“LA_Assignment_1 Report”

Name: ACHINT CHAUDHARY; Sr. No. 15879; M.Tech (Course)

Problem1 (Bonus Question)
YES, we are solving System of the form P*X = C, where 'P' is the percentage matrix & 'C' is vector representing the quantities to be prepared by each student. If we get some solution X, it should be checked element-wise to be less than or equal to with 'M', which is the vector denoting maximum available quantities in Inventory.

$$P*X = C \text{ and } X <= M$$

We solve for $X = P^{-1}*C <= M$

Multiplying Both sides by P $C <= P*M$

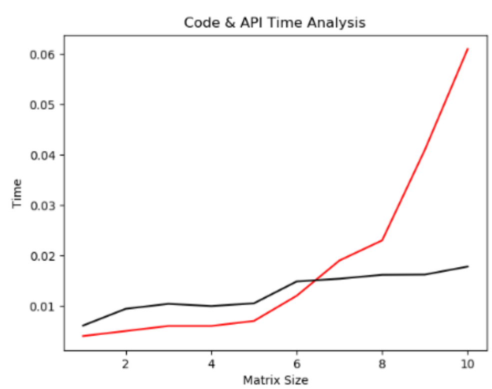
Maximum C can be obtained at $C_{max\_possible} = P*M$

In this part, if we need to find out maximum amount of potion that can be prepared, for this we can use 'M' in placed of Solution vector X itself, so maximum amount of Quantity can be Evaluated to be

$$C_{max\_possible} = P*M$$

Which is Matrix Multiplication of Percentage Matrix with Maximum Amount Vector

Problem 2 (Second Part)
Numpy is fast for finding Inverses of Large size matrices, the implementation I have done is considering operations in form of “Fractions” (not float), so that there is No loss of precision due to floating point representation errors & hence can be used in sensitive applications.
Below I have shown the graph that compares the time taken by my implementation & standard python API on various sizes to find inverses. Red line represents time taken by my code, & black line is the time taken by Numpy API.



1. I can speed the code up, by using Naive float type provided in Python, which I haven’t used for better precision right now.
2. Using vectorized implementation (instead of List type we can go for Numpy 2D array, on which all “MULTIPLY” & “MULTIPLY&ADD” row operations can be computed at very better speed).
3. Using parallel architecture available, we can speed up both vector implementation as well as multiple “MULTIPLY&ADD” in the rows below the row having current pivot can be computed in parallel.

Problem 2 (Third Part)

Row Operations & Column Operations are equivalent in terms of finding Inverses, (i.e., Transpose the matrix, apply Row operations & then do transpose again. Performed Row operations will be same as Column operations that can be done in directly without these Transposes.)

$$\mathbf{(A^T)^{-1}==(A^{-1})^T}$$

We know that if matrix A has an inverse, its Transpose also has an inverse. As Rank(A) == Rank(A^T) & Existence of Inverse implies Rank(A)==N, where (N,N) is Size of Square Matrix given

Problem 2 (Bonus Question)

Only the Swap operation can be be replaced by using other two operations left.
Suppose R[i] & R[j] need to be swapped, we can do it the way we do swapping without a extra variable.

R[i] = R[i] + R[j]
R[j] = R[i] - R[j]
R[i] = R[i] - R[j]