

## TOPIC: Matrix Multiplication

### OVERVIEW

In mathematics, a matrix is an 2-dimensional  $m$  by  $n$  table of numbers, where  $m$  and  $n$  are positive integers. We can multiply a matrix  $A$  with a matrix  $B$  only if  $A$  is  $m$  by  $n$  and  $B$  is  $n$  by  $p$ , for some positive integers  $m$ ,  $n$ , and  $p$ . We calculate the product  $AB$  in the following way: for all values  $0 \leq i < m$  and for all values  $0 \leq j < n$ , multiply each element in the  $i$ th row of  $A$  by its corresponding element in the  $j$ th column of  $B$ , add each product together, and store it row  $i$ , column  $j$  of  $AB$ . The resulting matrix  $AB$  will have dimensions  $m$  by  $p$ . For example:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad B = \begin{bmatrix} 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{bmatrix}, \quad AB = \begin{bmatrix} 50 & 68 \\ 122 & 167 \\ 194 & 266 \end{bmatrix}$$

The cell at row 0, column 0 of  $AB$  is 50 because  $50 = 1*7 + 2*8 + 3*9$ . This is because row 0 of  $A$  is  $[1 \ 2 \ 3]$  and column 0 of  $B$  is  $[7 \ 8 \ 9]$ . Similarly, the cell at row 1, column 1 in  $AB$  is 167 because  $167 = 4*10 + 5*11 + 6*12$ .

It is important to note here that we calculated the product  $AB$ , not  $BA$ . In fact, we can't calculate  $BA$  in this case, since the number of columns in  $B$  is not equal to the number of rows in  $A$  (but the number of columns in  $A$  is equal to the number of rows in  $B$ ).

### PROGRAMMING

We want you to develop a program that can read two matrices from `StdIn`, and if they are of the right dimensions, calculate and print out their product. There are two classes you will need to make: the `Matrix` class, and the `MatrixMultiplier` class. We are providing a partially filled-in `Matrix` class for you. All you need to implement in `Matrix` are the constructors, `setCell()`, `canMultiply()`, and `multiply()`.

We give you the skeleton for `Matrix` here: <http://www.cs.princeton.edu/~jcnelson/Matrix.java>

The `Matrix` class should implement the following API:

```
public class Matrix
-----
    Matrix(int m, int n)           // Create a Matrix with all 0's, but
                                   // with  $m$  rows and  $n$  columns.

    Matrix(double[][] data)       // Create a Matrix which will store the
                                   // doubles given in data. We want
                                   // data[i][j] to be the element in the
                                   // matrix at the  $i$ th row and  $j$ th column.
                                   // You may assume all of the rows in
                                   // data are the same length.

    int getNumRows()              // Return the number of rows in this
                                   // Matrix.
```

```

    int getNumColumns()           // Return the number of columns in this
                                // matrix.

    double getCell(int i, int j) // Get the value at row i, column j of
                                // the Matrix.

    void setCell(int i, int j, double v) // Put the value v at row i, column j of
                                        // the Matrix.

    boolean canMultiply( Matrix m ) // Return true if "this" Matrix can be
                                    // multiplied by Matrix m; false if not.

    Matrix multiply( Matrix m ) // Given another Matrix m, calculate
                                // "this" Matrix multiplied by Matrix m
                                // (as opposed to Matrix m multiplied by
                                // "this" Matrix). Put the values of
                                // the product into a new Matrix
                                // instance, and return it. Throw a
                                // RuntimeException if "this" Matrix
                                // cannot be multiplied by m.

    void show()                  // Print this Matrix to StdOut.

```

The `MatrixMultiplier` class should have a `main()` method that does the following:

- Read *m* and *n* from `StdIn`
- Read *m* sets of *n* doubles from `StdIn`, and create a `Matrix` to store them.
- Do the above two steps again to read a second `Matrix` from `StdIn`
- If the matrices can be multiplied, multiply them and print the product to `StdOut`. Otherwise, print an error message indicating that the matrices cannot be multiplied.

## EXAMPLES

Here's the original example:

```

jude@t510:~/cos126/practiceTest$ more matrix1.txt
3 3
1 2 3
4 5 6
7 8 9
3 2
7 10
8 11
9 12
jude@t510:~/cos126/practiceTest$ java MatrixMultiplier < matrix1.txt
[ 50.0000 68.0000 ]
[ 122.0000 167.0000 ]
[ 194.0000 266.0000 ]

```

Another example:

```
jude@t510:~/cos126/practiceTest$ more matrix2.txt
5 6
3 1 4 1 5 9
2 6 5 3 5 8
2 7 1 8 2 8
-1 -1 -3 -5 -8 -13
0.1 0.3 0.5 0.7 1.1 100
6 1
-0.1
-3
2
5
10
1.715
jude@t510:~/cos126/practiceTest$ java MatrixMultiplier < matrix2.txt
[ 75.1350 ]
[ 70.5200 ]
[ 54.5200 ]
[ -130.1950 ]
[ 186.0900 ]
```

An example of incompatible matrices:

```
jude@t510:~/cos126/practiceTest$ more matrix3.txt
3 2
7 10
8 11
9 12
3 3
1 2 3
4 5 6
7 8 9
jude@t510:~/cos126/practiceTest$ java MatrixMultiplier < matrix3.txt
Cannot multiply matrices--incompatible dimensions
jude@t510:~/cos126/practiceTest$
```

## ADVICE

- DON'T PANIC—this is a practice test. It's not a referendum on your future testing abilities.
- Look at the methods we gave you to see how the methods you will write can be used.
- Remember pass-by-reference and pass-by-value, and how it applies to arrays.
- The files matrix1.txt, matrix2.txt, and matrix3.txt can be found here:
  - <http://www.cs.princeton.edu/~jcnelson/matrix1.txt>
  - <http://www.cs.princeton.edu/~jcnelson/matrix2.txt>
  - <http://www.cs.princeton.edu/~jcnelson/matrix3.txt>