LA ASSIGNMENT REPORT

VISHAL GOEL - 15451 - MTECH CSA

Problem I, Bonus Question

If it is not possible to prepare the potion (i.e. no possible solution) because of limitation in ingredient quantity, can you tell the maximum amount of potion that can be prepared out of the given ingredients?

M : matrix of size n*k (n: number of students, k: number of ingredients)
V : vector of size k

If there is no solution (due to ingredients being limited), then

maximum amount of potion student S_i can make = M[i] . V
….{M[i] is the ith row of M and . is the dot product}

Thus, maximum amount of potion = $\mathbf{\Sigma}_{0<=i<=n-1}$ M[i] . V

Problem II, Second Part

You will have to report the time taken for the process to complete. Compare it with the standard library API available in Python and mention the difference. Can you find a way to improve the efficiency of your algorithm in terms of time taken?

Input size of matrix	Time taken by my code (sec)	Time taken by Numpy (sec)
10	0.025	5.9e-07
30	0.055	5.4e-07
50	1.577	0.157e-06

To improve efficiency of my algorithm (using Gauss Jordan Elimination):
Instead of directly performing Gauss Jordan Elimination(time complexity = $O(n^3)$) on the given matrix A, it is more efficient to follow the following steps:

1. Factorise the matrix into an LU decomposition. If not possible, then the inverse does not exist. Time Complexity is still $O(n^3)$ but the constant factor is less because of the zero entries in the matrices.
2. If LU decomposition was possible, then $A^{-1} = U^{-1} L^{-1}$. This step also takes $O(n^3)$ but quite less in comparison to the first step.

Problem II, Third Part

Consider you are given the same three operations, but this time instead of row operations you are allowed to do column operations. Can you find an inverse? Explain why or why not.

Yes, it is still possible to find an inverse, if at all it exists.

Suppose Col-op(matrix, op, indice(s) of columns, value(optional)) returns a matrix after performing a single column operation on it. Suppose we want to find inverse of a matrix A_{n*n}. Then,

1. Switch(A, row i, row j) is equivalent to
   (Col-op(A^T, switch, indices of columns = i and j))^T

2. Multiply(A, row i, c) is equivalent to
   (Col-op(A^T, multiply, index of column = i, c))^T

3. Multiply&Add(A, row i, row j, c) is equivalent to
   (Col-op(A^T, multiply&add, indices of columns = i and j , c))^T

Problem II, Bonus Question

Say you have only two out of the three operations at your disposal. Given a matrix, is it possible to get identity matrix? Also, what can you say about the complexity of the algorithm?

Switch(A, row i, row j) is equivalent to
Step1: Multiply&Add(A, row j, row i, 1)   i.e. row i = row j + row i
Step2: Multiply(A, row j, -1) followed by Multiply&Add(A, row i, row j, 1)   i.e. row j = row i - row j
Step3: Multiply&Add(A, row j, row i, -1)   i.e. row i = row i - row j

Multiply(A, row i, c) is equivalent to
Multiply&Add(A, row i, row i, c-1)

Multiply&Add(A, row i, row j, value c) cannot be replaced by a combination of Switch and Multiply

Thus, given an option of keeping only two operations, we need to keep Multiply&Add and one from Switch and Multiply to be able to find the inverse of a matrix.

<u>Time Complexity of the algorithm used in program to find matrix inverse:</u>
During the process of reducing a matrix of size n*n to row reduced echelon form, we try to reduce all entries in a pivot column to 0 using n row operations. Let a trivial operation be defined as an addition or multiplication. Then,

total trivial operations = number of pivots * number of elementary row operations per pivot * number of trivial operations per elementary row operation
<= n * n * n
= $O(n^3)$

In case we use only two row operations, number of elementary row operations per pivot can increase by a constant factor (1 or 4), thus asymptotic complexity remains the same.