Assignment – 1

Ashish Kumar Jayant
M.Tech(Research), CSA

Problem - 1 Task 2

Let's assume a matrix **A** of size m*n and its row echeleon form be **B** over field **R** (Real numbers).

Column Space of a Matrix – Column space of a matrix is equal to span of the columns vectors of a matrix A. It is a subspace of $\mathbf{R}^m$.
When we perform a row tranformation on a matrix to get echeleon form it **can change** its column space because suppose a matrix has no non zero row to begin with and nth row becomes zero in row echeleon form then span of the columns will not include any non zero value of the nth component of columns vector for any linear combination.

Row Space of Matrix – Row space of a matrix is equal to span of the row vectors of a matrix. It is subspace of $\mathbf{R}^n$.
Row space does not change because even if a row becomes zero in echeleon form it doesn't change span of row vectors.

Null Space of Matrix – It is a set of vectors x which satisfy -
Ax = **O**
where **A** is m*n, **x** is n*1 and **O** is zero vector of m*1 size.
Null space of matrix also does not change as solution set of A and B remains same. Another way to look at is that determinant doesn't change with row operations.

For e.g -

1	0	2
1	1	1
1	1	-1

Column space = span([1 1 1], [0 1 1], [2 1 1]) = $\mathbf{R}^3$
Row space = span([1 0 2], [1 1 1], [1 1 -1]) = $\mathbf{R}^3$
Null space = non trivial solutions since det(A) = 0

R2 -> R2 – R1, R3 -> R3 – R2, R3 -> R3+ 2*R1 :

1	0	2
0	1	-1
0	0	0

Column space = span([1 0 0], [0 1 0], [2 -1 0]) = [p q 0] for all p,q belongs to R (not whole $\mathbf{R}^3$)
Row space = span ([1 0 2], [0 1 -1]) = $\mathbf{R}^3$
Null Space = non trivial solutions as still det(A)=0

Problem – 2 Task 2

Worst case time complexity of algorithm is **$O(n^3)$** as it uses Gauss Jordan Elimination to find row echeleon form. Other calls in algorithm have less time complexity than this method so this gives the upper bound.

```
def gauss_jordan(a,m,n,E):
    pivot = 0
    i=0
    for j in range(n):                              //n
        l = [bool(k) for k in a[i]]
        try:
            pivot = l.index(True)
            for i in range(m):                          //m    *(m=n in our case)
                if i>j and a[i][j]!=0:
                    if a[pivot][j]==0:
                        a=row_exchange(a,pivot,i)
                        I=give_me_identity(n)
                        I=row_exchange(I,pivot,i)
                        E.append(I)

                    else:
                        c = a[i][j]/a[pivot][j]
                        for p in range(n):                          //n
                            a[i][p] = a[i][p] - c*a[pivot][p]
                        I=give_me_identity(n)
                        I[i][pivot]=-c
                        I[i][i]=1
                        E.append(I)



        except:
            pass
    return a,E
```

We observe three nested loop here going from (0 to n) , hence time complexity is $O(n^3)$