size of the matrix can be implemented as

A diagram of a mathematical equation

Description automatically generated

1. *Sum*

The sum of matrices *s* and *t* (type diag) goes to matrix *u* (diag *type*), where all of the diag types have to have the same size.

∀*i*∈[*0*..*n-1*]: *u.v1*[*i*]:= *s.v1*[*i*] + *t.v1*[*i*]

*u.v2*[*i*]:= *s.v2*[*i*] +*t.v2*[*i*]

sum will be outputted as u ( matrix of type diag with non zero main and secondary diagonals).

1. *Multiplication*

The product of matrices *s* and *t* (type diag) goes to matrix *u* (diag *type*), where all of the diag types have to have the same size.

A diagram of a mathematical equation

Description automatically generated with medium confidence

# Testing

Testing the operations (black box testing)

1. Creating, reading, and writing matrices of different size.
   1. 0, 1, 2, 5-size matrix
2. Getting and setting an entry
   1. Getting and setting an entry in the diagonal
   2. Getting and setting an entry outside the diagonal
   3. Illegal index, indexing a 0-size matrix
3. Copy constructor
   1. 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.
4. Assignment operator
   1. Executing command *b=a* for matrices *a* and *b* (with and without same size), comparing the entries of the two matrices. Then, changing one of the matrices and comparing the entries of the two matrices.
   2. Executing command *c=b=a* for matrices *a, b,* and *c* (with and without same size), comparing the entries of the three matrices. Then, changing one of the matrices and comparing the entries of the three matrices.
   3. Executing command *a=a* for matrix *a*.
5. Sum of two matrices, command *c*:=*a*+*b*.
   1. With matrices of different size (size of *a* and *b* differs, size of *c* and *a* differs)
   2. Checking the commutativity (a + b == b + a)
   3. Checking the associativity (a + b + c == (a + b) + c == a + (b + c))
   4. Checking the neutral element (a + 0 == a, where 0 is the null matrix) 6) Multiplication of two matrices, command *c*:=*a*\**b*.
   5. With matrices of different size (size of *a* and *b* differs, size of *c* and *a* differs)
   6. Checking the commutativity (a \* b b \* a)
   7. Checking the associativity (a \* b \* c == (a \* b) \* c == a \* (b \* c))
   8. Checking the neutral element (a \* 0 == 0, where 0 is the null matrix)
   9. Checking the identity element (a \* 1 == a, where 1 is the identity matrix)

Testing based on the code (white box testing) 1. Creating an extreme-size matrix (-1, 0, 1, 1000).

2. Generating and catching exceptions.