Submission for Project-I

Name: Amish Gupta

ID: 2019B5AA1386H

Course Title: Applied Stochastic Processes

Course Code: Math F424

Q7(iii) Write a program that determines whether a transition matrix corresponds to a regular Markov chain.

```
Answer: We divide the problem into two parts:
1) We check that sum of each row = 1 and each element>=0; to verify that it is
a valid transition matrix.
2) We check T^n should be such that each element > 0 for some n, then it's
regular else not.

part 1) is fairly simple, we iterate through each row and check for the
conditions as the statement demands, this will take O(n^2) time as we go
through each element in a matrix of the order n.

part 2)
This is fairly complicated as the following question arises: "at what power of
the matrix we stop checking?"
Some other equivalent checks would be:
a)A stochastic matrix is regular if it's irreducible and has at least one non-
zero entry on its main diagonal.
b)lim m→∞ A^m exists and has rank 1
However, none of them are useful. To solve our problem we use the following
theorem:
Theorem: If A is an n×n matrix with nonnegative entries, and some power of A
has only positive entries, then A^q has only positive entries, where
q=n2−2n+2. This is sharp; there is a matrix for which A^(q−1) does not have
only positive entries.

The reference to the above is:
https://math.stackexchange.com/questions/450090/if-p-is-a-regular-transition-
probability-matrix-then-pn2-has-no-zero-ele.
Now, using logic:
p: If A is an n×n matrix with nonnegative entries, is a regular matrix.
q: then A^q has no zero element
Theorem: p->q is true.
Therefor ~q->~p = true. (It's contrapositive is also true)

Now to solve the A^q we use the method from question 7_i). Hence the time
complexity: O(n^3)
```

Type *Markdown* and LaTeX: α^2

In [11]:
```
#below is the implementation of part 1
def validTransition(matrix):
    for row in matrix:
        if len(row)<len(matrix) :
            return [False,0]
        if sum(row)!=1:
            return [False,1]
    return [True,1]
import numpy as np

def inverse(matrix):
    import numpy as np
    try:
        i=np.linalg.inv(matrix)
        return i
    except:
        print("Inverse of the given matrix not possible")

def multiply(A,B):
    ans = [[0 for i in range(len(B))] for j in range(len(A[0]))]
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                ans[i][j] += A[i][k] * B[k][j]
    return ans

def eigenvectorMatrix(matrix):
    eigenvalues, eigenvectors = np.linalg.eig(matrix)
    return eigenvectors

def diagnolMatrix(matrix,m):
    eigenvalues, eigenvectors = np.linalg.eig(matrix)
    for i in range(len(eigenvalues)):
        eigenvalues[i]=eigenvalues[i]**m
    D=[[0 if i!=j else eigenvalues[j] for i in range(len(matrix))] for j in range
    return D

def exponentOfMatrix(matrix,m):
    V=eigenvectorMatrix(matrix)
    D=diagnolMatrix(matrix,m)
    Vinv=inverse(V)
    answer=multiply(V,multiply(D,Vinv))
    return answer



#below is the implementation of part2
def isRegular(matrix):
    q=len(matrix)**2-2*n+2
    answer=exponentOfMatrix(matrix,q)
    for row in answer:
        for elements in row:
            if elements<=0:
                return "Not Regular"

    return "Regular"
```

```
    #checking is


n=int(input("input the order of the matrix"))
print("input matrix")
matrix=[]
for i in range(n):
    matrix.append(list(map(float,input().split())))


valid,error= validTransition(matrix)
if not valid and error==0 :
    print("Invalid matrix")
elif not valid and error==1:
    print("Not a valid transition matrix")
else:
    print(isRegular(matrix))
```

```
input the order of the matrix3
input matrix
0.7 0 0.3
0 1 0
0.2 0 0.8
Not Regular
```

In [ ]:

In [ ]: