Jamala Khaligova 1.assignment/ 3rd task (N matrix) 15th March 2020

A0JCKR

cemale_2000@mail.ru

Teréz Anna Várkonyi's group.

Task

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

N matrix type

Set of values Mtx(n) = { a∈ $\mathbb{Z}^{n\times n}$ | ∀i,j∈[0..n-1]: j≠ 0,j≠n-1,i≠j → a[i,j]=0 }

Operations .

Getting an entry

Getting the entry of the ith column and jth row (i,j∈[0..n-1]): e:=a[i,j].

Formally:

A : Matrix(n) × ℤ × ℤ × ℤ

a *i* *j* *e*

Pre = (a=a' ∧ i=i' ∧ j=j' ∧ i,j∈[0..n-1])

Post = (Pre ∧ e=a[i,j])

This operation needs any action if *j=0,j=n-1,i=j*, otherwise the output is zero.

2. Setting an entry

Setting the entry of the ith column and jth row (i,j∈[0..n-1]): a[i,j]:=e.

Entries outside the N cannot be modified (j=0,j=n-1,i=j). Formally:

A = Matrix(n) × ℤ × ℤ × ℤ

a *i* *j* *e*

Pre = (e=e' ∧ a=a' ∧ i=i' ∧ j=j' ∧ i,j∈[0..n-1] ∧ j=0 ∧j=n-1 ∧ i=j)

Post = (e=e' ∧ i=i' ∧ j=j' ∧ a[i,j]=e ∧ ∀k,l∈[0..n-1]: (k≠i ∨ l≠j)→ a[k,l]=a'[k,l])

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

3. Sum

Sum of two matrices: c:=a+b.

The matrices have the same size.

Formally: $A = Matrix(n) \times Matrix(n) \times Matrix(n)$

a b c

Pre = ($a=a' \wedge b=b'$)

Post = (Pre $\wedge \forall i,j \in [0..n-1]: c[i,j]= a[i,j] + b[i,j]$)

In case of N matrices there is an easier version: $\forall i,j \in [0..n-1]$,: $c[i,j]= a[i,j] + b[i,j]$ and $\forall i,j \in [0..n-1]$: $j \neq 0, j \neq n-1, i \neq j \rightarrow c[i,j]=0$.

4. Multiplication Multiplication of two matrices:

c:=a*b. The matrices have the same size.

Formally:

$A = Matrix(n) \times Matrix(n) \times Matrix(n)$

a b c

Pre = ($a=a' \wedge b=b'$)

Post = (Pre $\wedge \forall i,j \in [0..n-1]: c[i,j]= \Sigma k=0..n-1\ a[i,k] * b[k,j]$)

In case of N matrices there is an easier version: $\forall i,j \in [0..n-1]: c[i,j]= \Sigma k=0..n-1\ a[i,k] * b[k,j]$ and $\forall i,j \in [0..n-1]: j \neq 0, j \neq n-1, i \neq j \rightarrow c[i,j]=0$.

Representation

Only the first,last columns and diagonal of the n×n matrix has to be stored.

$$a = \begin{array}{llllll} a_{00} & 0 & 0 & \dots & 0 & a_{0(n-1)} \\ a_{10} & a_{11} & 0 & \dots & 0 & a_{1(n-1)} \\ a_{20} & 0 & a_{22} & & & \\ a_{(n-1)0} & 0 & 0 & \dots & 0 & a_{(n-1)(n-1)} \end{array}$$

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

$$a[i,j] = \begin{cases} v[i] \ if\ j=0 \\ v[n+i] \ if\ j=n-1 \\ v[n*2+(i-1)] \ if\ i=j \wedge j \neq 0 \wedge j \neq n-1 \\ 0 \ if\ i \neq j \wedge j \neq 0 \wedge j \neq n-1 \end{cases}$$

Implementation

1. Getting an entry
   Getting the entry of the ith column and jth row ($i,j \in [0..n-1]$) $e:=a[i,j]$ where the matrix is represented by $v, 0 \le i \le n-1$, and n stands for the size of the matrix can be implemented as

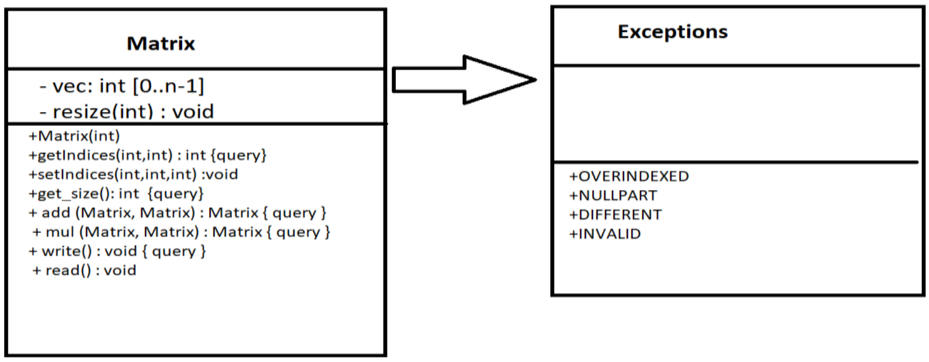
2. Setting an entry Setting the entry of the ith column and jth row ($i,j \in [0..n-1]$) $a[i,j]:=e$ where the matrix is represented by $v, 0 \le i \le n-1$, and n stands for the size of the matrix can be implemented as

3. 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. $\forall i \in [0..n-1]$: $u[i]:= v[i] + t[i]$

4 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.
   $\forall i,j \in [0..n-1]$: $u[i,j]= \Sigma k=0..n-1 \; v[i,k] * t[k,j]$

Class

The diagonal matrix type is worked out as a class. The size of the matrix can be set through the constructor. At the operations, the matrices have to have the same size, otherwise the program should throw an exception.


Matrix
- vec: int [0..n-1]
- resize(int) : void
+Matrix(int)
+getIndices(int,int) : int {query}
+setIndices(int,int,int) :void
+get_size(): int {query}
+ add (Matrix, Matrix) : Matrix { query }
+ mul (Matrix, Matrix) : Matrix { query }
+ write() : void { query }
+ read() : void
Exceptions
+OVERINDEXED
+NULLPART
+DIFFERENT
+INVALID