Lab 2 Analysis - Noboru Hayashi

I. Data Structure

In this lab assignment the program is required to calculate a determinant of a matrix which is read from a text file. While designing the program, I chose a 2-D int array to store the matrix, since the input file has an order of the matrix in the next couple lines. The information of the order allows the program to initialize the matrix as: int[][] mat = new int[order][order];

Also while calculating the determinant of the matrix, calcDeterminant (int[][] mat) function creates sub-matrices by initializing square matrices with a lower order (order-1). So it's more direct and simple to use the nested array to implement the program than using other data structures.

II. Iteration vs Recursion

With an iterative implementation, the program can run iterative calculations given the order of the matrix. More precisely, the program can use the Leibniz formula as below to calculate the determinant:

$$det(A) = \sum_{\sigma \in S_n} \left(sgn(\sigma) \prod_{i=1}^{n} a_{i,\sigma_i} \right)$$

This formula computes the sum over all permutations σ of the set {1,2, … , order}. If the order is three, Sn would be {[1,2,3], [1,3,2], [2,1,3],[2,3,1], [3,1,2],[3,2,1]}. So the determinant of a 3x3 matrix is:

$$= sgn([1,2,3]) \prod_{i=1}^{n} a_{i,[1,2,3]_i} + sgn([1,3,2]) \prod_{i=1}^{n} a_{i,[1,3,2]_i} + sgn([2,1,3]) \prod_{i=1}^{n} a_{i,[2,1,3]_i}$$

$$+ sgn([2,3,1]) \prod_{i=1}^{n} a_{i,[2,3,1]_i} + sgn([3,1,2]) \prod_{i=1}^{n} a_{i,[3,1,2]_i} + sgn([3,2,1]) \prod_{i=1}^{n} a_{i,[3,21]_i}$$

sgn() denotes the signature of the permutation, which is +1 if the permutation can be sorted by interchanging two entries an even number of times, and -1 for an odd number of such interchanges. Hence:

$$sgn([1,2,3]) = 1,\ sgn([1,3,2]) = -1,\ sgn([2,1,3]) = -1$$
$$sgn([2,3,1]) = 1,\ sgn([3,1,2]) = 1,\ sgn([3,2,1]) = -1$$

So the sum can be represented as:

$$= \prod_{i=1}^{n} a_{i,[1,2,3]_i} - \prod_{i=1}^{n} a_{i,[1,3,2]_i} - \prod_{i=1}^{n} a_{i,[2,1,3]_i} + \prod_{i=1}^{n} a_{i,[2,3,1]_i} + \prod_{i=1}^{n} a_{i,[3,1,2]_i} - \prod_{i=1}^{n} a_{i,[3,2,1]_i}$$

$$= a_{1,1} \cdot a_{2,2} \cdot a_{3,3} - a_{1,1} \cdot a_{2,3} \cdot a_{3,2} - a_{1,2} \cdot a_{2,1} \cdot a_{3,3}$$

$$+ a_{1,2} \cdot a_{2,3} \cdot a_{3,1} + a_{1,3} \cdot a_{2,1} \cdot a_{3,2} - a_{1,3} \cdot a_{2,2} \cdot a_{3,1}$$

Given this characteristic, the program can compute all permutations given the order, and access elements from the matrix to iteratively calculate each product term, then get the sum as the determinant. Since the calculation requires index accesses, the 2d array implementation would directly achieve the requirement.

Comparing two implementations, I think the recursive one is more easy to understand and effective to calculate the sign of each sub-determinant, since the iteration needs implement a function to decide the signature of the permutation.

III. Enhancement

This lab assignment requires the program to handle matrices up to and including order 6, I added some test cases that have more bigger sizes. Since the recursive method divides a base matrix into smaller sub-matrices and passes them to itself, the program can handle any size of the matrix.

IV. Conclusion

In this lab assignment, I learned how to utilize (2-D) array data structure, to operate recursive computations, in order to accomplish a complex mathematical task. However, some computing languages or libraries specialize in vector & matrix operations such as MatLab or Numpy, possibly have more efficient ways to calculate the determinant, I might do more investigations on how these programming accomplish this operation.