OOP Khongorzul Khenchbish: Documentation, 1st assignment 1.

Khongorzul Khenchbish
R8HP78
r8hp78@inf.elte.hu
Group 1

23rd March 2021

Task

Implement the block matrix type which contains integers. These are square matrices that can contain nonzero entries only in two blocks on their main diagonal. Let the size of the first and second blocks be b1 and b2, where 1≤b1,b2≤n-1 and b1+b2=n (in the example, b1=4 and b2=5). Don't store the zero entries. Store only the entries that can be nonzero in a sequence or two smaller matrices. Implement as methods: getting the entry located at index (i, j), adding and multiplying two matrices, and printing the matrix (in a square shape).

Block matrix type

Set of values

Block(k,l) = $\{a \in \mathbb{Z}^{(k+l)x(k+1)} | \forall i,j \in [1..k+l]: (i < k \wedge j >= k) \vee (i >= k \wedge j < k) \rightarrow a[i][j] = 0\}$

Operations

1. Getting an entry

Getting the entry of the *i* th row and *j* th column $(i,j \in [1..k+l])$: $e := a[i,j]$.

Formally: A: *Block(k, l)* × ℤ × ℤ × ℤ
$\quad a \quad\quad i \quad j \quad e$
Pre = (a=a'∧ i=i'∧ j=j'∧ i,j∈[1..k+l])
Post = (Pre∧e=a[i,j])

This operation needs any action only if $(i < k \wedge j < k) \vee (i >= k \wedge j >= k)$, otherwise the output is zero.

2. Sum

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

Formally: A = Block(k, l) × Block(k, l) × Block(k, l)

$$\underset{a}{\text{Block}(k,l)} \times \underset{b}{\text{Block}(k,l)} \times \underset{c}{\text{Block}(k,l)}$$

Pre = (a=a'∧b=b')

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

In case of block matrices there is another version:

∀i, j∈[1..k]: c[i,j] = a[i,j] + b[i,j] and

∀i, j∈[k..l]: c[i,j] = a[i,j] + b[i,j] and

∀i, j∈[1..k+l]: $(i < k \wedge j >= k) \vee (i >= k \wedge j < k)$ → c[i,j]=0.

4. Multiplication

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

Formally: A = Block(k,l) × Block(k,l) × Block(k,l)

$$\underset{a}{\text{Block}(k,l)} \times \underset{b}{\text{Block}(k,l)} \times \underset{c}{\text{Block}(k,l)}$$

Pre = (a=a' ∧ b=b')

Post = (Pre ∧ ∀i,j∈[1..k+l]: c[i,j]= $\sum_{m=1..k+l}$ a[i,m] * b[m,j])

In case of block matrices there is another version:

∀i, j∈[1..k] and ∀*m*∈[*1..k*]: c[i,j] = $\sum_{m=1..k}$ a[i,m] * b[m,j] and

∀i, j∈[k..l] and ∀*m*∈[*k..l*]: c[i,j] = $\sum_{m=k..l}$ a[i,m] * b[m,j] and

∀i, j∈[1..k+l]: $(i < k \wedge j >= k) \vee (i >= k \wedge j < k)$ → c[i,j]=0.

Representation

Only the nonzero blocks of the $(k + l)x(k + l)$ matrix has to be stored.

$$a = \begin{matrix} a_{11} & ... & a_{1k} & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... \\ a_{k1} & ... & a_{kk} & 0 & ... & 0 \\ 0 & ... & 0 & a_{ll} & ... & a_{l(l+k)} \\ ... & ... & ... & ... & ... & ... \\ 0 & ... & 0 & a_{(k+l)l} & ... & a_{(k+l)(k+l)} \end{matrix}$$

We need two 2-dimensional array m1 and m2 is needed, element of the block matrix can be get:

$$a[i, j] = \{ \begin{matrix} m1[i][j] & if\ (i < k \wedge j < k) \\ a[i, j] = m2[i - k][j - k] & if\ (i >= k \wedge j >= k) \\ 0 & if\ (i < k \wedge j >= k) \vee (i >= k \wedge j < k) \end{matrix} \}$$

Implementation

1. Getting an entry

Getting the entry of the ith column and jth row (i,j∈[1..k+l]) e:=a[i,j] where the matrix is represented by smaller block m1 and m2 matrices can be implemented as

$i < k \wedge j < k$	$i >= k \wedge j >= k$	$(i < k \wedge j >= k) \vee (i >= k \wedge j < k)$
$e := m1[i][j]$	$e := m2[i - k][j - k]$	$e := 0$

3. Sum

The sum of matrices a and b goes to matrix c, where all of the corresponding smaller matrices m1 and m2 are added and arrays have to have the same size.

```
i, j∈[0..k+l-1]
    $i < k \wedge j < k$
        c.m1[i,j] := a.m1[i,j]+b.m1[i,j]
    $i >= k \wedge j >= k$
        c.m2[i-k,j-k] := a.m2[i-k,j-k]+b.m2[i-k,j-k]
    $(i < k \wedge j >= k) \vee (i >= k \wedge j < k)$
        SKIP
```

4. Multiplication

The product of matrices a and b goes to matrix c where all of the corresponding smaller matrices m1 and m2 are multiplied and have to have the same size.

```
i, j∈$[0..k-1]$
    m∈ $[0..k-1]$, sum := 0
        sum := sum + a.m1[i,m]+b.m1[m,i]
    c.m1[i,j] := sum
```

```
i, j∈$[0..l-1]$
    m∈ $[0..l-1]$, sum := 0
        sum := sum + a.m2[i,m]+b.m2[m,i]
    c.m2[i,j] := sum
```

Testing

Testing the operations (black box testing)

1) Creating, reading, and printing matrices of different size.
   a) (2,1) (1,1) (2,3) -size matrix

2) Getting an entry
   a) Getting an entry in the smaller blocks
   b) Getting an entry outside the blocks
   c) Illegal index, indexing a 0-size matrix

3) Copy constructor
   a) Creating matrix b based on matrix a, comparing the entries of the two matrices.

4) Assignment operator
   a) Executing command b=a for matrices a and b (with and without same size), comparing the entries of the two matrices.
   b) Executing command c=b=a for matrices a, b, and c (with and without same size), comparing the entries of the three matrices.
   c) Executing command a=a for matrix a.

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

6) Multiplication of two matrices, command c:=a*b.
   a) With matrices of different size (size of a and b differs)
   b) Checking if it is working c == a*b
   d) Checking the neutral element (a * 0 == 0, where 0 is the null matrix)
   e) 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,-2), (0,0), (1,1) (1000,1000)).
2. Generating and catching exceptions.