CSE101: Home Assignment 4

The assignment involves writing program to perform matrix operations. Your task would be to find the rank of a matrix. Before proceeding to the implementation specifications of the assignment, you should go through some definitions and examples given below, for better clarity.

Background

Linear dependence
A set of vectors is said to be linearly dependent if one of the vectors in the set can be defined as a linear combination of the others; if no vector in the set can be written in this way, then the vectors are said to be linearly independent. You can represent a row or a column in a matrix as a vector.

Rank of a matrix
Rank of a matrix A of dimensions M x N is defined as
(a) Maximum number of linearly independent column vectors in the matrix(known as column rank) or
(b) Maximum number of linearly independent row vectors in the matrix(known as row rank)

Note: Our focus in this assignment would be on finding the row rank of the matrix. However, an interesting fact is that the column and row ranks of any matrix are equal. You may look up proof of it for an extra reading.

Example

```
Consider the following matrix, A:

 {{10,   20,   10},
  {20,   40,   20},
  {30,   50,   0}}

Rank of A is 2
Explanation: Ist and IInd rows are linearly dependent.
             But Ist and  3rd or IInd and IIIrd  are
             independent.


Consider the another matrix, B:

{{10,   20,   10},
 {-20, -30,   10},
 {30,   50,   0}}

Rank of B is 2
Explanation: Ist and IInd rows are linearly independent.
             So rank must be atleast 2. But all three rows
             are linearly dependent (the first is equal to
             the sum of the second and third) so the rank
             must be less than 3.
```

From these examples, we may infer that rank of a matrix is the largest order of any non-zero minor in the matrix where order of a minor is the side-length of the square sub-matrix of which it is determinant.

So if M < N then maximum rank of A can be M else it can be N. In general, rank of matrix can't be greater than min(M, N). The rank of a matrix would be zero only if the matrix had no non-zero elements. If a matrix had even one non-zero element, its minimum rank would be one.

Applications of Rank Calculation of a matrix

Rank calculation finds an interesting application in computing the number of solutions to a system of linear equations[.For extra reading: Rouché–Capelli theorem]
Another application: If we know that a matrix is of low rank, then we can compress and store the matrix and can do efficient matrix operations using it. The idea can be extended for any linear operator and these in fact form the basis for various compression techniques. [For extra reading: Low Rank Approximation]

How to find Rank?

The idea is based on conversion to Row echelon form.

```
1) Let the input matrix be mat[][].  Initialize rank equals
   to number of columns

2) Do following for row = 0 to rank-1.

  a) If mat[row][row] is not zero, make all elements of
     current column as 0 except the element mat[row][row]
     by finding appropriate multiplier and adding a
     multiple of row 'row'

  b) Else (mat[row][row] is zero). Two cases arise:
       (i) If there is a row below it with non-zero entry in
           same column, then swap current 'row' and that row.
       (ii) If all elements in current column below mat[row][row]
            are 0, then remove this column by swapping it with
            last column and  reducing number of rank by 1.
     Reduce row by 1 so that this row is processed again.

3) Number of remaining columns is rank of matrix.
```

Example:

```
Input:  mat[][] = {{10,   20,   10},
                   {-20, -30,   10},
                   {30,   50,   0}}

row = 0:
Since mat[0][0] is not 0, we are in case 2.a of above algorithm.
We set all entries of 0'th column as 0 (except entry mat[0][0]).
To do this, we subtract R1*(-2) from R2, i.e., R2 --> R2 - R1*(-2)
         mat[][] = {{10,   20,   10},
                    { 0,    10,   30},
                     {30,   50,   0}}
            And subtract R1*3 from R3, i.e., R3   --> R3 - R1*3
         mat[][] = {{10,   20,   10},
                    { 0,   10,   30},
                    { 0,  -10,  -30}}
```

```
row = 1:
Since mat[1][1] is not 0, we are in case 2.a of above algorithm.
We set all entries of 1st column as 0 (except entry mat[1][1]).
To do this, we subtract R2*2 from R1, i.e., R1 --> R1 - R2*2
         mat[][] = {{10,    0,  -50},
                    { 0,   10,   30},
                    { 0,  -10,  -30}}
           And subtract R2*(-1) from R3, i.e., R3   --> R3 - R2*(-1)
         mat[][] = {{10,   0,   -50},
                    { 0,   10,   30},
                    { 0,   0,     0}}

row = 2:
Since Since mat[2][2] is 0, we are in case 2.b of above algorithm.
Since there is no row below it to swap. We reduce the rank by 1 and
keep row as 2.

The loop doesn't iterate next time because loop termination condition
row <= rank-1 returns false.
```

Task 1

Create a module, **a4.py,** and define the functions as described in the subtasks.

Subtask 1

Define the matrix function, **swapRows(A,row1,row2)**, that performs a swap between two rows of a given matrix.
The function takes 3 arguments:

- A represents the matrix
- row1 and row2 are indices of the rows to swap. The possible values for the arguments *row1* and *row2* are *0, ..., nrows - 1,* where nrows in number of rows in A

Sample:
A=[[1,2,3],[4,5,6],[7,8,9]]
swapRows(A,0,2) results in A changed to [[7,8,9],[4,5,6],[1,2,3]]

Subtask 2

Define a matrix function, **Row_Transformation(A, x, row1, row2)**, that performs row transformation on matrix A by transforming as

row2 → row2 + x * row1

Preconditions:

- A is a matrix containing integers
- x is of type double
- The possible values for the arguments *row1* and *row2* are *0, ..., nrows - 1,* where nrows in number of rows in A

Sample:
A=[[1,2,3],[4,5,6],[7,8,9]]

Row_Transformation(A,3,0,2) results in A changed to [[1,2,3],[4,5,6],[10,14,18]]

Subtask 3

Using the algorithm described in the previous section, define a function that supports finding rank of a given matrix. In the module define a function **MatrixRank(A)** that

- takes a nested list A(to represent a matrix) as an argument and
- returns an integer representing the rank of A
- Preconditions: A is a matrix containing integers

You may use the functions defined in other subtasks.

Task 2

In another module, **testa4.py**, define 5 extensive testcases to test your implementation of MatrixRank(A) function.

Submission

- Mention name and roll number of all team-mates, as comments, in both modules.
- Zip a4.py and testa4.py and submit on the Backpack deadline.
- Adhere to the deadline. No submission shall be accepted over email.