Documentation

Object-oriented programming

Assignment 1, Task 3

Name: Sofiia Isakova

Neptune code: A1MVYY

Email address: a1mvyy@inf.elte.hu

Group: 9

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).

Set type

Values

NMatrix(n) = {a ∈ z_{nxn} | $\forall i, j \in [1..n]: j \neq 1 \ \vee j \neq n \vee i \neq j \ \rightarrow a[i,j] = 0\}$

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]
   Formally: A : NMatrix(n) × $\mathbb{Z}$ × $\mathbb{Z}$ × $\mathbb{Z}$

   a i j e

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

   Post = (Pre ∧ e=a[i,j])

   This operation needs any action only if j = 1 or j =n or j = i, otherwise the output is zero.

2. **Setting an entry**
   Setting the entry of the ith column and jth row (i,j∈[1..n]): a[i,j]:=e.
   Entries outside of (i = j or j=1 or j = n) cannot be modified .
   Formally: A = NMatrix(n) × $\mathbb{Z}$ × $\mathbb{Z}$ × $\mathbb{Z}$

   a i j e

Pre = (e=e' $\wedge$ a=a' $\wedge$ i=i' $\wedge$ j=j' $\wedge$ i,j$\in$[1..n] $\wedge$ (j = 1$\vee$ j = n $\vee$j =i)

Post = (e=e' $\wedge$ i=i' $\wedge$ j=j' $\wedge$ a[i,j]=e $\wedge$ $\forall$k,l$\in$[1..n]: (k≠i $\vee$ l≠j)→ a[k,l]=a'[k,l])

This operation needs any action only if j = 1 or j =n or j = i, 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 = NMatrix(n) × NMatrix(n) × NMatrix(n)
   a b c

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

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

4. **Multiplication**

   Multiplication of two matrices: c:=a*b.

   The matrices have the same size.

   Formally: A = NMatrix (n) × NMatrix (n) × NMatrix (n)
   a b c

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

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

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

Representation

Only the matrix which fulfil necessary requirements (j = 1 or j =n or j = i) n×n has to be stored.

a11 0 0 0 .... an1

a12 a22 0 0 .... an2

a13 0 a33 0 .... an3 ↔ numList = <a11, a12, a13, a1n, a22, a33, an1, an2, an3, ann>

....

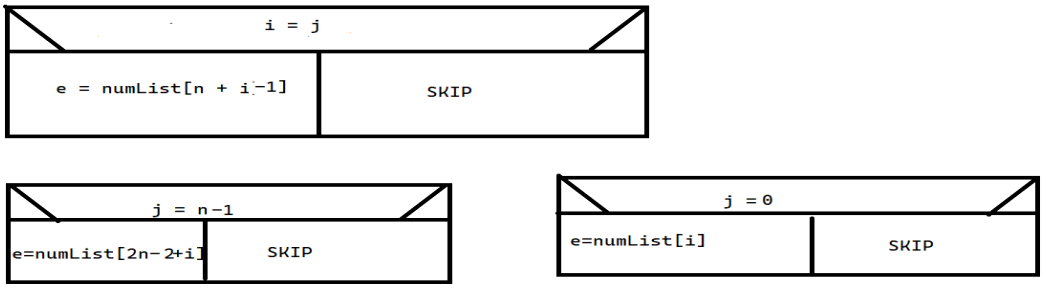
a1n 0 0 0 .... ann

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

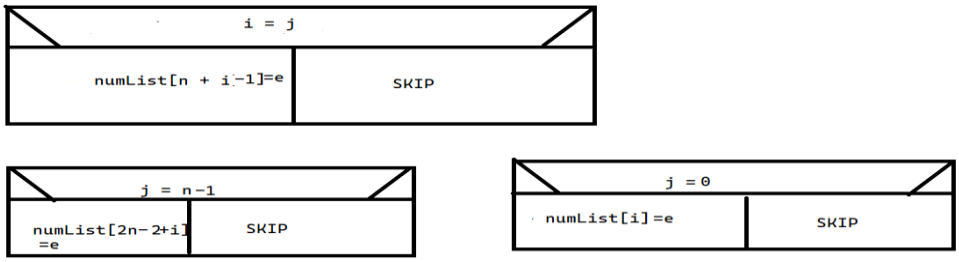
$$a[i,j] = \begin{cases} numList[i] & \text{if } j = 0 \\ numList[n + n - 2 + i] & \text{if } j = n-1 \\ numList[n + i - 1] & \text{if } j = i \end{cases} \qquad \text{where I, j indexes } i,j \in [0..n-1]$$

Implementation

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



2. Setting an entry
   Setting the entry of the ith column and jth row (i,j∈[0..n-1]) a[i,j]:=e where the matrix is represented by numList,0≤i≤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 v) 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 v), where all of the arrays have to have the same size.
   $\forall i \in [0..n-1]$: $\forall j \in [0..n-1]$: j=i v j=n-1 v j=0: $\forall k \in [0..n-1]$: v[i, j] := $k\Sigma$u[i,k] * t[k, j]

Testing

Testing the operations (black box testing)

1) Setting and getting matrix.

a) Creating empty matrix and checking if it is filled with 0

b) setting elements and checking if elements changed.

2) Creating, reading, and writing matrices of different size.

a) 0, 1, 3, 4-size matrix

3) Comparing matrixes

a) Executing command b=a for matrices a and b (with same size), comparing the entries of the two matrices. Then, changing one of the matrices and comparing the entries of the two matrices.

b) Executing command a=a for matrix a.

4) Sum of two matrices, command c:=a+b.

a) Checking correctness of summation.

b) Checking the neutral element (a + 0 == a, where 0 is the null matrix)

c) Checking the commutativity (a + b == b + a)

d) Checking the associativity (a + b) + c == a + (b + c)

e) With matrices of different size (size of a and d differs)

5) Multiplication of two matrices, command c:=a*b.

a) Checking correctness of multiplication.

b) Checking the neutral element (a * 0 == 0, where 0 is the null matrix).

c) Comparing element of matrix b with a*b element.

d) Checking the associativity a * (b * c)) == (a * b) * c

e) With matrices of different size (size of a and d differs)