OOP 1st Assignment 1 - Documentation

Deepak Kumar Upadhayay **1. assignment/1st task** 5th March 2022

DD79DY
dku9751@gmail.com // dd79dy@inf.elte.hu
Group 5

TASK

Implement the chessboard matrix type which contains integers. In these matrices, every second entry is zero. The entries that can be nonzero are located like the samecolored squares on a chessboard, with indices (1, 1), (1, 3), (1, 5), ..., (2, 2), (2, 4), ... The zero entries are on the indices (1, 2), (1, 4), ..., (2, 1), (2, 3), ... Store only the entries that can be nonzero in row-major order in a sequence. Don't store the zero entries. Implement as methods: getting the entry located at index (i, j), adding and multiplying two matrices, and printing the matrix (in a shape of m by n).

Chess matrix type

Set of values

cm1(A) = { $a \in Z^{N*N}$ | $\forall$i,j$\in$[1..n]: i mod 2 = 0 && j mod 2 = 0 || i mod 2 = 1 && j mod 2 = 1

$\rightarrow$ a[i,j] != 0 }

Operations

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

Formally: $\boldsymbol{A = cm1(N) \times Z \times Z \times Z}$
$\qquad\qquad\qquad\boldsymbol{a \qquad\quad i \quad j \quad e}$

$$\boldsymbol{Pre = (\ a=a' \wedge\ i=i' \wedge\ j=j' \wedge\ i,j \in [1..N]\ )}$$

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

This operation needs any action only i mod 2 = 0 && j mod 2 = 0 || i mod 2 = 1 && j mod 2 = 1, otherwise the output is zero.

2. Setting an entry
Setting the entry of the *i*th column and *jth* row ($i,j\in[1..(b1+b2)]$): $a[i,j]:=e$. Entries outside the blocks cannot be modified $((i<b1\&\&j<b2)||(i>b1\&\&j>b1))$.

Formally: $A = \underset{a}{cm1(N)}\times \underset{i}{Z}\times \underset{j}{Z}\times \underset{e}{Z}$

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

$$((i \bmod 2 = 0\ \&\&\ j \bmod 2 = 0)\ ||\ (i \bmod 2 = 1\ \&\&\ j \bmod 2 = 1))\ )$$

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

This operation needs any action only if only i mod 2 = 0 && j mod 2 = 0 || i mod 2 = 1 && j mod 2 = 1 , 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: $A = \underset{a}{cm1(N)}\times \underset{b}{cm1(N)}\times \underset{c}{cm1(N)}$

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

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

We can ignore the above calculation if $i>N\ ||\ j<0\ ||\ i < 0\ ||\ j>N$

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

Formally: $A = \underset{a}{cm1(N)}\times \underset{b}{cm1(N)}\times \underset{c}{cm1(N)}$

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

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

We can ignore the above calculation if $i>N\ ||\ j<0\ ||\ i < 0\ ||\ j>N$

Representation

Only the alternate index of the $n\times n$ matrix has to be strored.

For example here we have matrix of size 4 (Dmatrix(4,2)) :

$$A = \begin{matrix} A11 & 0 & A13 & 0 & A15 \\ 0 & A22 & 0 & A24 & 0 \\ A31 & 0 & A33 & 0 & A35 \\ 0 & A42 & 0 & A44 & 0 \\ A51 & 0 & A53 & 0 & A55 \end{matrix} \quad \leftrightarrow \quad v = < A_{11}\, A_{13}\, A_{15}\, A_{22}\, A_{24}\, A_{31}\, A_{33}\, A_{35}\, A_{42}\, A_{44}\, A_{51}\, A_{53}\, A_{55} >$$

Only a one-dimension array (v) is needed, with the help of which any entry of the Chess matrix can be done:

$$a[i,j] = \begin{cases} \text{v[ind(i,j)]}, & if\ \ ((i\%2\ =\ 0\ \ j\%2\ =\ 0)\ ||\ (i\%2\ =\ 1\ \ j\%2\ =\ 1)) \\ & 0 \end{cases}$$

The total number of elements in the vector will be { *[(square of size of matrix) + 1] divided by 4* } always.

*(ind(i,j) calculates the indices of a array by row and column number of the matrix.)

Illustration and implementation of ind(i,j) function:

size%2 == 0	
((i-1)*size/2) + ((j+1)/2)	((i-1)*size/2) + ((j+2)/2)

-

Implementation

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 $v, 1 \le i,j \le (N)$, and N stands for the size of the Chess matrix can be implemented as:-

Here inOrder is a private method to check either the given index is in order or not.

inOrder(i,j)	
e=_vec[(ind(i,j,size))-1]	--

```
int cm1::getElement(int i, int j) const
{
    int size = msize();
    if(i <= size && j <= size)
    {
    if (inOrder(i,j))
    {
        return _vec[(ind(i,j,size))-1];
    }
    else
    {
        return 0;
    }
    }
    else
    {
        throw OVERINDEXED;
    }

}
```

2. Setting an entry

Setting the entry of the *i*th column and *jth* row ($i,j \in [1..N]$) $a[i,j]:=e$ where the matrix is represented by $v, 1 \le i,j \le (N)$, and Nstands for the size of the Chess matrix can be implemented as:-

<table>
<tr><td colspan="3">a <= size && b <= size</td></tr>
<tr><td colspan="2">inOrder(a,b)</td><td rowspan="2"></td></tr>
<tr><td>v[ind(i,j)]:=e</td><td>SKIP</td></tr>
</table>

```
void cm1::setElement(int a,int b,int c)
{
    int size = msize();
    if((a <= size && b <= size ) && (a > 0 && b > 0))
    {
         if (inOrder(a,b))
        {
            _vec[ind(a,b,size)-1] = c;
        }
        else if(!inOrder(a,b) && c == 0 )
        {

        }
        else
        {
            throw INVALID;
        }
    }
    else
    {
        throw OVERINDEXED;
    }

}
```

3. Sum

The sum of matrices a and b (represented by vectors t and u) goes to matrix c (represented by vector u), where all of the arrays have to have the same size. $\forall i \in [0..n-1]$: u[i]:= v[i] + t[i]

```
cml operator+(const cml& a ,const cml& b)
{
    int s = a.getSize();
    if(s != b.getSize()) throw cml::DIFFERENT;

    cml c(s);

    for(int i = 0; i < s; ++i)  c._vec[i] = a._vec[i] + b._vec[i];
    return c;
}
```

4. Multiplication

The product 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 :-

$$\forall i,j \in [1..N]: c[i,j] = \Sigma_{k=1..N}\, a[i,k] * b[k,j]$$

```
cml operator*(const cml& a ,const cml& b)
{
    int s = a.getSize();
    if(s != b.getSize())
    {
        throw cml::DIFFERENT;
    }
    else
    {
    int n = sqrt((a._vec.size())*2);
    cml m(s);
    for(int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            int c = 0;
            for (int k = 1; k <= n; k++)
            {
                c = c + a.getElement(i,k)*b.getElement(k,j);
                 m.setElement(i,j,c);
            }
        }
    }
    return m;
    }
}
```

Testing

Testing the operations (black box testing)

1) Creating, reading, and writing matrices of different size.
    a) 0, 1, 2, 5-size matrix

2) Getting and setting an entry
    a) Getting and setting an entry in the chess matrix
    b) Getting and setting an entry outside the chess matrix
    c) Illegal index, indexing a 0-size matrix

5) 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)

6) 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 (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)
    e) Checking the identity element (a * 1 = a)

Testing based on the code (white box testing)

1) Creating an extreme-size matrix (-1, 0, 1, 1000).
2) Generating and catching exceptions.