groupproject

Connor Baker
Rae Bouldin

November 27, 2016
Version 0.2a

Contents

1 Summary of Problem Specification

1.1 Abstract

Write a program that reads two 3×3 matrices from file and computes the sum and product of the two matrices. Then, find the transpose, cofactor matrix, and determinant of the two resultant matrices. Then, find the inverse of the first matrix, and multiply it by the the first column of the second matrix. Finally, compute the standard deviation of the diagonal elements of the two inputted matrices. All input and output should be stored in files.

2 Formulae

2.1 Determinant

The determinant of a 3×3 matrix is most readily computed by row reducing to a triangular matrix, and taking the product of the main diagonal. However, failing that, one can calculate the determinant by doing cofactor expansion. Though a horribly inefficient algorithm for larger matrices, it gets the job done. For a 3×3 matrix A, it's determinant, $\det(A)$, can be computed using placeholder values as follows:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$\det(A) = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} \tag{1}$$

As such, we can calculate the determinant of a 3×3 matrix by expanding across the top row. This yields the approach implemented in `determinantOfMatrix()`, detailed in Section X.X.X.

2.2 Transpose

Using the same approach as above, we can easily compute the transpose of a matrix. It involves creating a matrix filled with placeholder values, calculating the transpose matrix by hand, and tracking the positions of the elements in the matrix. For a 3×3 matrix A, let its transpose be A^T. Then:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$A^T = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \tag{2}$$

This yields the approach implemented in `transposeOfMatrix()`, detailed in Section X.X.X.

2.3 Matrix Addition

The approach used to create an algorithm for matrix addition is similar to that used above in finding the determinant. We again create matrices full of placeholder values, and track them as we perform the operation. For two 3×3 matrices A and B, the process is as follows:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \qquad B = \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix}$$

$$A + B = \begin{bmatrix} a+j & b+k & c+l \\ d+m & e+n & f+o \\ g+p & h+q & i+r \end{bmatrix} \tag{3}$$

This yields the approach implemented in `sumOfMatrices()`, detailed in Section X.X.X.

2.4 Matrix Multiplication

The approach used to create an algorithm for matrix multiplication is similar to that used above in finding the determinant. We again create matrices full of placeholder values, and track them as we perform the operation. For two 3×3 matrices A and B, the process is as follows:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \qquad B = \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix}$$

$$AB = \begin{bmatrix} aj+bm+cp & ak+bn+cq & al+bo+cr \\ dj+em+fp & dk+en+fq & dl+eo+fr \\ gj+hm+ip & gk+hn+iq & gl+ho+ir \end{bmatrix} \tag{4}$$

This yields the approach implemented in `productOfMatrices()`, detailed in Section X.X.X.

2.5 Cofactor Matrix

The approach used to calculate the cofactor matrix is identical to what was done above. It involves creating a matrix filled with placeholder values, calculating the cofactor matrix by hand, and tracking the positions of the elements

in the matrix. For a 3×3 matrix C, let its cofactor matrix be C'. Then:

$$C = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$C' = \begin{bmatrix} ei - fh & fg - di & dh - eg \\ ch - bi & ai - cg & bg - ah \\ bf - ce & cd - af & ae - bd \end{bmatrix} \tag{5}$$

This yields the approach implemented in `cofactorOfMatrix()`, detailed in Section X.X.X.

2.6 Inverse of a Matrix

The inverse of a matrix A, denoted A^{-1} can be calculated as follows:

$$A^{-1} = \frac{1}{\det(A)} * A^T \tag{6}$$

Which is very doable using Equations (1) and (2). This approach is used in `inverseOfMatrix()`, detailed in Section X.X.X.

3 Main.class

4 Matrix.class

4.1 Matrix()

4.2 readMatrixFromFile()

4.3 determinantOfMatrix()

4.4 transposeOfMatrix()

4.5 sumOfMatrices()

4.6 productofMatrices()

4.7 cofactorOfMatrix()

4.8 inverseOfMatrix()

4.9 print()

4.10 printMatrixToConsole()

4.11 printMatrixToFile()

4.12 printIntegerToFile()

5 Notes

5.1 A Note About the Methods

Instead of using matrix.length and matrix[i].length in all of the for loops, I've decided to use the constant 3, because this project deals only with matrices of the third order. I believe that using a constant instead of referencing the size of the array (which in of itself requires passing the array to another method and having java tally the length) provides a meaningful speedup.

References

http://download.java.net/java/jdk9/docs/api/