Muhammad Eman Aftab IJE4R1 ije4r1@inf.elte.hu Group 1

1. assignment/4.task

17th March 2024

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=2 and b2=4). 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

$Block\_Matrix(n,b1,b2) = \{ a \in \mathbb{Z}^{n\times n} \mid b1,b2 \in \mathbb{Z} \wedge b1+b2 = n, \forall i,j\in[1..n]: (i >= b1 \vee j >= b1)) \wedge (i < b1 \vee j < b1 \vee i >= N \vee j >= N) \rightarrow a[i,j]=0 \}$

Operations

1. Getting an entry

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

Formally:
A: (a:*BlockMatrix*, i:ℤ, j:ℤ, e:ℤ)

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

$Post = (Pre \wedge e=a[i,j])$

This operation needs any action only if $(i < b1 \vee j < b1) \wedge (i >= b1 \vee j >= b1 \vee i < N \vee j < N)$, otherwise the output is zero.

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

Formally: A :(a:*BlockMatrix,* b:*BlockMatix.* c:*BlockMatix*)

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

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

There is an easier version, In case of block matrices, we just add the corresponding square matrices within the block matrices. All the other entries are zeros.

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

Formally: A: (a:*BlockMatrix,* b:*BlockMatix.* c:*BlockMatix*)

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

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

There is an easier version In case of block matrices, we just multiply the corresponding square matrices within the block matrices. All the other entries are zeros.

Representation

Only the Block matrices of the $n \times n$ matrix have to be stored.

$$a = \begin{matrix} a_{11} & \dots & a_{1b1} & 0 & \dots & 0 \\ : & \ddots & : & 0 & \dots & 0 \\ a_{b1\,1} & 0 & a_{b1\,b1} & 0 & \dots & 0 \\ 0 & \dots & 0 & a_{(b1+1)(b1+1)} & \dots & a_{(b1+1)\,(b1+b2)} \\ 0 & \dots & 0 & : & \ddots & : \\ 0 & \dots & 0 & a_{(b1+b2)(b1+1)} & \dots & a_{(b1+b2)(b1+b2)} \end{matrix} \qquad v = < a_{11}\ a_{1b1}\ a_{b1\,1}\ a_{b1b1}\ a_{(b1+1)(b1+1)}\ a_{(b1+1)(b1+b2)}\ a_{(b1+b2)(b1+1)}\ a_{(b1+b2)(b1+b2)} >$$

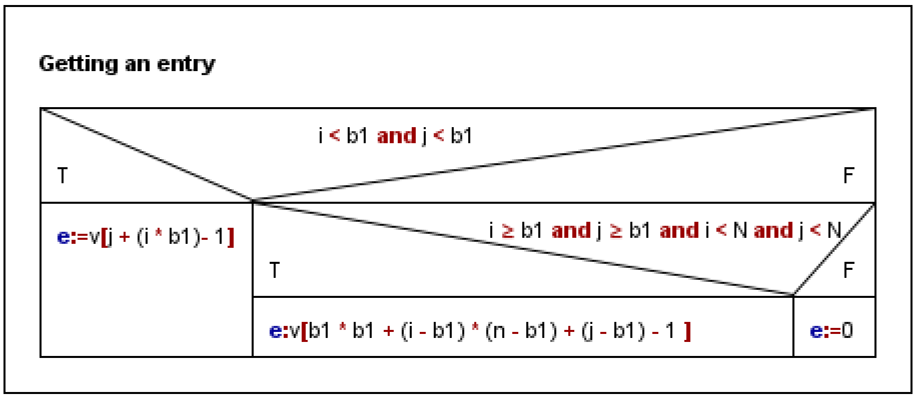
Only a one-dimension list (v) is needed, with the help of which any entry of the block matrices can be get:

$a[i, j]$ = {v[j+(i*b1)]} *if* (i < b1 ^ j < b1)
v[(b1 * b1 + (i - b1) * (n - b1) + (j - b1))] if (i >= b1 ^ j >= b1 ^ i < N ^ j < N)
0 *if* not ((i < b1 ^ j < b1) *v* *(*(b1 * b1 + (i - b1) * (n - b1) + (j - b1))] if (i >= b1 ^ j >= b1 ^ i < N ^ j < N)*))*

Implementation[1]

1. Getting an entry

Getting the entry of the *i*th column and *jth* row ($i,j \in [1..n]$) $e{:=}a[i,j]$ where the matrix is represented by v, $1 \le i \le (b1*b1)+(b2*b2)$,, and n stands for the size of the matrix, n which is equal to b1+b2 can be implemented as



```
Getting an entry
if i < b1 and j < b1
  T: e:=v[j + (i * b1)- 1]
  F: if i ≥ b1 and j ≥ b1 and i < N and j < N
       T: e:v[b1 * b1 + (i - b1) * (n - b1) + (j - b1) - 1 ]
       F: e:=0
```

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.

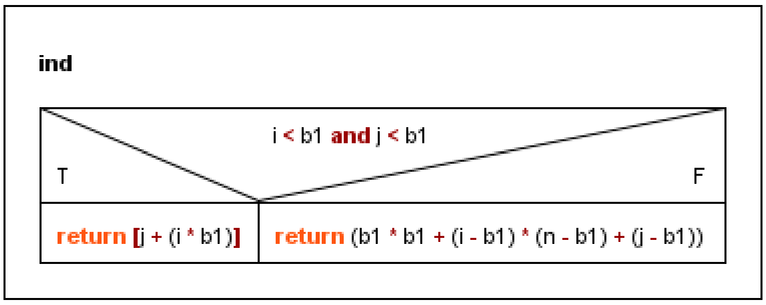
$$\forall i \in [0..(b1*b1)+(b2*b2)]: u[i] := v[i] + t[i]$$

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.

$$\forall i \in [0..b1], \forall j \in [0..b1], \forall k \in [0..b1]: u[i] {:+=}\, v[ind(i,k)] * t[ind(k,j)]$$

$$\forall i \in [b1..b2], \forall j \in [b1..b2], \forall k \in [b1..b2]: u[i] {:+=}\, v[ind(i,k)] * t[ind(k,j)]$$



```
ind
if i < b1 and j < b1
  T: return [j + (i * b1)]
  F: return (b1 * b1 + (i - b1) * (n - b1) + (j - b1))
```

[1] To implement an operation, a program has to be given (not necessarily structogram).

Testing

Testing the operations (black box testing)

1) Creating, reading, and writing matrices of different size.
   a) 0, 4, 3, 5-size matrix
2) Getting entry
   a) Getting an entry in the block matrices
   b) Illegal size of matrix, indexing a -1 size matrix
   c) Illegal block matrices size, checking when sum of b1 or b2 not equal n, n=4 , b1=2, b2=1

3) Sum of two matrices, command *c*:=*a*+*b*.
   a) With matrices of different size (size of *a* and *b* differs, size of *c* and *a* 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 + 0 == a, where 0 is the null matrix)
   e) Checking addition not possible when matrices have different size

4) Multiplication of two matrices, command *c*:=*a***b*.
   a) With matrices of different size (size of *a* and *b* differs, size of *c* and *a* differs)
   b) Checking the commutativity with identity matrix (a * b == b * a)
   c) Checking the associativity (a * b * c == (a * b) * c == a * (b * c))
   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)
   f) Checking addition not possible when matrices have different size
5) Checking Index Function giving correct address of list with two indexes like 2D array.

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